III Bootcamp Full Stack Cibersecurity

Módulo 5 - Blue Team

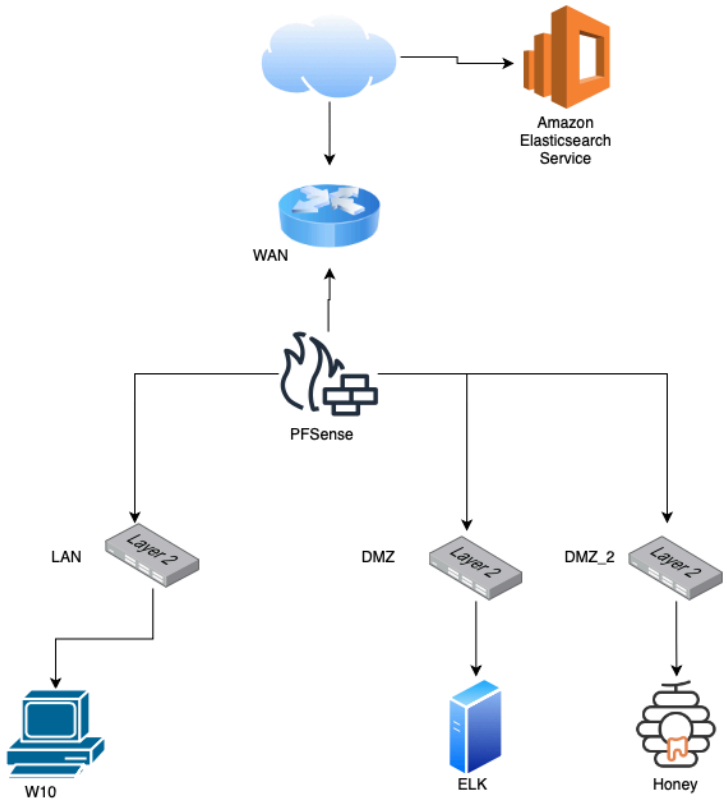
Caso práctico

Marcos Alonso González
alonsogonzalezmarcos@gmail.com
https://github.com/magalorn
18 de marzo de 2022

Índice

Enunciado del caso práctico

Queremos montar la siguiente infraestructura:



La cual debe cumplir los siguientes requisitos:

- Creación de un PfSense en bridge que conecte 3 redes, LAN, DMZ y DMZ2 estas como red interna.
- Un equipo en LAN (nombre IT) , un stack ELK en otro equipo DMZ y un honeypot en un tercer equipo que denominamos DMZ2.
- El IT debe correr Suricata y poder conectarse al ELK vía Kibana para mostrar los logs.
- El equipo DMZ2 debe alojar un honeypot sin acceso a las otras redes (solo para transmitir logs) pero si debe ser accesible desde la red WAN.
- El servidor ELK debe almacenar y mostrar los logs del honeypot.

*__Nota:__ Todos los equipos utilizados para la creación de la red interna (IT, DMZ y DMZ2) son distribuciones Kali Linux con SO Debian 64bit.

1. Proceso de creación de la infraestructura

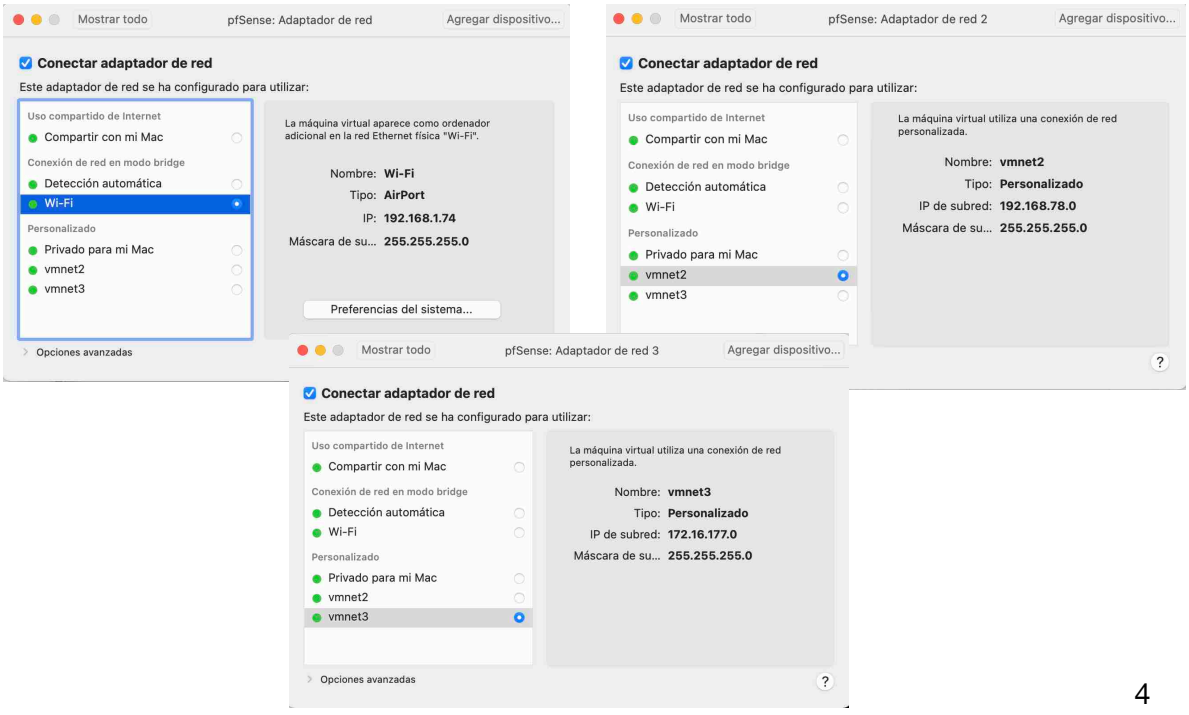
1.1. Creación de UTM y configuración de interfaces

Utilizamos una imagen ISO de **pfSense**, una distribución personalizada de FreeBSD adaptado para su uso como firewall y enrutador.

La importamos a VMWare con la siguiente configuración:

1. Se crean 2 tarjetas o adaptadores de red más, además de la existente por defecto

2. Se conecta la primera en modo Bridge a Wi-Fi y las otras dos a las redes personalizadas creadas (vmnet2 y vmnet3)



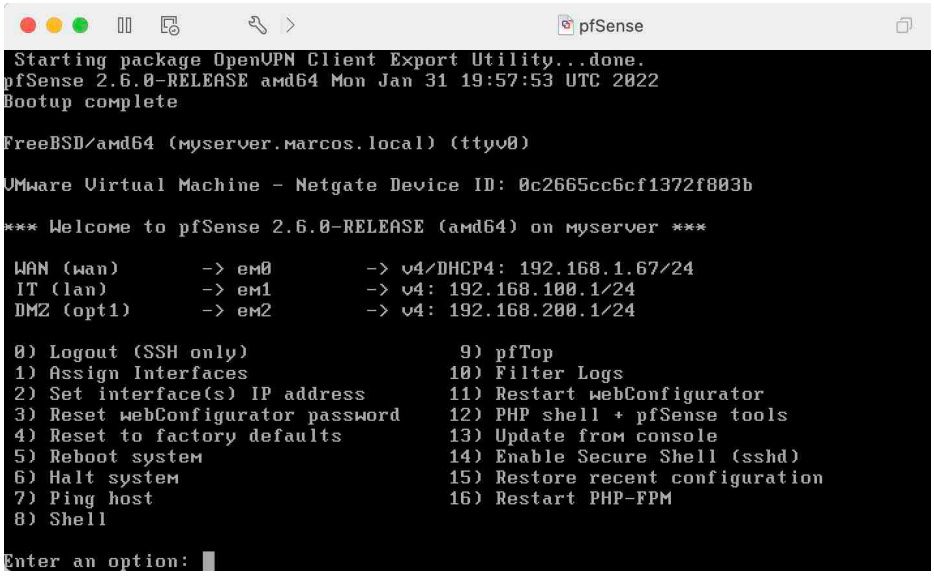
3. Se importa la imagen de disco de pfSense y se procede a la instalación siguiendo los pasos preconfigurados que marca el instalador.

4. Una vez instalado y reiniciado, se asignan las interfaces correspondientes a las. con la opción 1) Assign interfaces

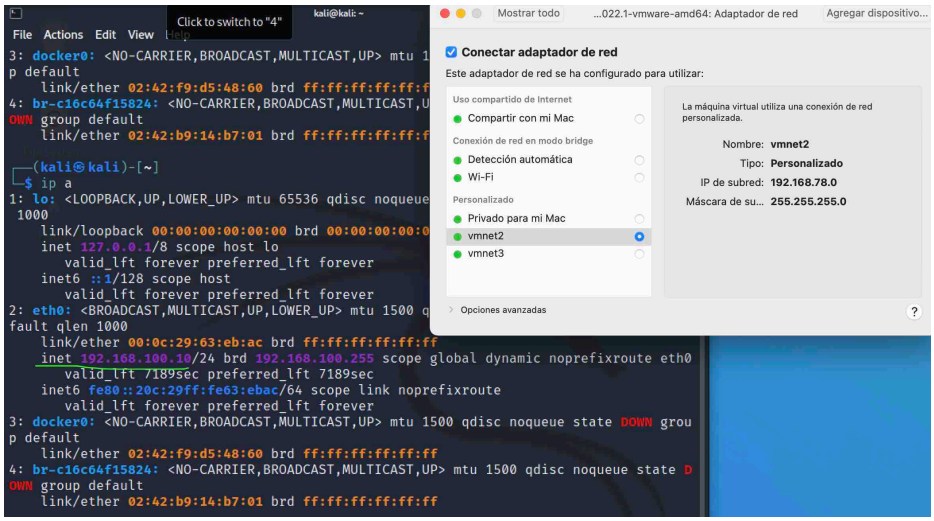
5. Se asigna la WAN a la em0, es la máquina por donde entraremos a internet

6. Se asigna la LAN a la interfaz em1, que hará de LAN o IT y será encargada del Firewalling y el modo NAT.

7. Se asigna la DMZ a la tercera red creada, la em2, como red opcional.

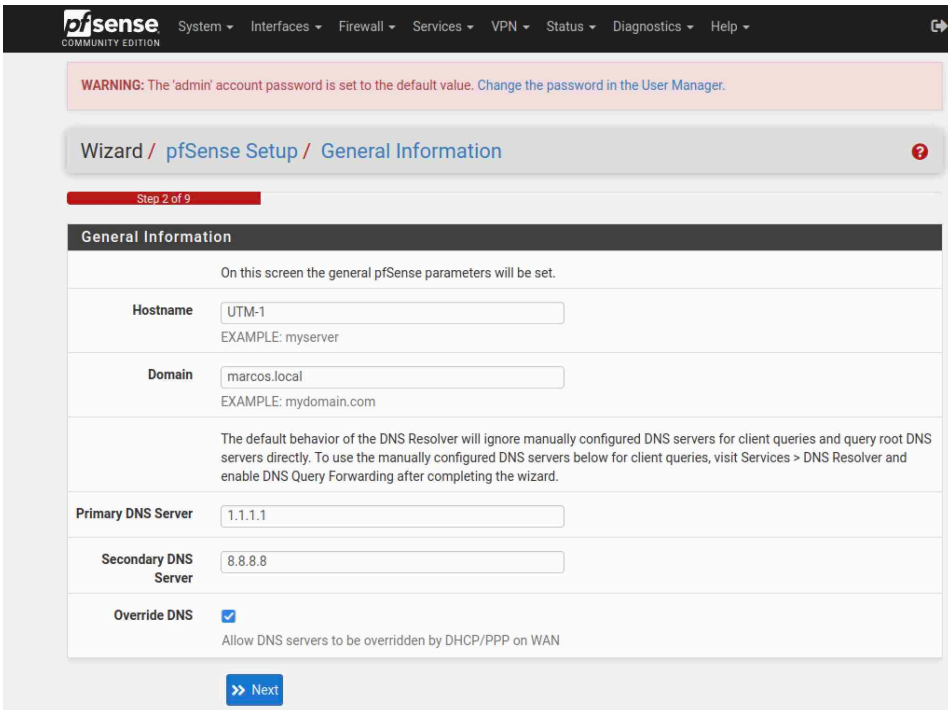


8. En la máquina Kali en que nos conectaremos a la interfaz de PfSense, seleccionamos el adaptador de red vmnet2, y la inet que se asigna a Kali será la 192.168.100.1

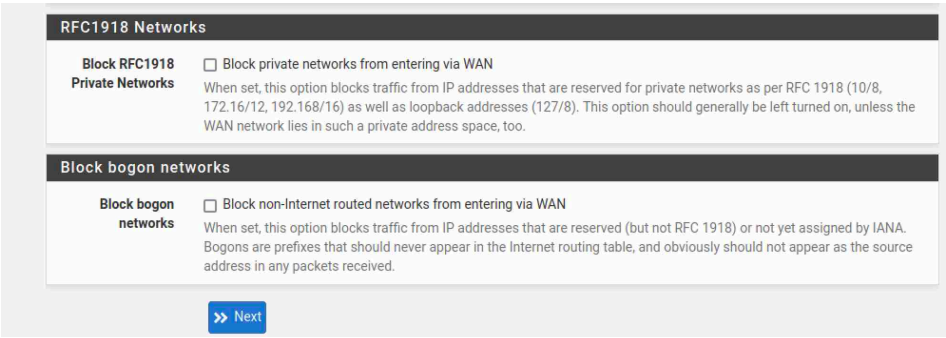


9. Vamos con el navegador a 192.168.1.1 y nos logueamos

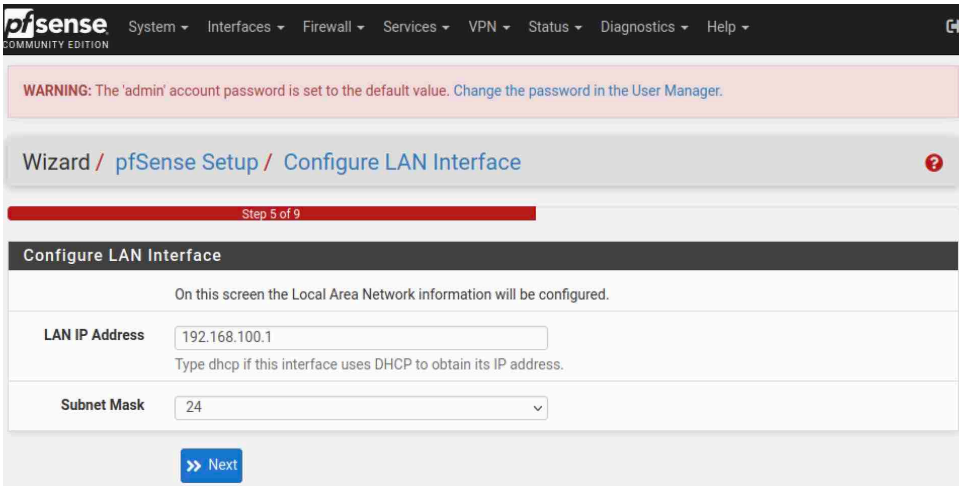
10. En la pantalla inicial de configuración se introducen estos valores para definir el hostname y el rango de dominio de nuestra red y los DNS de CloudFlare en este caso.



11. En la siguiente página se configura la interface WAN, se deja por defecto en DHCP. En esta misma página se desbloquean las redes RFC1918 y Bogon. Es una regla que declara cuales son las IPs publicas y las privadas, si esta activado no permite a las redes privadas salir afuera. Si pfSense como router o firewall está conectado directamente a internet hay que dejarlo seleccionado, como no está conectado a internet en este caso, se deja desbloqueado.



