

Seguridad redes.

Detección de intrusos con Honeyport

Vamos a analizar la herramienta honeyport (conocida de inglés como tarro de miel), esta herramienta es miel para los atacantes.

Esta herramienta se usa principalmente en el campo de la seguridad informática y su función consiste en atraer y analizar ataques realizados por bots o hackers.

Los atrae haciéndose pasar por un sistema vulnerable para ver cuantos ataques recibe.

Procedamos a ver cómo funciona.

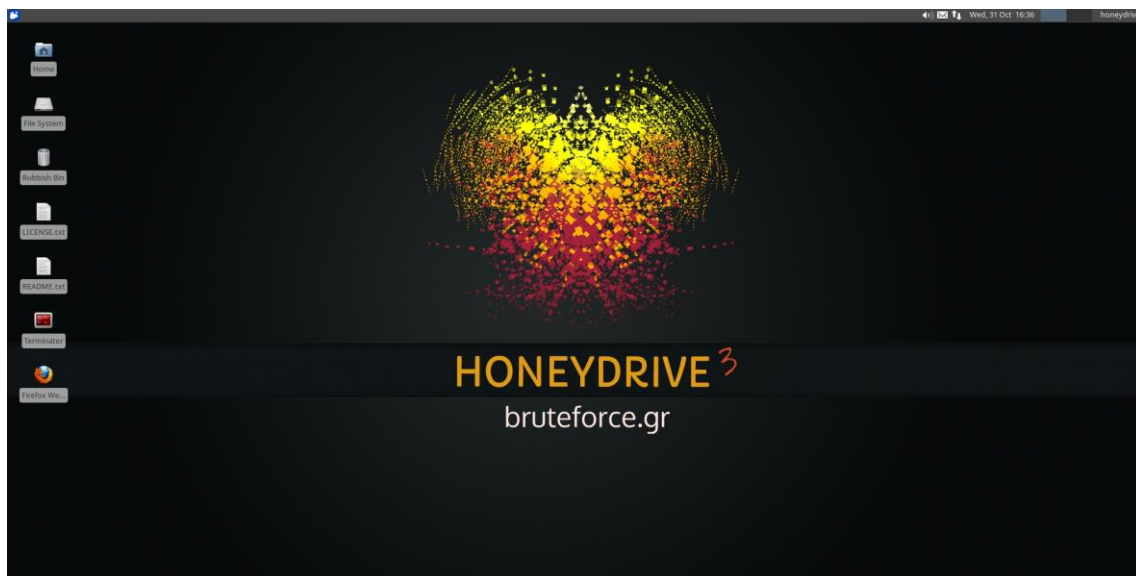
Vamos a descargarnos honeydrive para descargarla nos vamos a este link

<https://bruteforcelab.com/honeydrive> y le damos a descargar.

DOWNLOAD HoneyDrive

The latest version of HoneyDrive Desktop, released on July 2014, is hosted at SourceForge.net:
<http://sourceforge.net/projects/honeydrive/>

Después lo exportaríamos a la máquina virtual y cuando lo encendamos veremos que esta es su interfaz gráfica.



Si nos vamos a README nos Sandra una nota de donde se encuentra cada cosa.

```
File Edit Search Options Help
README.txt

[Specs]
OS: Xubuntu Desktop 12.04.4 LTS i386
HDD: 80GB VMDK (dynamically allocated)
Localization: English (United States)
Keyboard layout: English (United States)
Timezone: UTC (Coordinated Universal Time)

[System]
Connectivity: DHCP
Hostname: honeydrive
User: HoneyDrive
Username/password: honeydrive/honeydrive
Sudo password: honeydrive
Log in automatically: enabled

[Virtualization]
VBox Guest additions: installed
Shared Clipboard: bidirectional
Drag'n'Drop: disabled

[LAMP]
Apache 2 support: PHP, Perl, Python, Ruby/Rails
Document root: /var/www/
AllowOverride All (/var/www/), ServerTokens Minimal, ServerSignature Off
Apache 2 changes: max_execution_time = 300
                  max_input_time = 180
                  memory_limit = 256M
                  post_max_size = 256M
                  upload_max_filesize = 256M
                  max_file_uploads = 40
MySQL root password: honeydrive

[Kippo]
Location: /honeydrive/kippo/
Start script: /honeydrive/kippo/start.sh
Stop script: /honeydrive/kippo/stop.sh
Downloads: /honeydrive/kippo/dl/
TTY logs: /honeydrive/kippo/log/tty/
Credentials: /honeydrive/kippo/data/userdb.txt
MySQL database: kippo
MySQL user/password: root/honeydrive

[Kippo-Graph]
Location: /var/www/kippo-graph/
Configuration: /var/www/kippo-graph/config.php
URL: http://local-or-remote-address/kippo-graph/
MySQL database: kippo
MySQL user/password: root/honeydrive
```