12. Seguidamente se configura la interfaz LAN, con la IP 192.168.100.1 y la subnet mask /24.

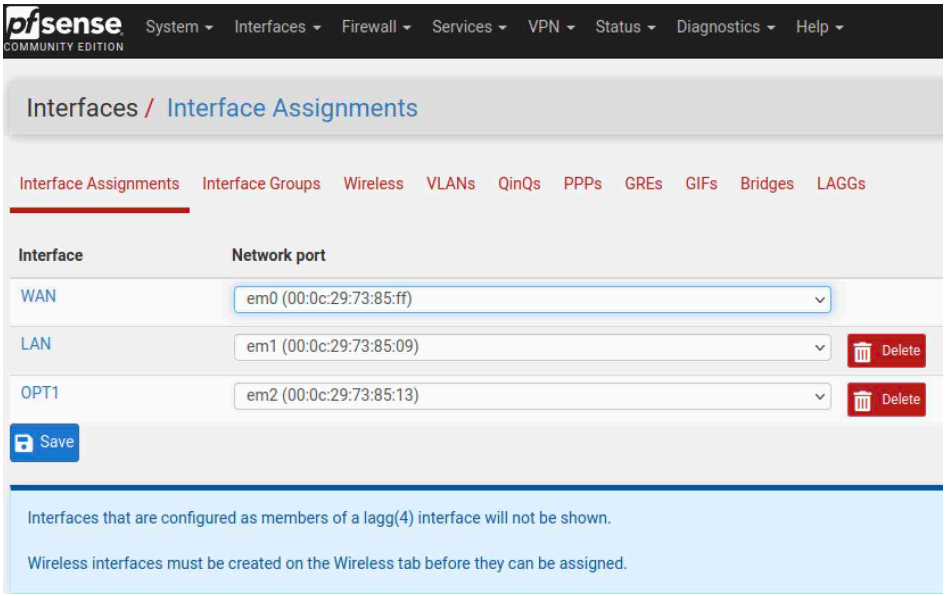


13. Finalmente se cambia la password por defecto y se recarga la herramienta.

14. Se desconecta la Kali de la red y se vuelve a conectar para que haga efecto la configuración de las interfaces.

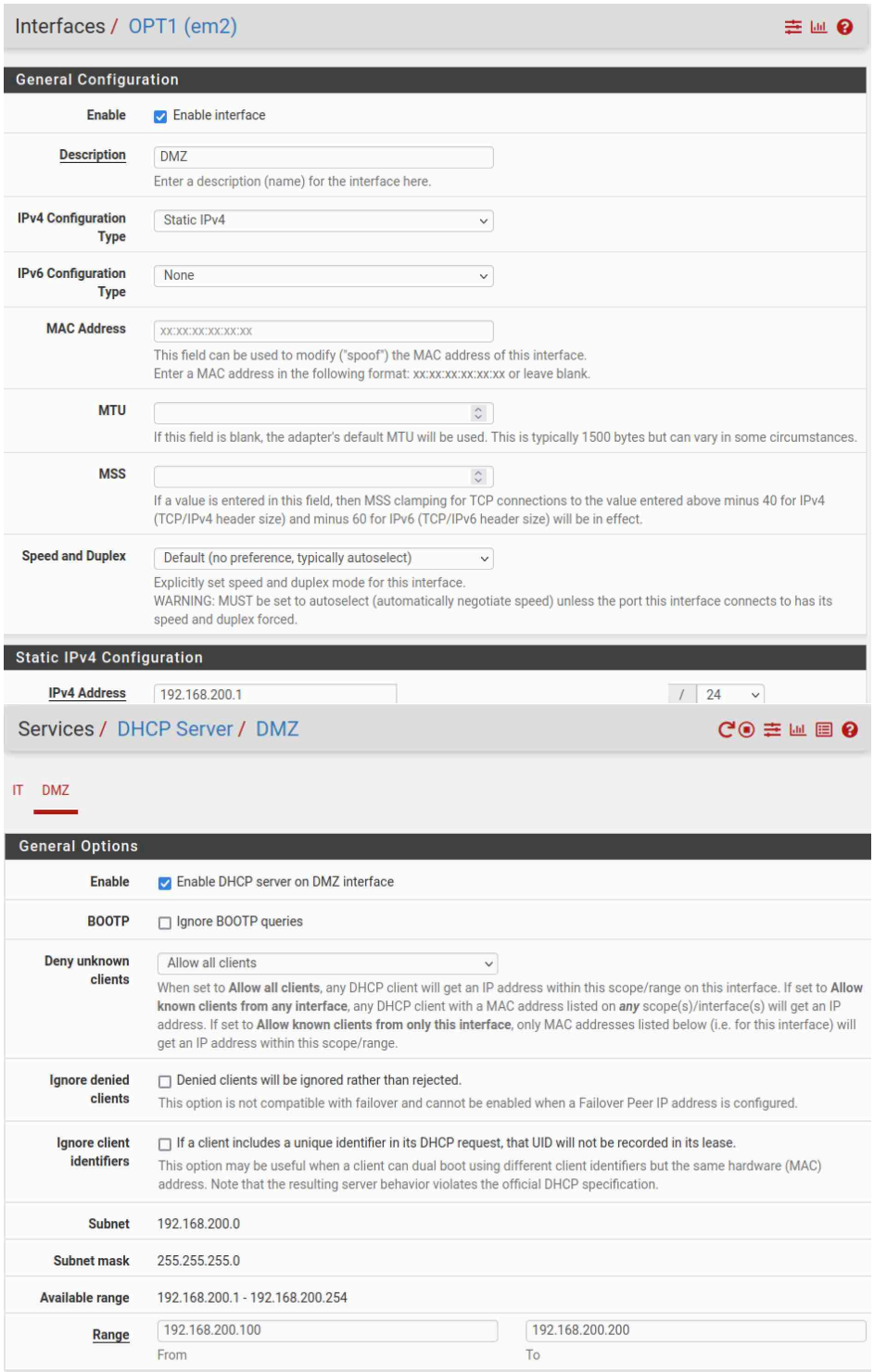
15. Navegamos a la IP de la LAN (192.168.100.1) y llegamos al Dashboard de la herramienta.

16. Por último, vamos a configurar la interface em2. En Interfaces/Interface Assignments, se selecciona y se añade la interface em2.

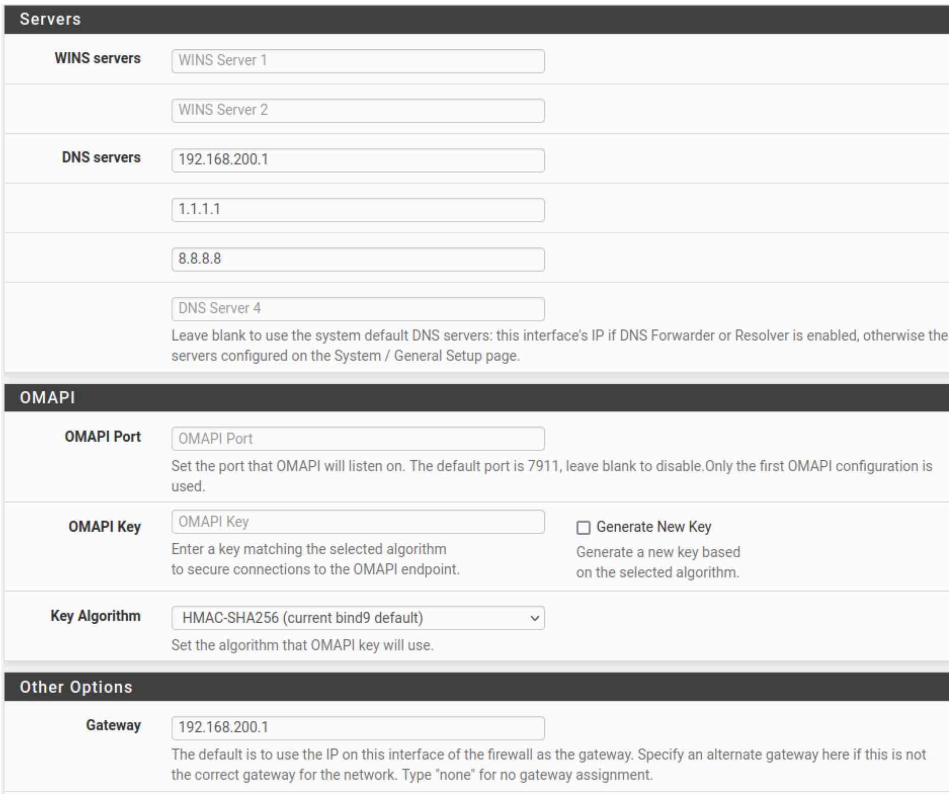


17. Después, seleccionamos la **interface LAN** (em1) y le cambiamos el nombre a **IT**.

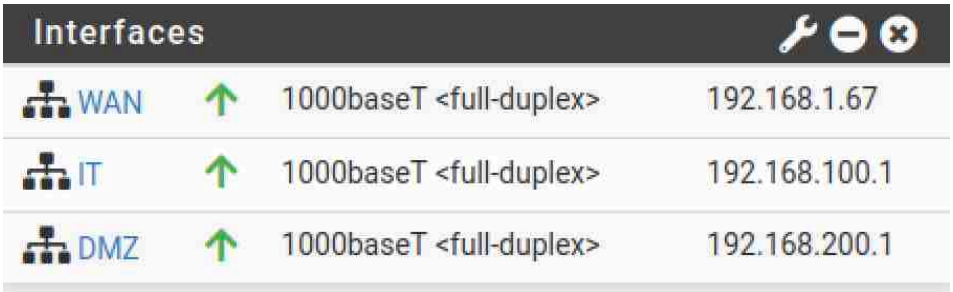
18. Lo mismo con **OPT1** (em2), se habilita, cambiamos el nombre a DMZ y le asignamos una IP estática. Después en Services/DHCP server, se habilita el DHCP para esta **interface DMZ** y se le asigna un rango de IPs. Seguidamente se configura el DNS servers a la misma IP que hemos puesto al propio servidor pfSense (192.168.200.1), y los DNS servers establecidos en el propio UTM porque el UTM-1 va a ser también un servidor DNS.



En Other options/Gateway establecemos la misma IP ya que también vamos a querer navegar por internet desde esta IP.



19. Comprobamos en el Dashboard que aparezcan las 3 interfaces con sus IPs correspondientes.



1.2. Creación de NAT

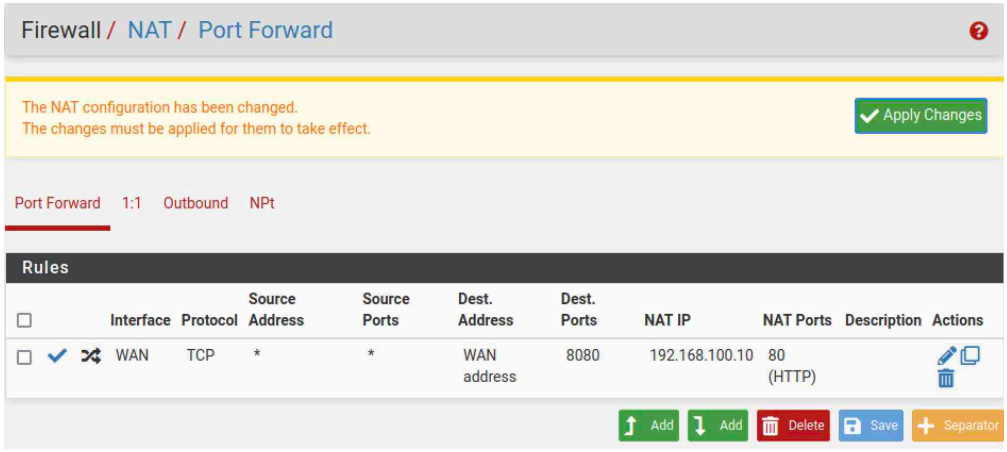
NAT permite acceder a un servicio que está dentro de una red interna, se va a crear a través de Port forwarding.

Los pasos a seguir serían:

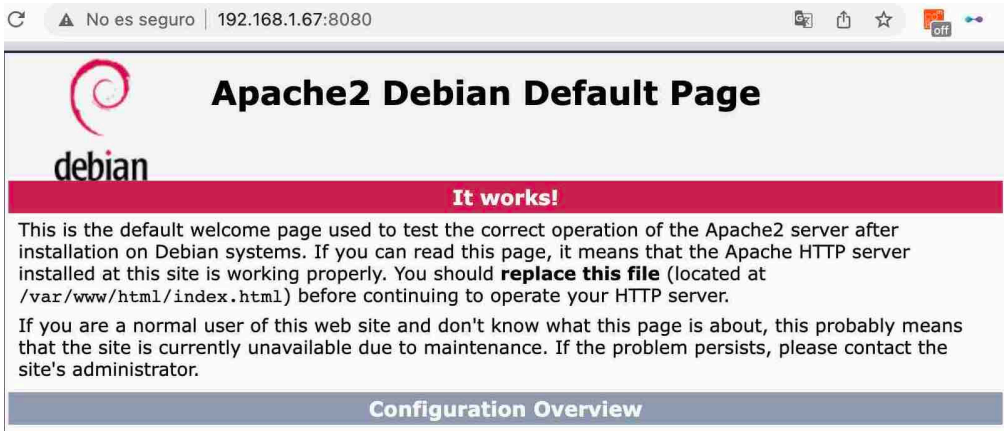
1. Levantar en nuestro Kali un servidor web con el comando

```
sudo service apache2 start
```

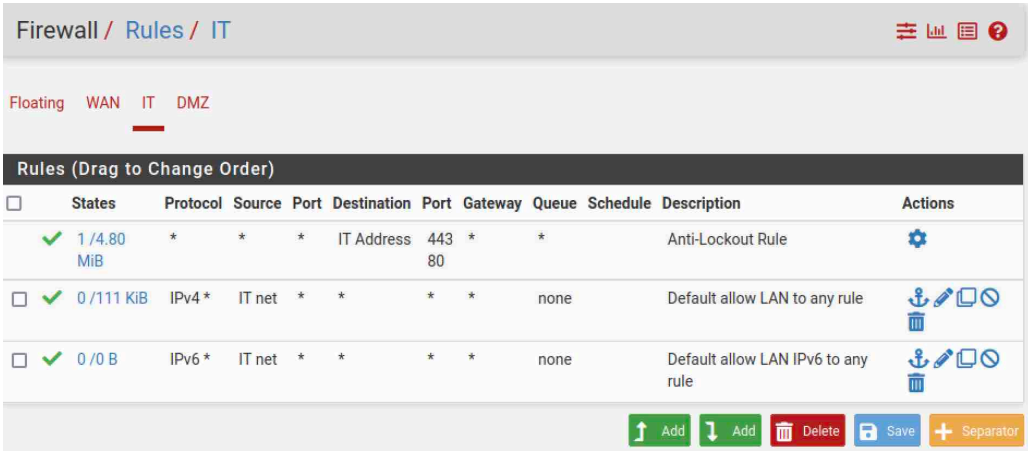
2. En pfSense, vamos a Firewall/NAT/Port Forward/Edit añadimos reglas, con destino WAN address puerto 8080, redirección a la IP de la Kali puerto 80. Esto significa que la IP que va a recibir la petición de nuestro localhost es la de la WAN (UTM-1) y cuando llegue esa petición, el firewall la va a reenviar al Kali (IT) por un puerto concreto, en el que está corriendo el servidor web apache2.