La opción que queremos para escuchar atacantes es Kippo, para acceder a ella nos iríamos a la ruta que nos indica en README, /honeydrive/kippo/, hacemos un ls y nos metemos en el archivo de configuración (kippo.cfg) para ver por qué puertos está escuchando.

```
honeydrive@honeydrive: /honeydrive/kippo
honeydrive@honeydrive: /honeydrive/kippo 80x24
honeydrive@honeydrive:~$ cd /honeydrive/kippo
honeydrive@honeydrive: /honeydrive/kippo$ ls
data  fs.pickle  kippo      kippo.tac  public.key  stop.sh
dl    .gitignore  kippo.cfg  log        README.md  txtcmds
doc   honeyfs    kippo.cfg.dist  private.key  start.sh   utils
honeydrive@honeydrive: /honeydrive/kippo$ sudo nano kippo.cfg
```

Nota: para meternos en el root de honeydrive la contraseña se “honeydrive”.

Y vemos que tiene el puerto 22 por defecto, lo dejaremos como está.

```
honeydrive@honeydrive: /honeydrive/kippo
honeydrive@honeydrive: /honeydrive/kippo 80x24
GNU nano 2.2.6 File: kippo.cfg

#
# Kippo configuration file (kippo.cfg)
#
[honeypot]

# IP addresses to listen for incoming SSH connections.
#
# (default: 0.0.0.0) = any address
#ssh_addr = 0.0.0.0

# Port to listen for incoming SSH connections.
#
# (default: 2222)
ssh_port = 22

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment.
#
[ Read 195 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

Para iniciar kippo usaremos el comando `./start.sh` y así lo pondremos a la escucha de posibles atacantes.

```
honeydrive@honeydrive:/honeydrive/kippo$ ./start.sh
Starting kippo in the background...

Loading dblog engine: mysql
honeydrive@honeydrive:/honeydrive/kippo$
```

Ahora miraos cual es nuestra IP para habilitar el puerto desde el router, con esta IP le diremos que vaya al puerto.

```
honeydrive@honeydrive:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:38:d1:ec
          inet addr:192.168.0.164  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe38:d1ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:105 errors:0 dropped:0 overruns:0 frame:0
          TX packets:59 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10562 (10.5 KB)  TX bytes:10048 (10.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:60 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4733 (4.7 KB)  TX bytes:4733 (4.7 KB)