3. Se comprueba que desde cualquier navegador de nuestro localhost tenemos acceso a través de la IP:8080 WAN al servidor apache2 que corre en la IP:80 IT.



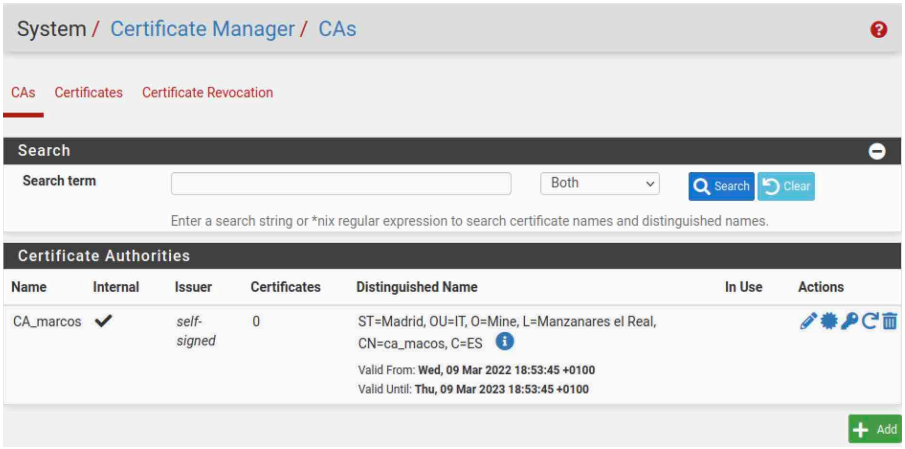
4. En Firewall/Rules/IT se pueden comprobar todas las reglas que pfSense ha creado automáticamente a partir de la configuración realizada.



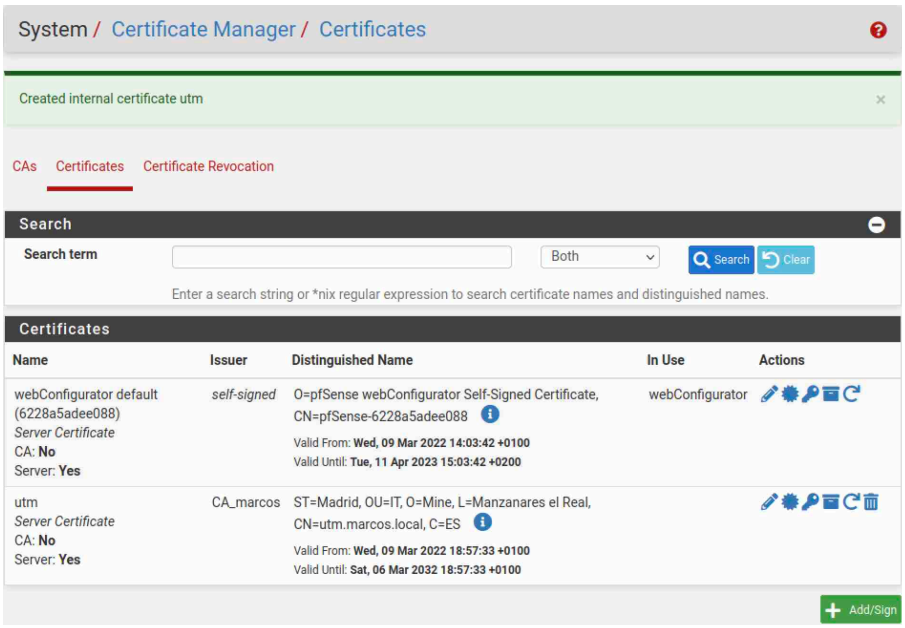
1.3. Creación de OpenVPN

Vamos a crear una VPN para que las conexiones entrantes y salientes del WAN a la red interna y viceversa pasen por el UTM y estén controladas.

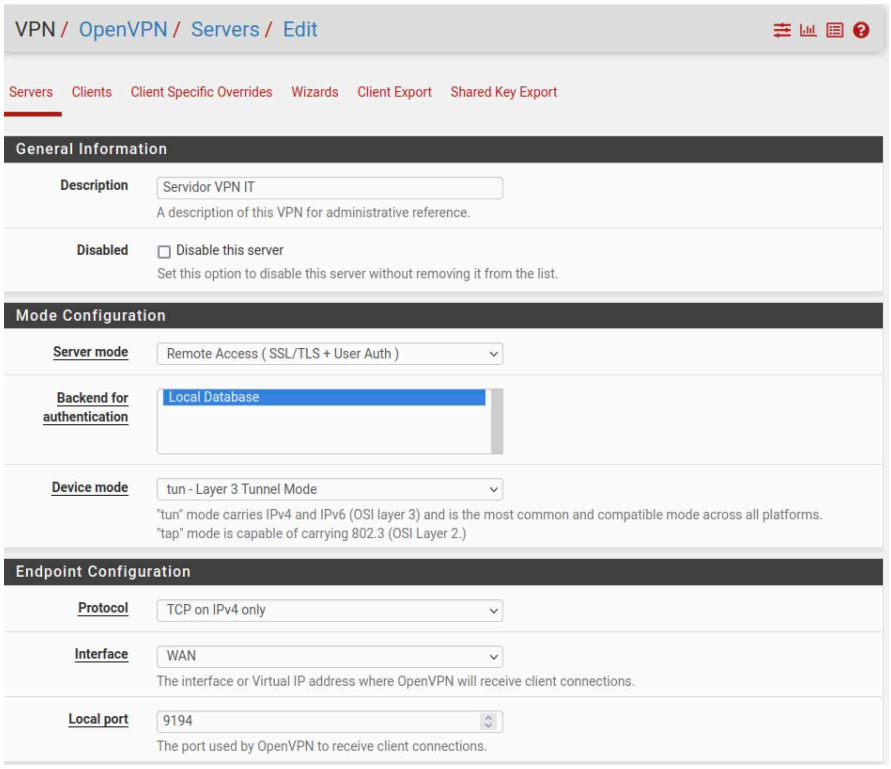
1. En pfSense, entramos en System/Package Manager/Available Packages
2. Buscamos e instalamos el paquete openvpn-client-export
3. Seguidamente vamos a System/Certificate Manager/CAs. Vamos a crear una entidad certificadora dentro de pfSense.
4. Añadimos una nueva autoridad certificadora con esta configuración

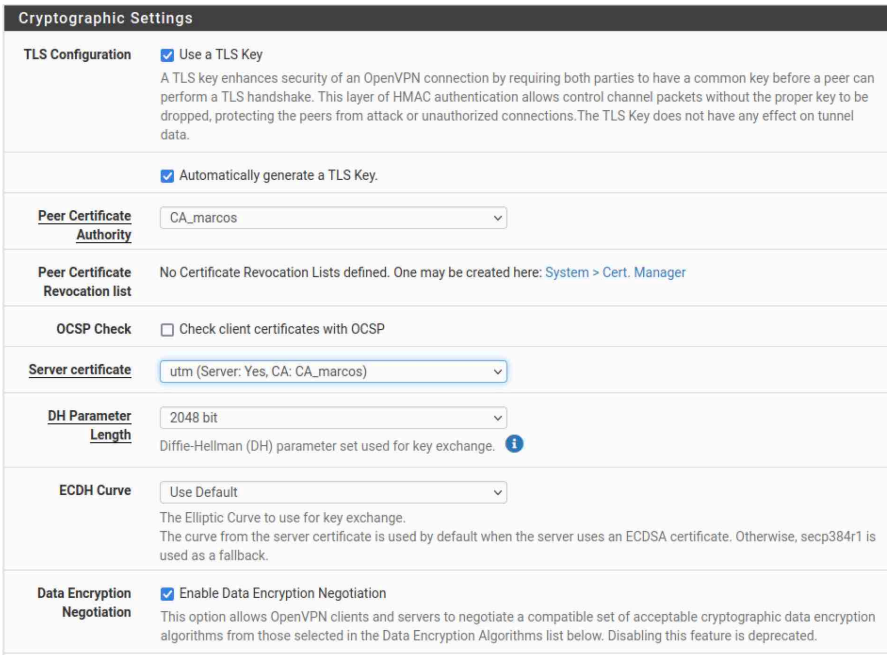


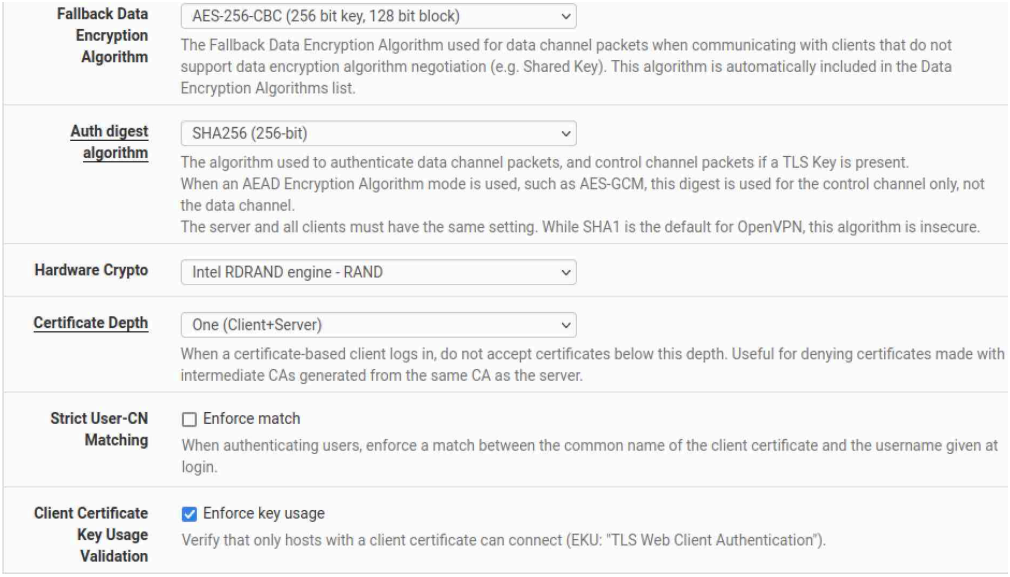
5. Vamos a System/Certificate Manager/Certificates y añadimos un nuevo certificado con la opción Server Certificate y esta configuración



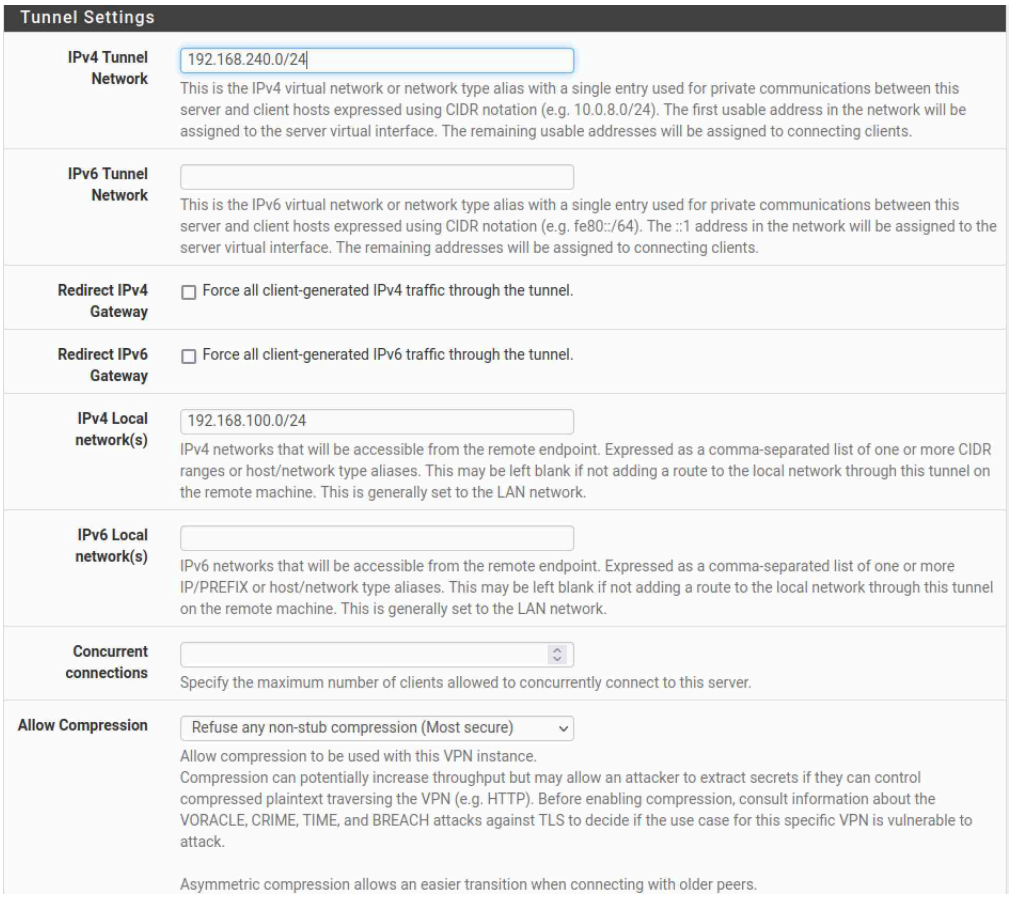
6. Seguidamente vamos a VPN/Open VPN/Servers para crear un servidor que va a ir a IT por la VPN. Seleccionamos Remote Access, doble factor de autenticación, por certificado y usuario y el server certificate UTM que hemos creado antes.



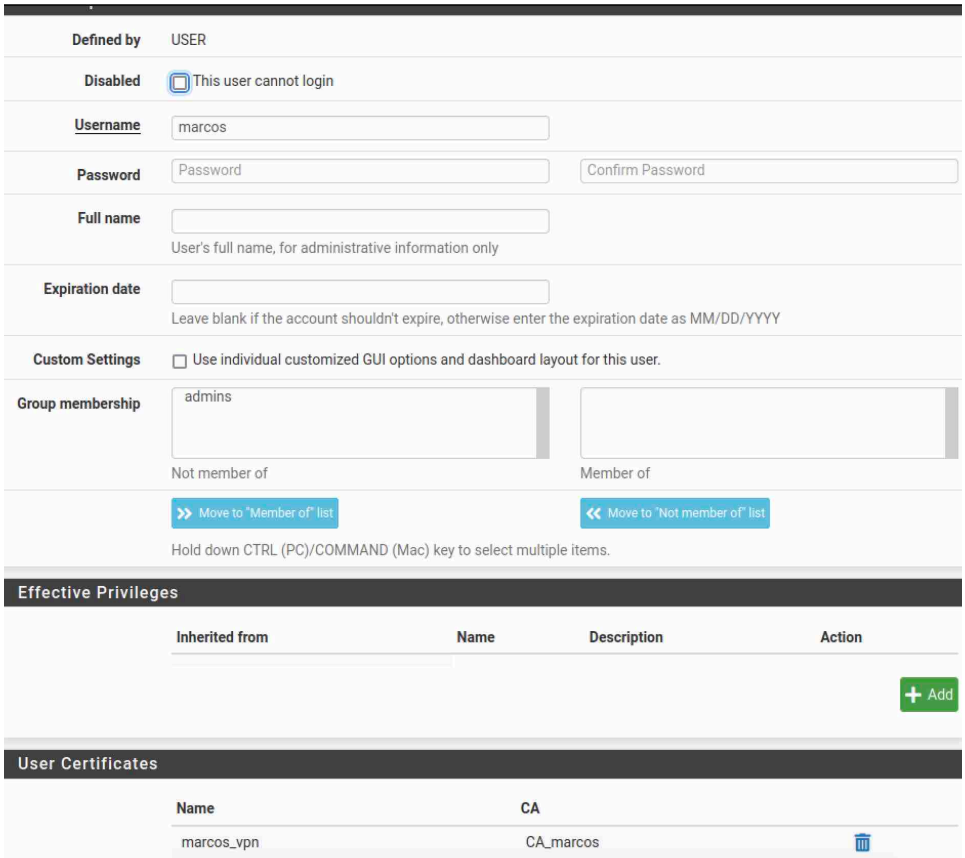




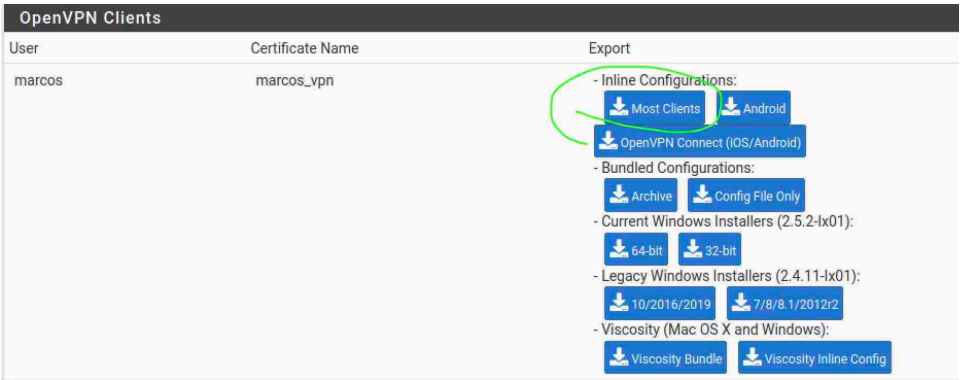
7. Importante también es la configuración de Tunnel Settings. Si se habilita aquí la opción Redirect IPv4/IPv6 Gateway, todo el tráfico que salga de localhost va a pasar a través del UTM, y si localhost sale a internet, lo hará con la IP publica de UTM. En este caso, se deshabilita y solo se incluye la opción IPv4 Local Networks, para que solo el tráfico que se dirige de localhost a IT vaya por VPN. El resto de las opciones las dejamos con están por defecto.



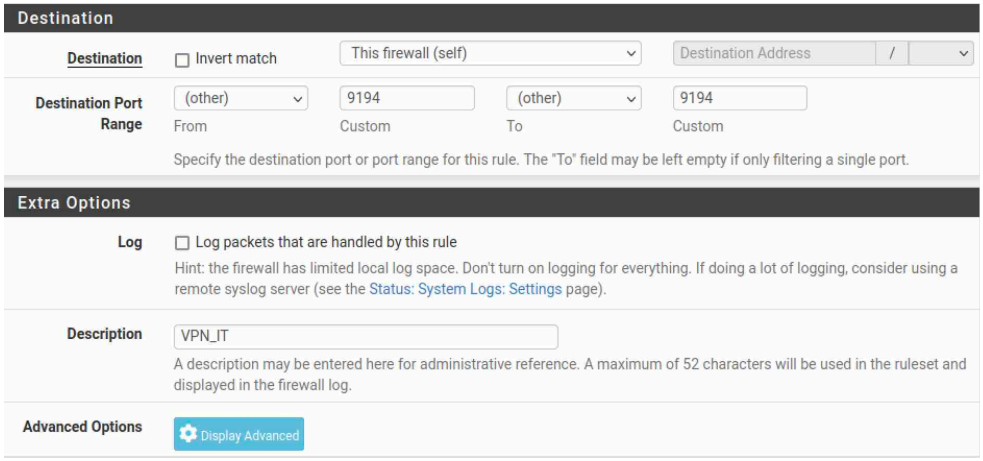
8. Seguidamente vamos a crear un usuario que se pueda conectar al tunnel network. Vamos a System/User Manager/Users y añadimos nuevo usuario donde lo importante es crear un certificado de usuario con el CA creado anteriormente. Comprobamos después en System/Certificate Manager/ Certificates que este certificado de usuario se ha creado correctamente.



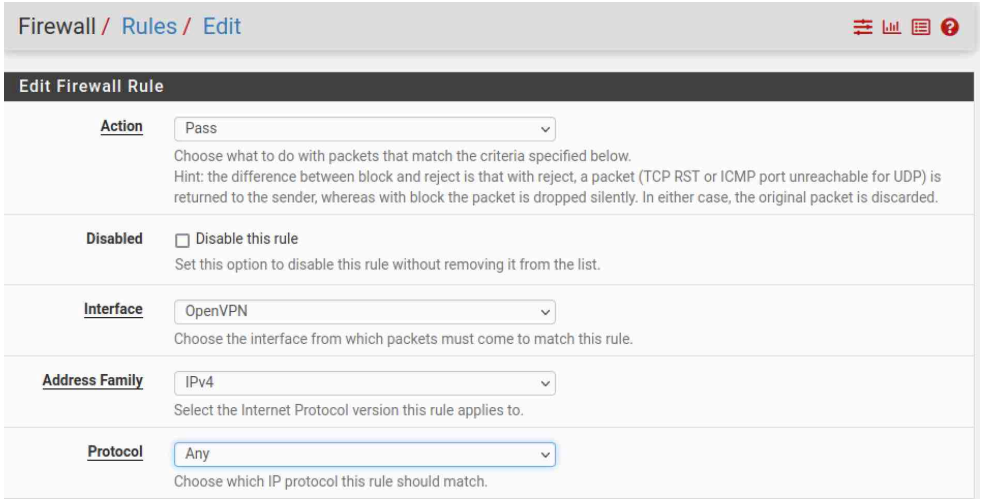
9. Después vamos o OpenVPN/Client export. Descargamos el archivo Inline configurations/Most clients y lo llevamos y abrimos en nuestro localhost.



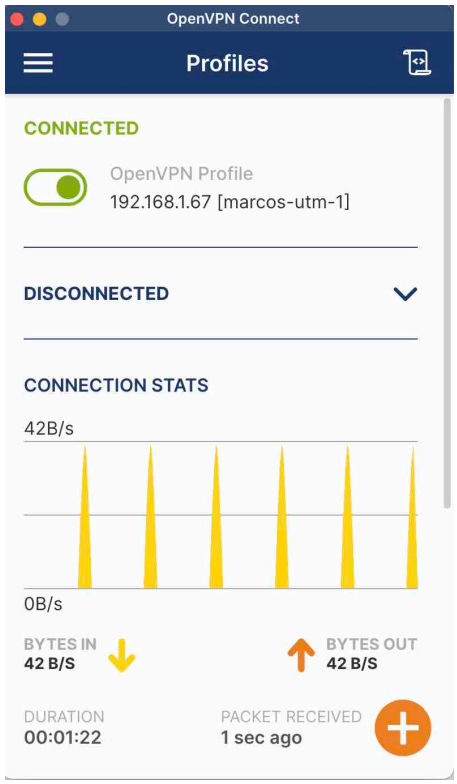
10. Antes de conectar nuestro localhost a la VPN, hay que crear una nueva regla en Firewall/Rules/WAN. Añadimos nueva regla, dejando todas las opciones por defecto excepto Destination y Extra options, que las configuramos así



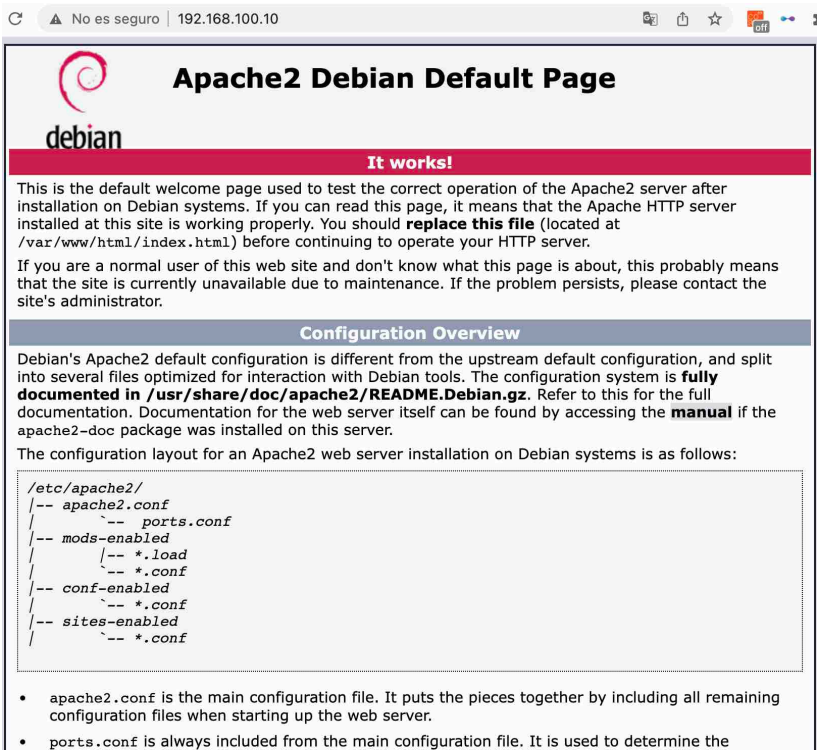
11. Y otra nueva regla en Firewall/Rules/OpenVPN. En este caso, la única opción a configurar es Edit Firewall/Rule/Protocol a Any.



12. Descargamos la aplicación OpenVPN Connect en nuestro localhost y arrastramos el fichero `.ovpn` para agregar al usuario que hemos creado en pfSense. Si todo está bien configurado, al conectar la VPN a este usuario, obtenemos este resultado



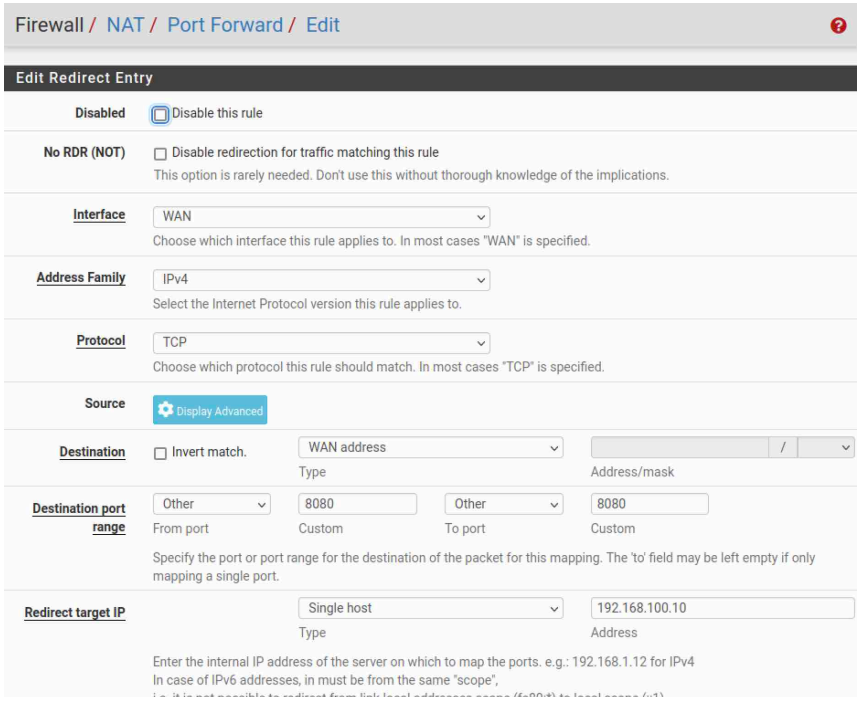
13. Comprobamos después que desde el navegador de localhost podemos alcanzar el servidor apache2 activo en la IP del IT (nuestra Kali en este caso)



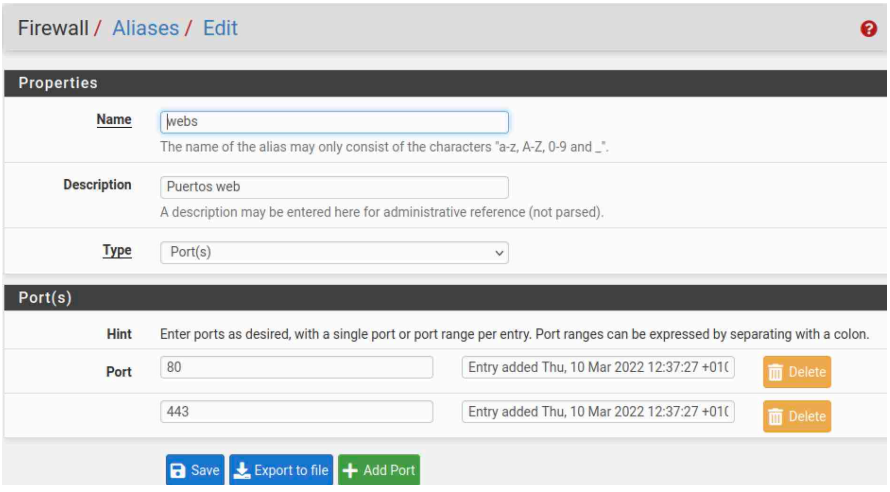
1.4. Reglas DMZ

Vamos a crear reglas para que la DMZ pueda conectarse a la WAN.

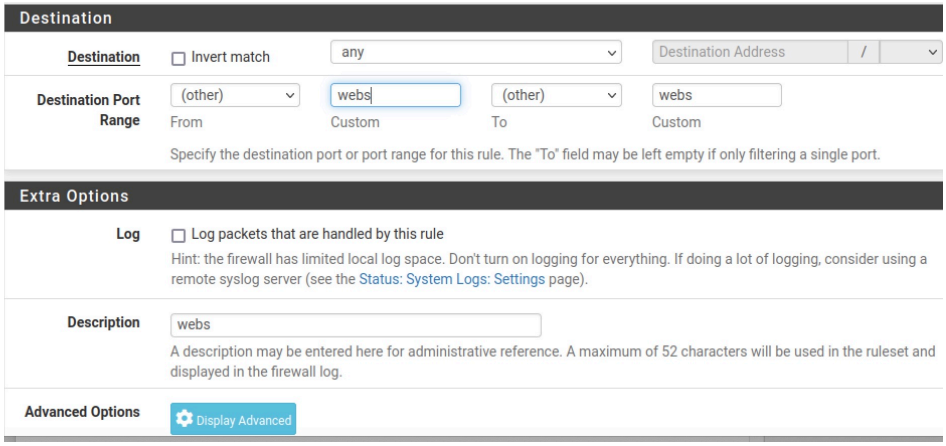
1. Primero creamos un nuevo NAT en Firewall/NAT/Port Forward con la siguiente configuración. Desde la interfaz WAN con destino a la IP de WAN y tráfico HTTPS, va a reenviar el tráfico al puerto 80. La IP a la que va a redirigir es la 192.168.200.103, que es la que habíamos creado para DMZ.



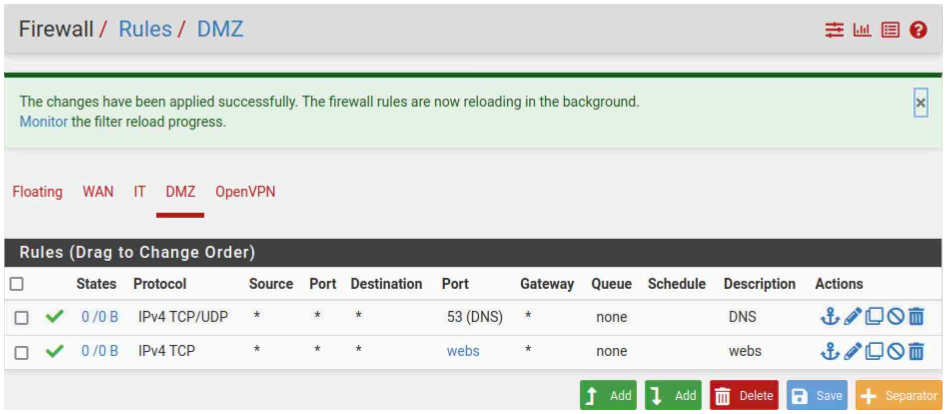
2. Después, creamos un alias en Firewall/Aliases/Ports. Un alias es un conjunto de reglas que se pueden agrupar. En este caso vamos a agrupar diferentes puertos web (80 y 443 TCP), para que se puedan agregar directamente a un grupo de equipos la misma regla.



3. Vamos luego a Firewall/Rules/DMZ y aplicamos este alias a las rules de DMZ



4. También creamos otra rule para el acceso de DMZ a DNS por el puerto 53 TCP/ UDP



5. Después se asigna la interfaz DMZ a otra máquina, en este caso, a otro Kali, colocando el adaptador de red en vmnet3. Para tener conexión en esta máquina, creamos una nueva regla en Firewall/Rules/DMZ para que permita conexiones ICMP.

2. Instalación de herramientas

2.1. Suricata

Suricata es una herramienta IDS (Intrusión Detection System), utilizada para controlar el tráfico de red. Permite rastrear eventos de seguridad que pueden indicar ataques o posibles intrusiones en equipos de la red.

1. En primer lugar, en la máquina Kali IT con usuario root instalamos Suricata con el comando

```
apt install suricata
```

2. Levantamos el servicio. Con la flag `-c` utilizamos el archivo de configuración por defecto y con `-i` la red o IP que se quiere escuchar, en este caso seleccionamos toda la red eth0. Comprobamos con system.ctl status que el servicio está activo y funciona con normalidad.

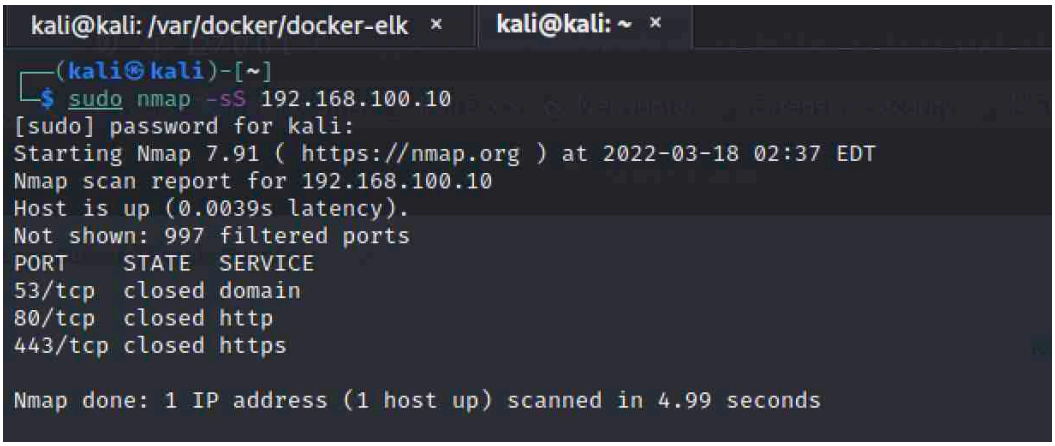
```
┌──(kali㉿kali)-[/var/log/suricata]
└─$ sudo suricata -c /etc/suricata/suricata.yaml -i eth0
[sudo] password for kali:
17/3/2022 -- 11:39:10 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
17/3/2022 -- 11:39:11 - <Notice> - all 4 packet processing threads, 4 management threads initialized, engine started.
^C17/3/2022 -- 11:54:53 - <Notice> - Signal Received.  Stopping engine.
17/3/2022 -- 11:54:54 - <Notice> - Stats for 'eth0':  pkts: 101, drop: 0 (0.00%), invalid chksum: 0

┌──(kali㉿kali)-[/var/log/suricata]
└─$ sudo systemctl start suricata.service
[sudo] password for kali:

┌──(kali㉿kali)-[/var/log/suricata]
└─$ sudo systemctl status suricata.service
● suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/lib/systemd/system/suricata.service; disabled; vendor preset: disabled)
     Active: active (running) since Thu 2022-03-17 11:55:10 EDT; 8s ago
       Docs: man:suricata(8)
             man:suricatasc(8)
             https://suricata-ids.org/docs/
    Process: 68301 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited, status=0/SUCCESS)
   Main PID: 68304 (Suricata-Main)
      Tasks: 10 (limit: 4572)
     Memory: 57.7M
        CPU: 3.007s
     CGroup: /system.slice/suricata.service
             └─68304 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

Mar 17 11:55:10 kali systemd[1]: Starting Suricata IDS/IDP daemon...
Mar 17 11:55:10 kali suricata[68301]: 17/3/2022 -- 11:55:10 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
Mar 17 11:55:10 kali systemd[1]: Started Suricata IDS/IDP daemon.
```

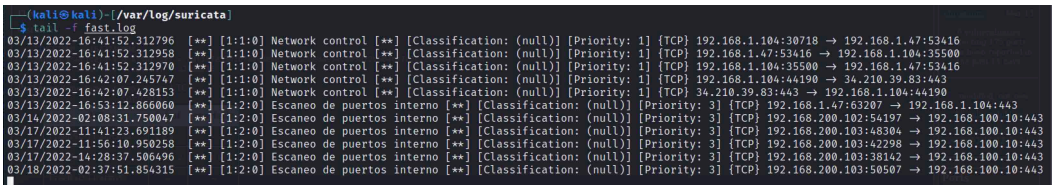
3. Vamos a editar el fichero suricata.yaml que está en /etc/suricata, dejando la configuración por defecto, creamos nuevo fichero de reglas

```
default-rule-path: /etc/suricata/rules

rule-files:
  - suricata.rules
  - reglas.rules
```

4. Vamos a /etc/suricata/rules para editar el archivo reglas.rules. Creamos una nueva regla que recoja los de todas las conexiones que se realicen en la red, que provengan de cualquier sitio.

5. Puede comprobarse su funcionamiento lanzando un escaneo nmap desde la máquina Kali DMZ. Podemos ver los logs recogidos por Suricata con `tail -f fast.log` en el directorio `/var/log/suricata`



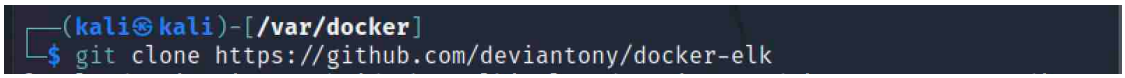


6. Dejamos activo el servicio para utilizarlo posteriormente en la ingesta y visualización de los logs en la máquina Kali DMZ con Elastic Stack.

2.2. Elastic Stack

Elastic Stack es una combinación de tres herramientas que sirven para recoger, procesar, almacenar, analizar y mostrar los datos de tráfico de red de un equipo o una red o sistema de equipos.

1. Vamos a realizar la instalación de Elastic Stack (ELK) con docker. En la máquina Kali en que lo vamos a instalar (DMZ) clonamos el repositorio de GitHub con el contenedor docker de ELK, situándonos previamente en una carpeta llamada docker que habremos creado previamente dentro de `/var`.



2. Dentro del directorio `/docker-elk`, instalamos Docker Compose y modificamos el permiso de ejecución

3. Levantamos el servicio Docker Compose y ya tenemos ELK corriendo

4. Accedemos a la interfaz web desde el navegador de esta maquina a http://127.0.0.1:5601. Nos logueamos con user: elastic y password: changeme.

3. Conexión entre equipos de la red interna

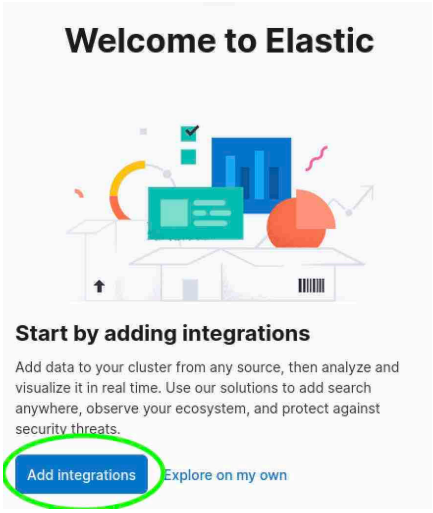
En este apartado se describen los pasos seguidos para las siguientes fases del ejercicio:

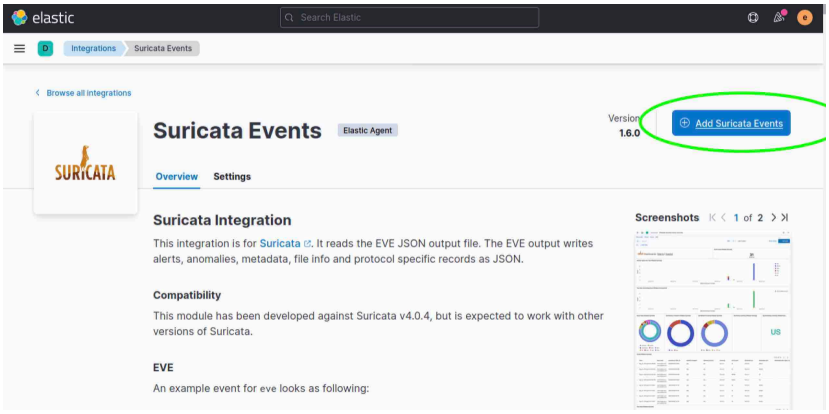
- El IT debe correr Suricata y poder conectarse al ELK vía Kibana para mostrar los logs.
- El equipo DMZ2 debe alojar un honeypot sin acceso a las otras redes (solo para transmitir logs) pero si debe ser accesible desde la red WAN.
- El servidor ELK debe almacenar y mostrar los logs del honeypot.

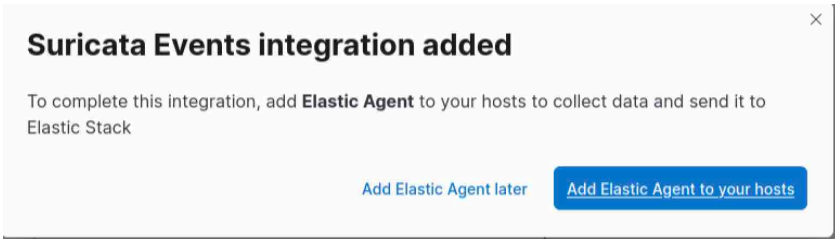
3.1. Integración de Suricata en Elastic Stack

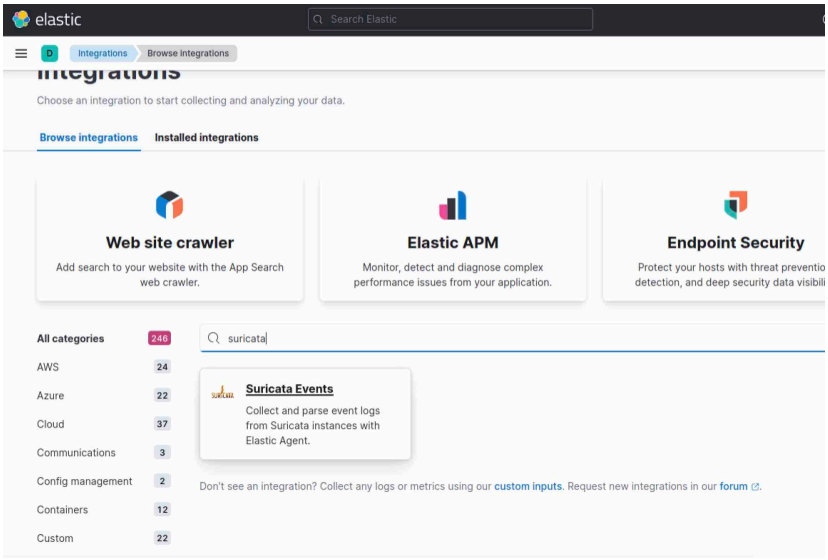
1. En primer lugar vamos a añadir la integración con Suricata en el ELK.

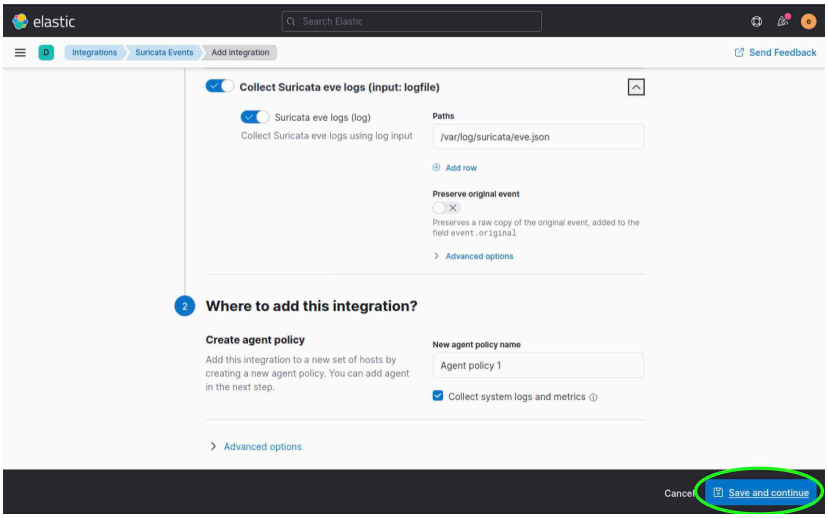
2. Lo podemos agregar con un Fleet, un equipo que va a agregando servicios o de manera individual, como va a ser el caso. Seguimos los siguientes pasos:



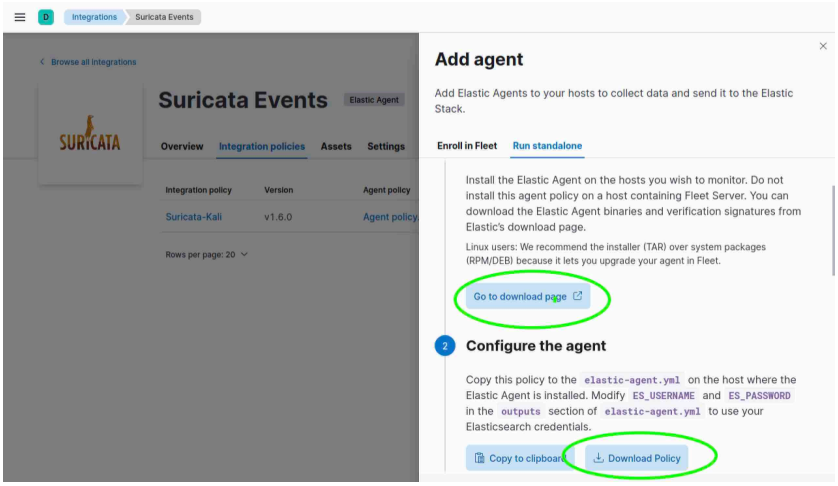

elastic
Search Elastic
Integrations
Suricata Events
Browse all integrations
Suricata Events
Elastic Agent
Version
1.6.0
Add Suricata Events
Overview
Settings
Suricata Integration
This integration is for Suricata. It reads the EVE JSON output file. The EVE output writes alerts, anomalies, metadata, file info and protocol specific records as JSON.
Compatibility
This module has been developed against Suricata v4.0.4, but is expected to work with other versions of Suricata.
EVE
An example event for eve looks as following:
Screenshots
1 of 2

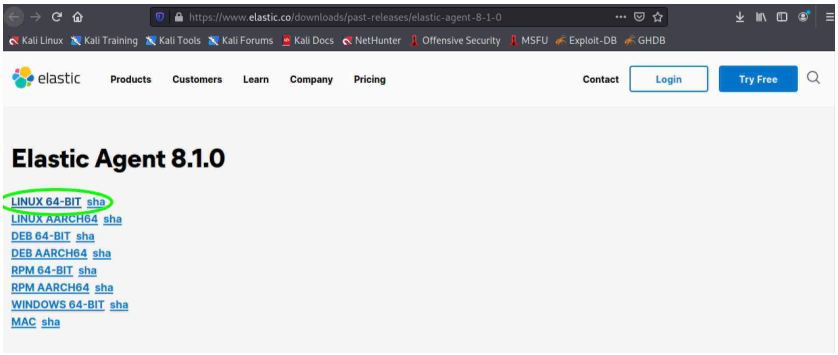

Suricata Events integration added
To complete this integration, add Elastic Agent to your hosts to collect data and send it to Elastic Stack
Add Elastic Agent later
Add Elastic Agent to your hosts


elastic
Search Elastic
Integrations
Browse integrations
Choose an integration to start collecting and analyzing your data.
Browse integrations
Installed integrations
Web site crawler
Add search to your website with the App Search web crawler.
Elastic APM
Monitor, detect and diagnose complex performance issues from your application.
Endpoint Security
All categories 246
AWS 24
Azure 22
Cloud 37
Communications 3
Config management 2
Containers 12
Custom 22
suricata
Suricata Events
Collect and parse event logs from Suricata instances with Elastic Agent.
Don't see an integration? Collect any logs or metrics using our custom inputs. Request new integrations in our forum.


elastic
Search Elastic
Integrations
Suricata Events
Add integration
Send Feedback
Collect Suricata eve logs (input: logfile)
Suricata eve logs (log)
Collect Suricata eve logs using log input
Paths
/var/log/suricata/eve.json
Add row
Preserve original event
Preserves a raw copy of the original event, added to the field event.original
Advanced options
2
Where to add this integration?
Create agent policy
Add this integration to a new set of hosts by creating a new agent policy. You can add agent in the next step.
New agent policy name
Agent policy 1
Collect system logs and metrics
Advanced options
Cancel
Save and continue

3. Abrimos la pagina de descarga del agente en otra pestaña del navegador y descargamos la Policy (archivo .yml).

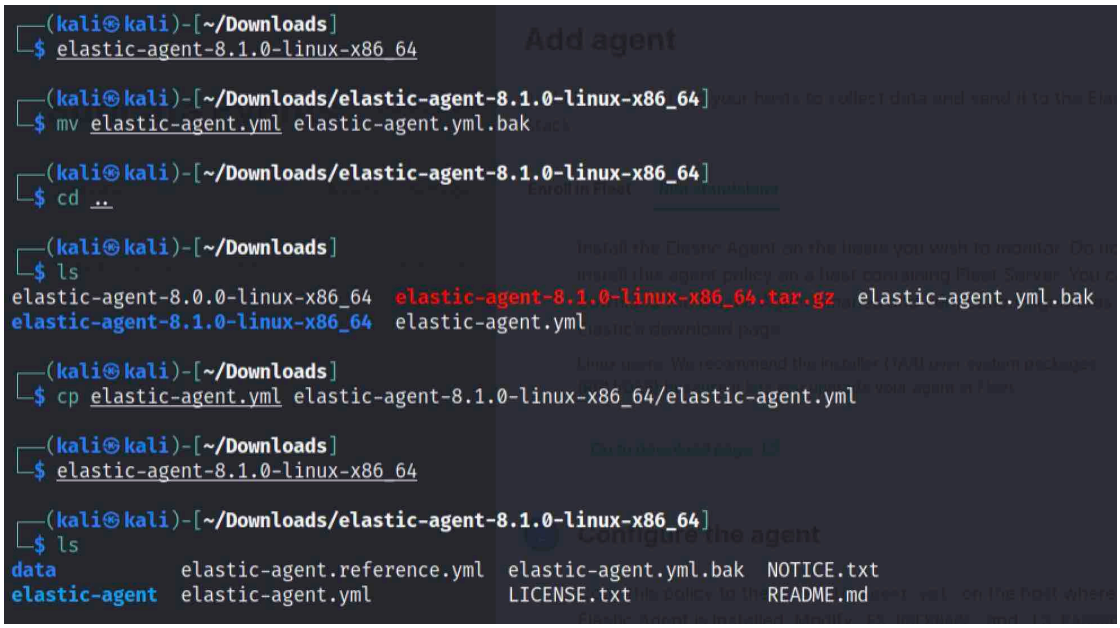




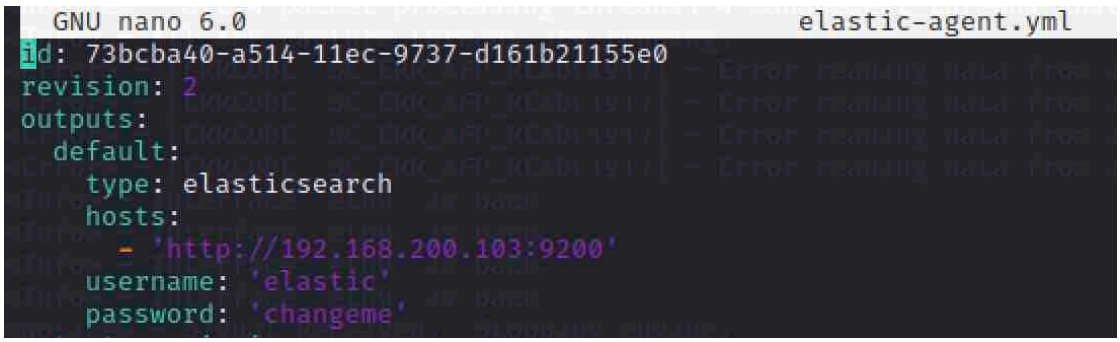
4. Nos llevamos los tanto el archivo Elastic Agent 8.1.0 descargado como el archivo .yml de la Policy al directorio Downloads del equipo Kali IT

5. Descomprimimos el archivo Elastic Agent 8.1.0 con

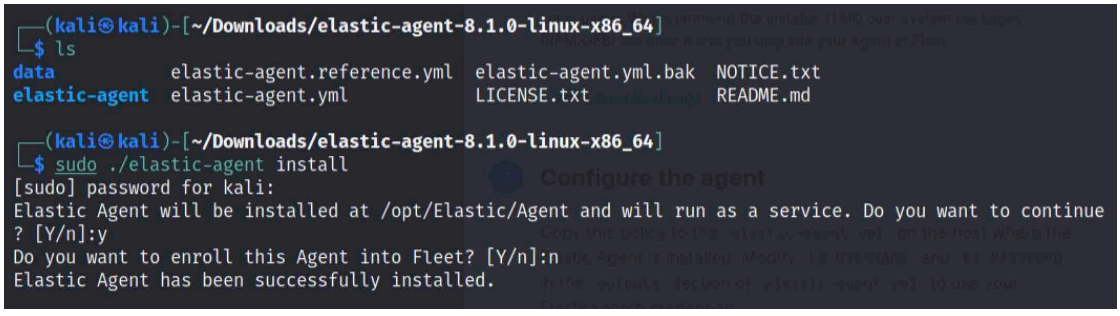
6. El archivo .yml de la Policy lo copiamos dentro del directorio del elastic-agent 8.1.0



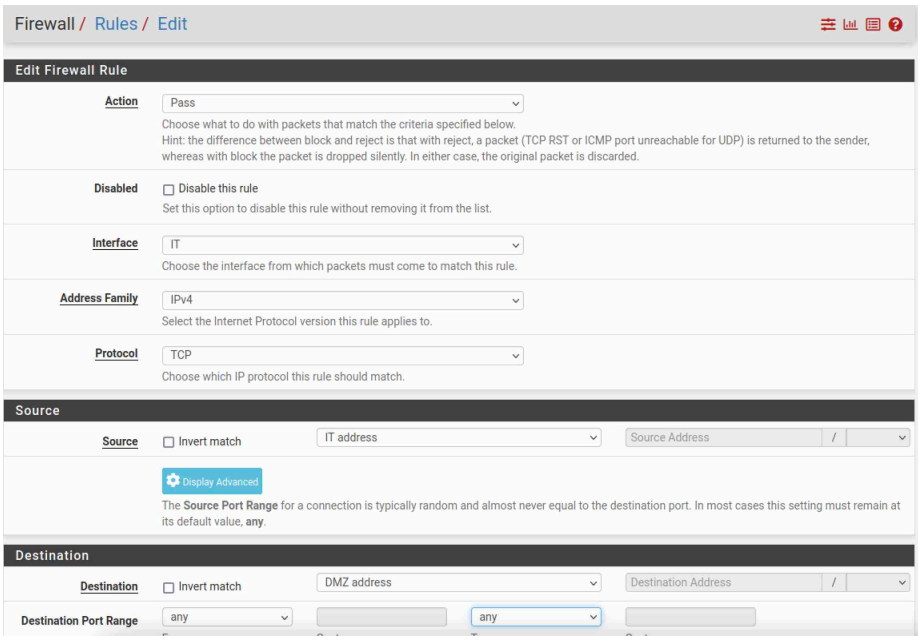
8. Con usuario root vamos a /opt/Elastic/Agent y modificamos el archivo elastic-agent.yml para introducir la IP de la máquina Kali (DMZ) que aloja el ELK y el usuario y contraseña de inicio de sesión del Stack.



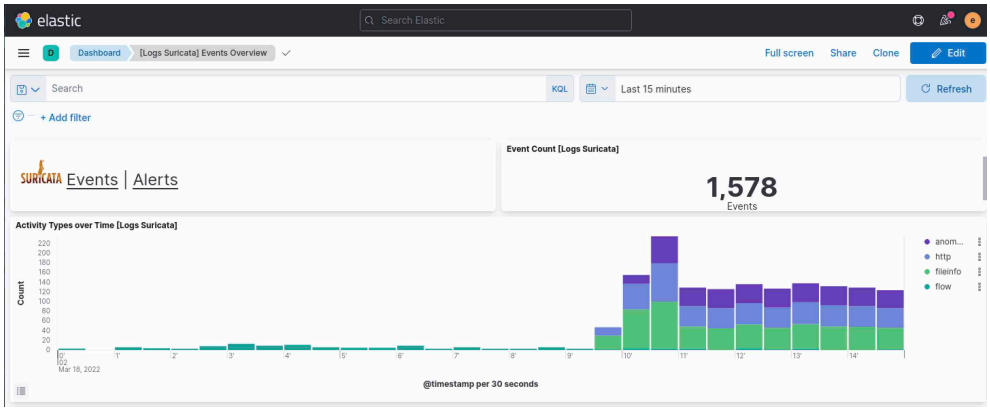
9. Ejecutamos elastic-agent como servicio y sin fleet (options y:n)

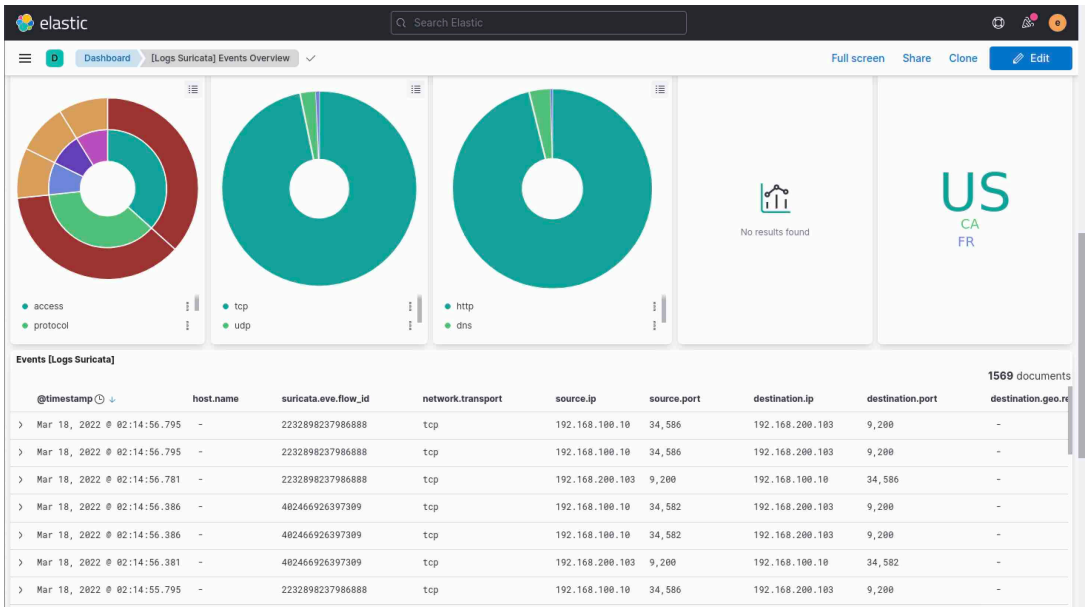


10. Ahora vamos a pfSense y creamos una regla en el firewall (Firewall/Rules/IT) para permitir las conexiones TCP desde la IP del IT (192.168.100.10) a la IP del DMZ (192.168.200.104) , lo cual nos permitirá enviar los logs del Suricata instalado en el Kali IT al ELK alojado en el Kali DMZ, y permitirán la visualización de los mismos con Kibana. Abrimos todos los puertos del destino (Kali DMZ) para la comunicación entre ambas máquinas, ya que el enunciado del ejercicio no lo restringe.



11. Nos vamos a la interfaz del ELK en la Kali DMZ, cerramos las ventanas de integraciones y vamos al dashboard principal pulsando el logo de Elastic arriba a la izquierda en la interfaz. Nos movemos a Analytics/Discover desde el menú lateral izquierdo y podremos ver los eventos de tráfico registrados desde el Suricata de la máquina Kali IT





3.2. Creación e integración de honeypot en Elastic Stack

Un honeypot es un sistema o equipo informático que se configura como si fuese un sistema real con vulnerabilidades que pueden ser explotadas. Se encuentra aislado del resto de la red y monitorizado para poder observar y analizar cuales son los procedimientos que están siguiendo los posibles atacantes para vulnerar el sistema o equipo real.

Se han seguido estos pasos:

1. Vamos a instalar el honeypot “Cowrie” en la máquina Kali DMZ2. Primero hay que instalar el soporte para entornos virtuales de Python y otras dependencias necesarias con el comando

```
sudo apt-get install git python-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv
```

2. Creamos un usuario sin contraseña para cowrie con

```
sudo adduser --disabled-password cowrie
```

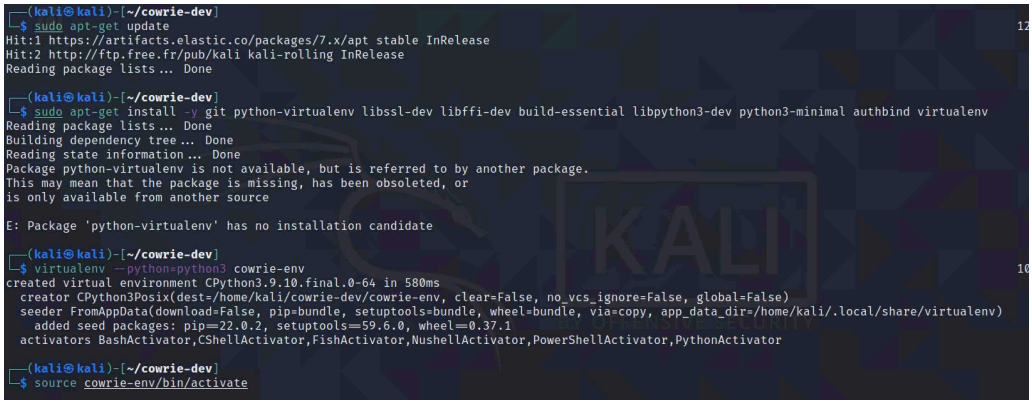
3. Instalamos cowrie con los siguientes comandos

```
sudo su - cowrie
```

```
git clone https://github.com/micheloosterhof/cowrie-dev.git
```

```
cd cowrie
```

4. Activamos el entorno virtual para cowrie con



5. Seguidamente instalamos más dependencias con

```
pip install -U -r requirements.txt
```

5. Aunque Cowrie te permite simular servicio ssh y telnet, en este caso, sólo vamos a utilizar el honeypot para simular el servicio ssh, ubicándolo en el puerto 2222. Para ello cambiamos los siguientes parámetros el archivo `cowrie.cfg`, que previamente hemos copiado de `cowrie.cfg.dist` en `cowrie-dev/etc`. El resto de parámetros los dejamos como están por defecto.

```
  GNU nano 5.4                                          cowrie.cfg
# DO NOT EDIT THIS FILE!
# Changes to default files will be lost on update and are difficult to
# manage and support.
#
# Please make any changes to system defaults by overriding them in
# cowrie.cfg
#
# To override a specific setting, copy the name of the stanza and
# setting to the file where you wish to override it.

# ================================================================
# General Cowrie Options
# ================================================================
[honeypot]

# Sensor name is used to identify this Cowrie instance. Used by the database
# logging modules such as mysql.
#
# If not specified, the logging modules will instead use the IP address of the
# server as the sensor name.
#
# (default: not specified)
sensor_name=database

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = database


# Directory where to save log files in.
#
# (default: log)
log_path = var/log/cowrie
```

```
# ================================================================
# SSH Specific Options
# ================================================================
[ssh]

# Enable SSH support
# (default: true)
enabled = true
```

```
# Endpoint to listen on for incoming SSH connections.
# See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
# (default: listen_endpoints = tcp:2222:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
# listen_endpoints = systemd:domain=INET:index=0
# For both IPv4 and IPv6: listen_endpoints = tcp6:2222:interface=\:\:
# Listening on multiple endpoints is supported with a single space seperator
# e.g listen_endpoints = "tcp:2222:interface=0.0.0.0 tcp:1022:interface=0.0.0.0" will result listening both on ports 2222 and>
# use authbind for port numbers under 1024

listen_endpoints = tcp:2222:interface=0.0.0.0
```

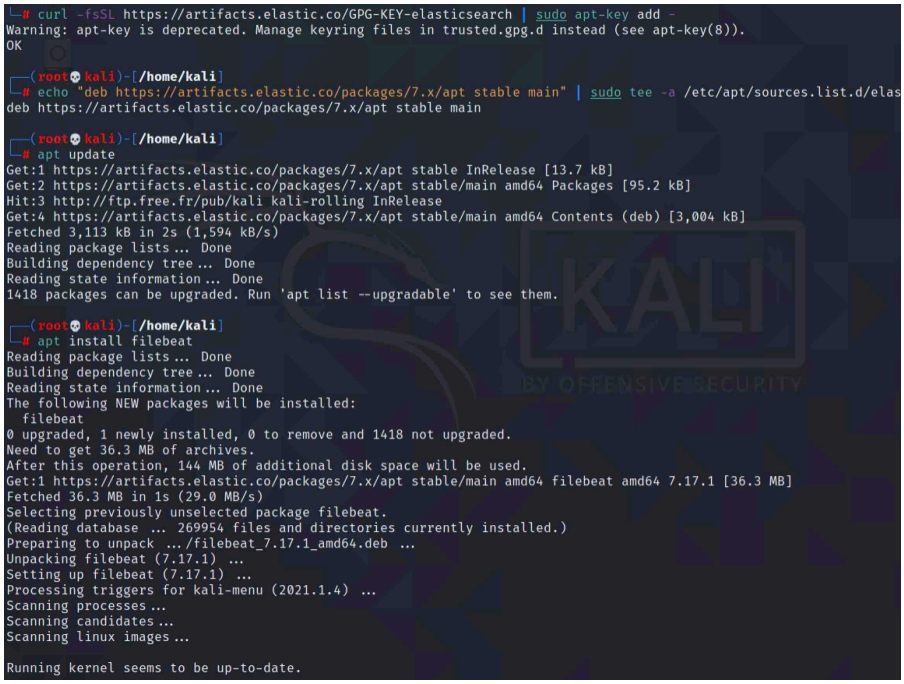
6. Arrancamos cowrie y comprobamos que está activo

```
┌──(cowrie-env)(kali㉿kali)-[~/cowrie-dev]
└─$ ./bin/cowrie start

Join the Cowrie community at: https://www.cowrie.org/slack/

Using activated Python virtual environment "/home/kali/cowrie-dev/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.logger cowrie ]...

┌──(cowrie-env)(kali㉿kali)-[~/cowrie-dev]
└─$ ./bin/cowrie status
cowrie is running (PID: 13851).

┌──(cowrie-env)(kali㉿kali)-[~/cowrie-dev]
└─$
```

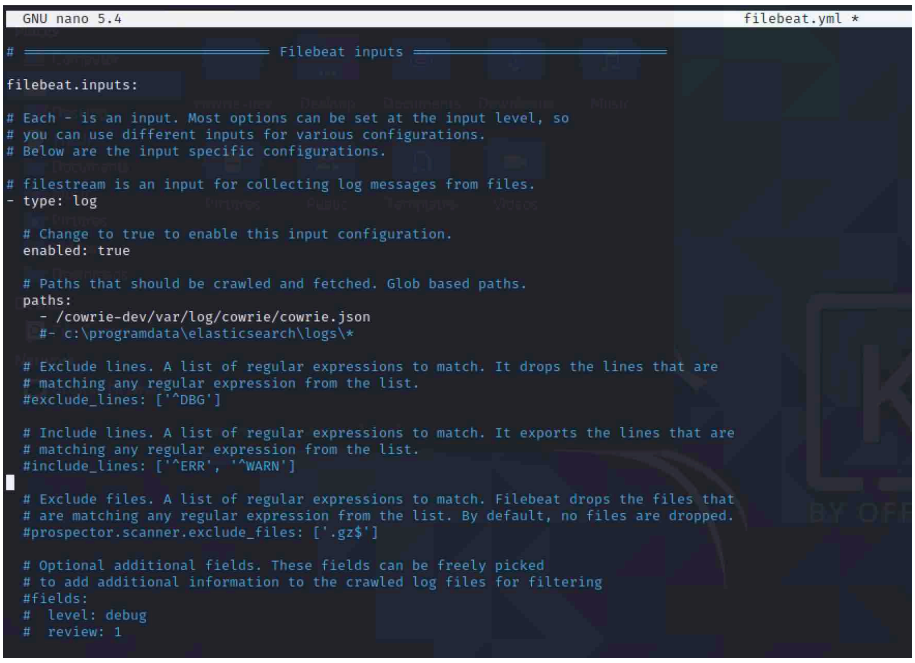
A continuación, vamos a instalar en la misma máquina Kali DMZ2 la herramienta Filebeat de Elastic Stack, que permite enviar logs a ELK ya paseados, por lo que no necesita de enviarlos a Logstash y pueden visualizarse directamente con Kibana.

Los pasos a seguir son:

1. Instalamos desde root con los siguientes comandos



2. Seguidamente configuramos el archivo filebeat.yml ubicado en /etc/filebeat



```
# ------------------------------ Elasticsearch Output ------------------------------
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["192.168.200.103:9200"]

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "changeme"
```

En el apartado Filebeat inputs hemos habilitado y establecido la ruta de la cual Filebeat recogerá los logs de cowrie.

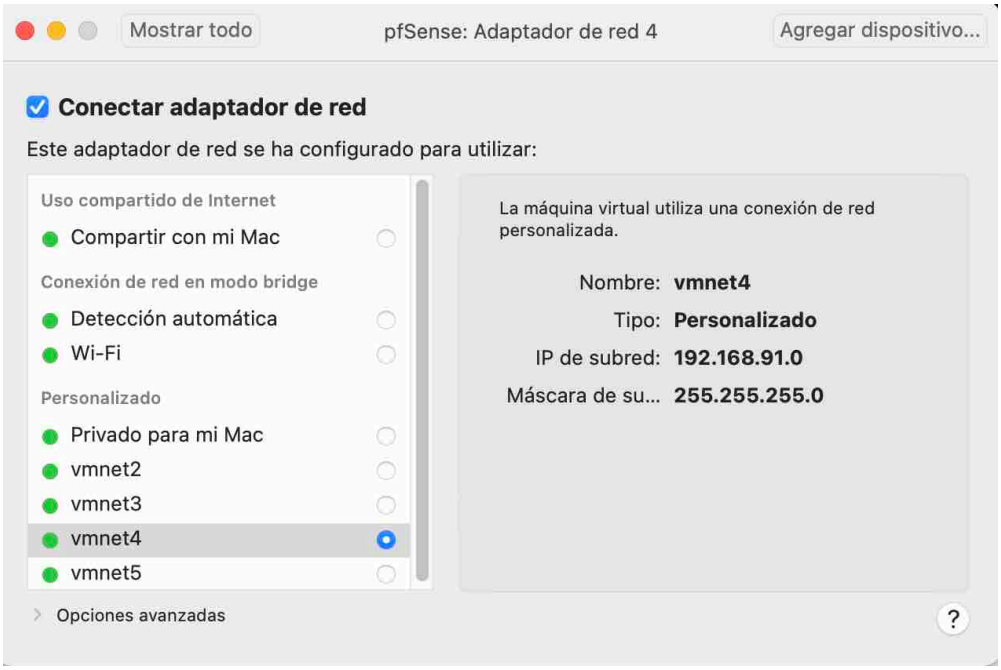
Después, en Elasticsearch Output indicamos la IP y puerto de la maquina DMZ donde tenemos alojado Elastic Stack y el usuario y password de acceso.

3. Arrancamos Filebeat y comprobamos que está activo

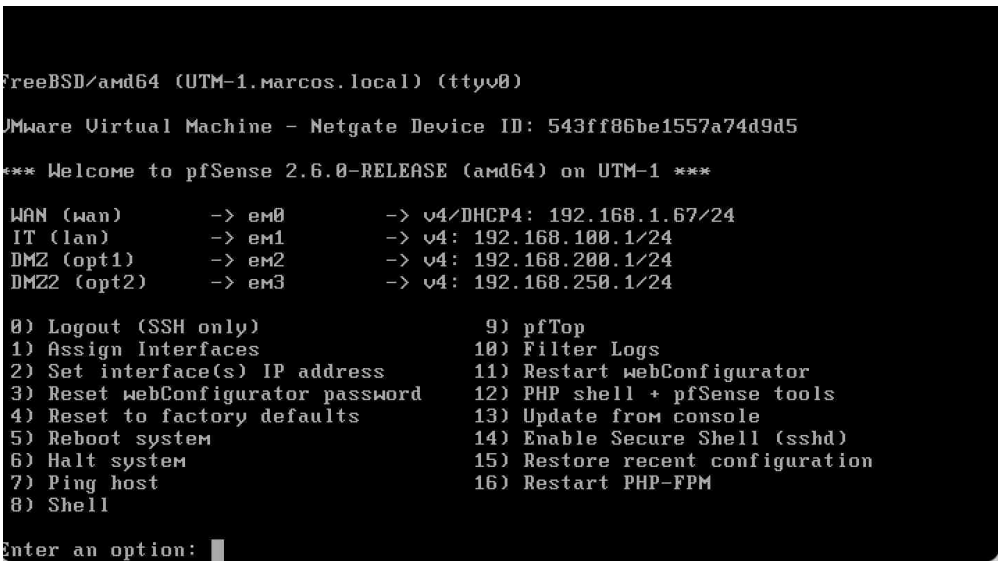
```
┌──(root㉿kali)-[/etc/filebeat]
└─# nano filebeat.yml                                                                                    130 ⨯ 1 ⚙

┌──(root㉿kali)-[/etc/filebeat]
└─# systemctl enable filebeat                                                                              1 ⚙
Synchronizing state of filebeat.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable filebeat

┌──(root㉿kali)-[/etc/filebeat]
└─# systemctl start filebeat                                                                               1 ⚙

┌──(root㉿kali)-[/etc/filebeat]
└─# systemctl status filebeat                                                                              1 ⚙
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
     Loaded: loaded (/lib/systemd/system/filebeat.service; enabled; vendor preset: disabled)
     Active: active (running) since Sat 2022-03-19 12:12:30 EDT; 35min ago
       Docs: https://www.elastic.co/beats/filebeat
   Main PID: 13139 (filebeat)
      Tasks: 9 (limit: 2290)
     Memory: 58.2M
        CPU: 4.150s
     CGroup: /system.slice/filebeat.service
             └─13139 /usr/share/filebeat/bin/filebeat --environment systemd -c /etc/filebeat/filebeat.yml --path.home /usr/share/filebeat --path.co>

Mar 19 12:46:53 kali filebeat[13139]: 2022-03-19T12:46:53.901-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:03 kali filebeat[13139]: 2022-03-19T12:47:03.900-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:03 kali filebeat[13139]: 2022-03-19T12:47:03.902-0400        INFO        [monitoring]        log/log.go:184        Non-zero metrics in>
Mar 19 12:47:13 kali filebeat[13139]: 2022-03-19T12:47:13.900-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:23 kali filebeat[13139]: 2022-03-19T12:47:23.901-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:33 kali filebeat[13139]: 2022-03-19T12:47:33.901-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:33 kali filebeat[13139]: 2022-03-19T12:47:33.905-0400        INFO        [monitoring]        log/log.go:184        Non-zero metrics in>
Mar 19 12:47:39 kali filebeat[13139]: 2022-03-19T12:47:39.538-0400        ERROR        [esclientleg]        transport/logging.go:37        Error di>
Mar 19 12:47:43 kali filebeat[13139]: 2022-03-19T12:47:43.901-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
Mar 19 12:47:53 kali filebeat[13139]: 2022-03-19T12:47:53.901-0400        INFO        [file_watcher]        filestream/fswatch.go:137        Start >
lines 1-21/21 (END)
```

Llegados este punto, vamos a crear una nueva red en pfSense para poder hospedar esta máquina DMZ2.

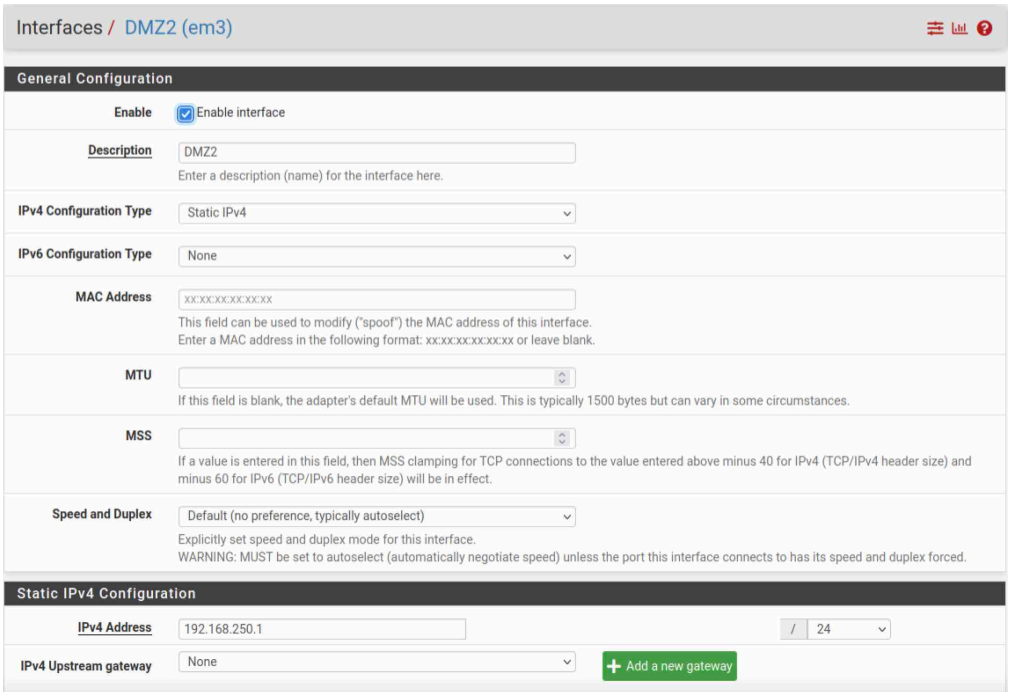
1. Creamos un nuevo adaptador de red (4) en los ajustes de la máquina virtual y le asignamos la red vmnet4



2. Comprobamos que la nueva red está creada (opt2)



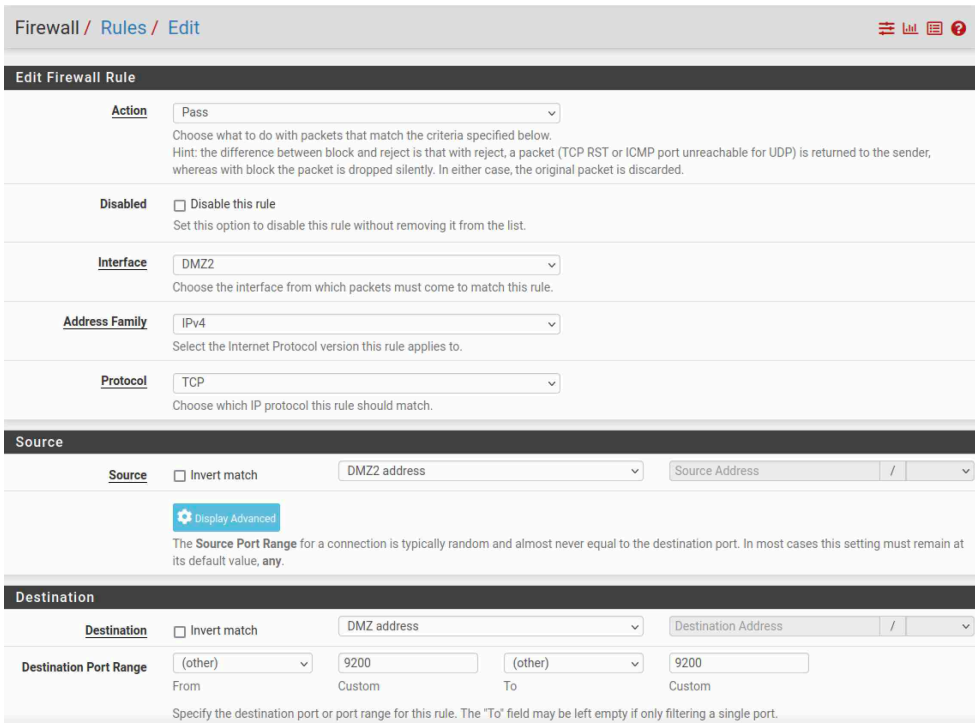
3. Ahora, en la herramienta pfSense, vamos a asignar a la nueva red opt2 creada a la interfaz DMZ2

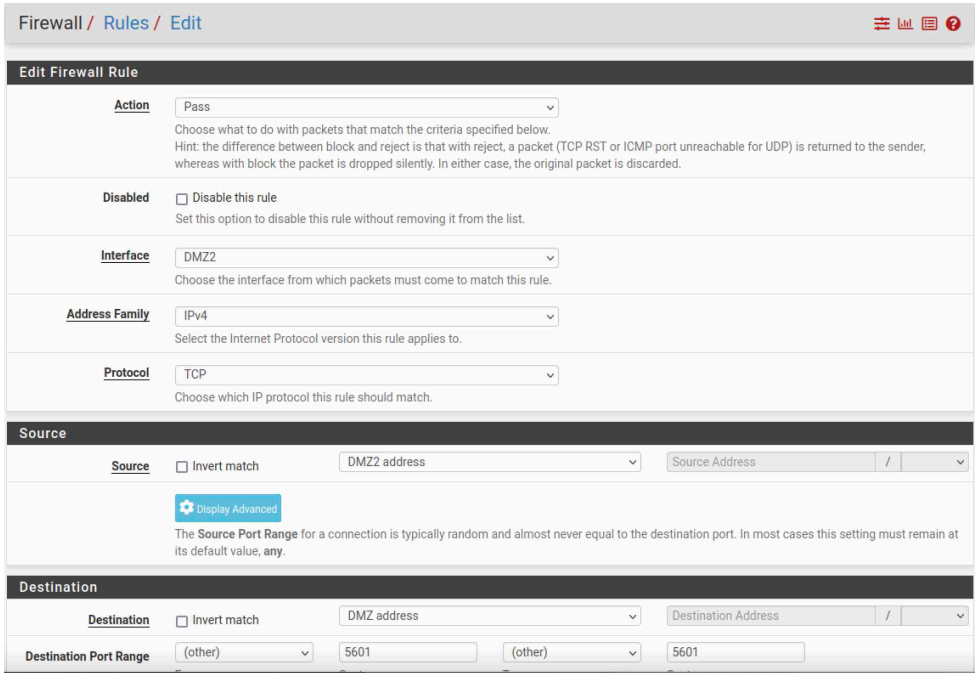


4.Comprobamos que la red DMZ2 está configurada correctamente yendo al dashboard principal

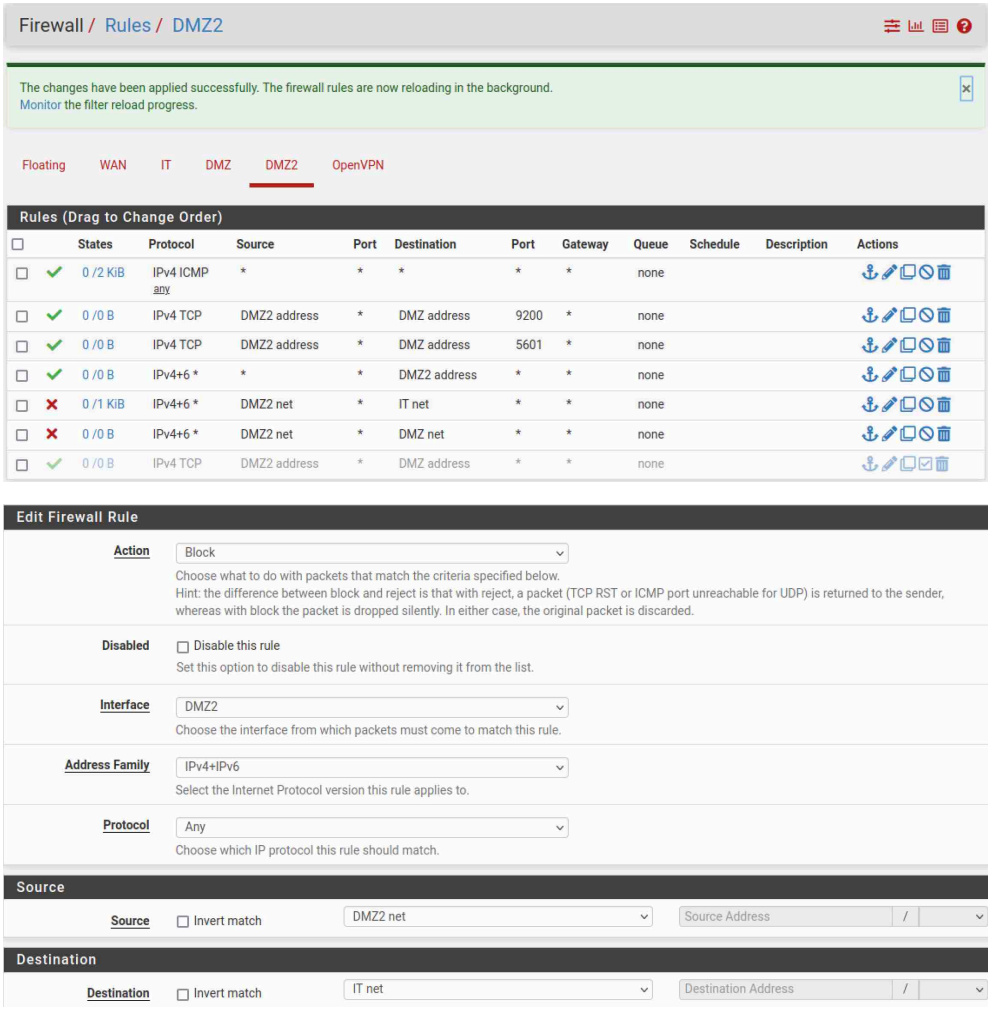
6. Habilitamos el DHCP server y establecemos los DNS servers que utilizaremos con DMZ2

5. Seguidamente vamos a crear las reglas necesarias para que DMZ2 (que aloja cowrie y filebeat) se pueda conectar a DMZ (que aloja ELK, puertos 9200 Kibana y 5601 Elasticsearch) y mostrar los logs del honeypot. Estas son las configuraciones que necesitamos:

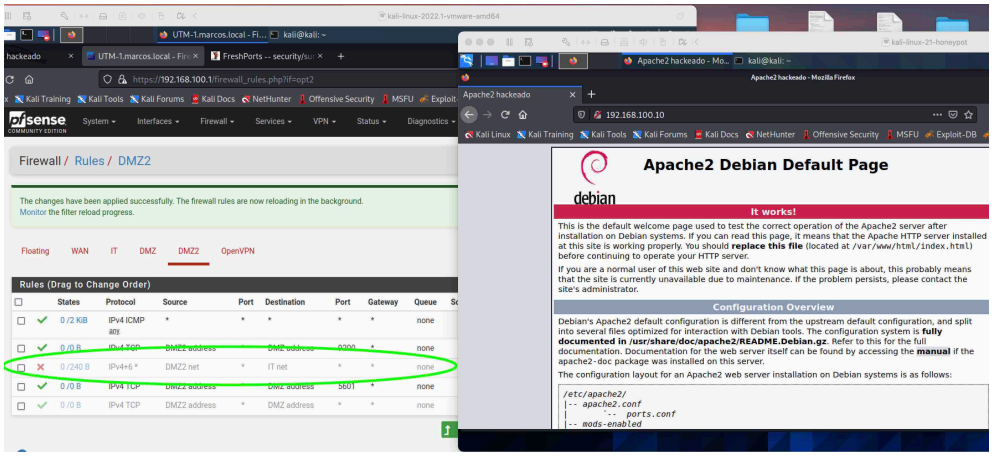




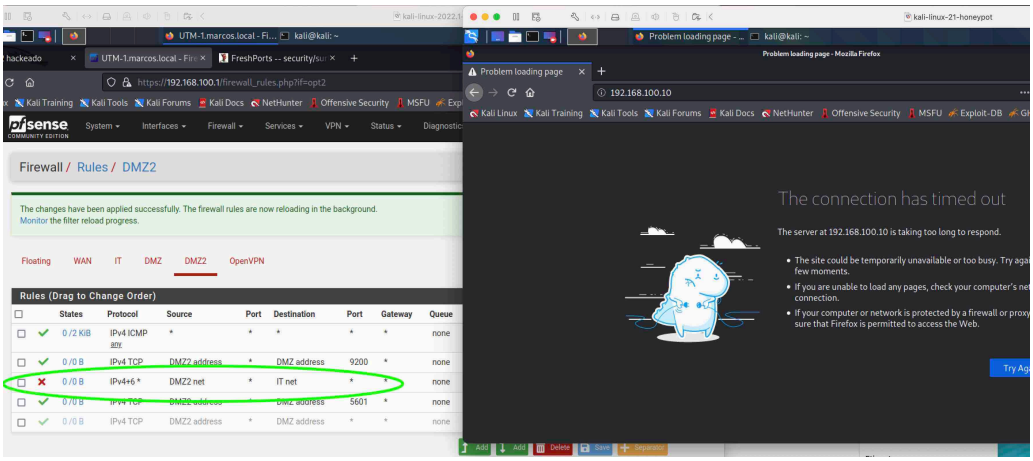
6. Después creamos reglas Block para que no permitan el tráfico proveniente de DMZ2 al resto de equipos de la red (IT y DMZ). Lo único que permitimos es el tráfico desde DMZ2 (honeypot) a DMZ (ELK) por los puertos 9200 y 5601 situando las anteriores reglas Pass por encima de las reglas Block, de esta manera:



Comprobamos que funciona, deshabilitando primero la regla y conectando desde DMZ2 a IT por el puerto 80, logrando la conexión.

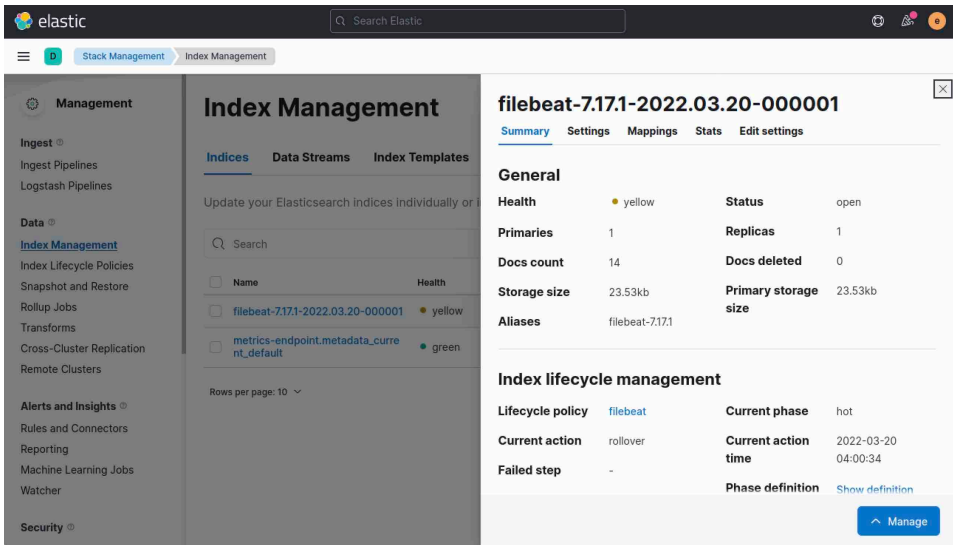


Habilitamos la regla Block y ya la conexión DMZ2-IT por el puerto 80 no se puede hacer



Para terminar, vamos a comprobar que cowrie está recogiendo los logs correctamente y como se visualizan en Kibana gracias a Filebeat.

1. En Elastic Stack, vamos a Management/Index Management y comprobamos que muestra la conexión con Filebeat



2. Comprobamos que se muestran correctamente los logs en Kibana/Discover haciendo con un nmap desde otra maquina al puerto del honeypot