honeydrive@honeydrive:~$
```

Y la buscamos en el navegador.

| 192.168.0.1/login.html

Y abríamos los puertos

Mobile

Redirección de Puertos

DMZ

Control Parental

DNS & DDNS

UPnP

WoLAN

Añadir asignación de puertos

Servicio

TCP

▼

Dispositivo

honeydrive

▼

LAN IP

192

168

0

164

Tipo

☒ Puerto

☐ Intervalo de puertos

Puerto público

22

Puerto LAN

22

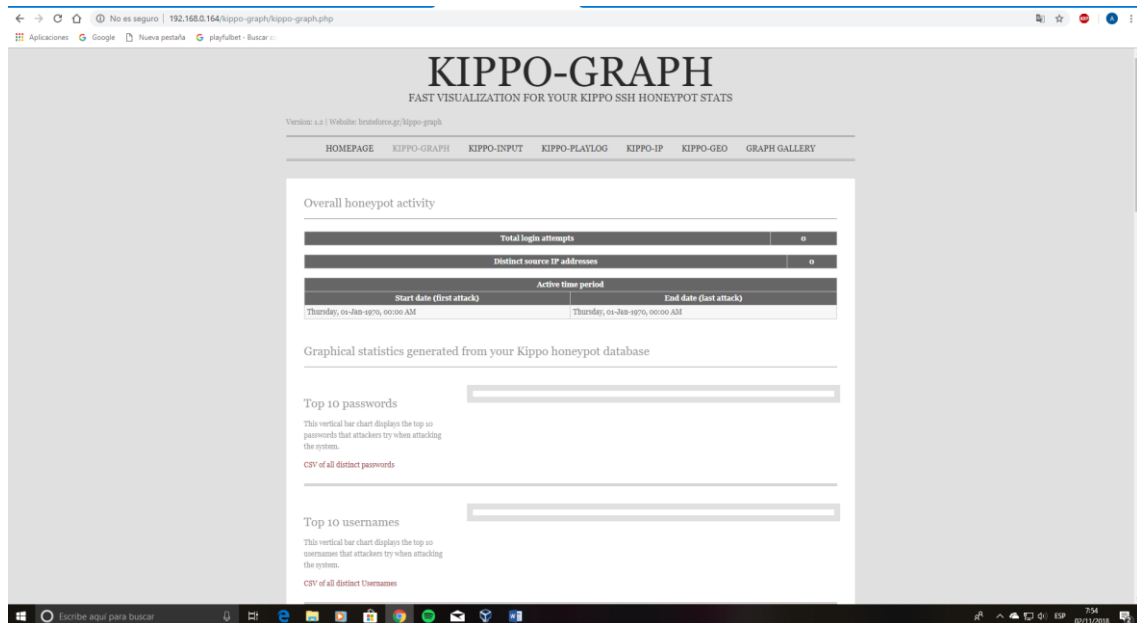
Guardar

Cancelar

Esperamos a que se produzca un ataque.

Podemos ver los ataques que hemos recibido poniendo esta dirección
<http://192.168.0.164/kippo-graph/kippo-graph.php>

Y podremos ver en entorno grafico los ataques que hemos recibido.



Ya todo sería esperar a que nos ataquen y nos iría dando información de los ataques que recibimos.

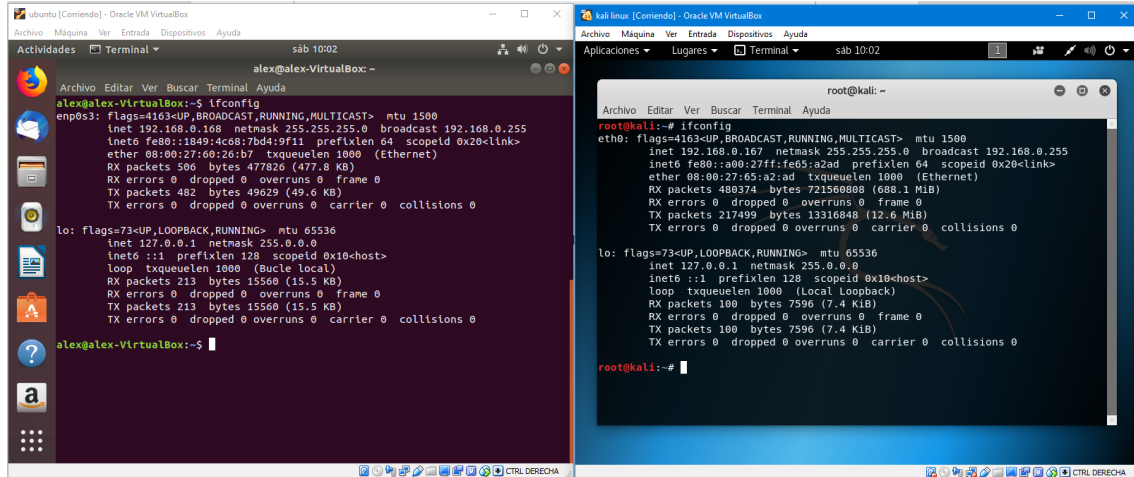
Detección de intrusos con IDS Suricata

Vamos a hacer detección de intrusos con IDS Suricata.

Para ello usaremos del host una tendrá Ubuntu y el otro Kali Linux.

Primero miraremos en que IP están ambos con ifconfig.

Y vemos que Ubuntu tiene la 192.168.0.168 y la Kali la 192.168.0.167



The image shows two side-by-side terminal windows. The left window is titled 'Ubuntu [Comando] - Oracle VM VirtualBox' and shows the output of the 'ifconfig' command for the 'alex@alex-VirtualBox' user. It displays details for the 'enp0s3' and 'lo' interfaces. The right window is titled 'kali linux [Comando] - Oracle VM VirtualBox' and shows the output of the 'ifconfig' command for the 'root@kali:~' user. It displays details for the 'eth0' and 'lo' interfaces.

```
alex@alex-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.168 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::1849:4c68:7bd4:9f11 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:60:26:b7 txqueuelen 1000 (Ethernet)
    RX packets 506 bytes 477826 (477.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 482 bytes 49629 (49.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Bucle local)
    RX packets 213 bytes 15560 (15.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 213 bytes 15560 (15.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

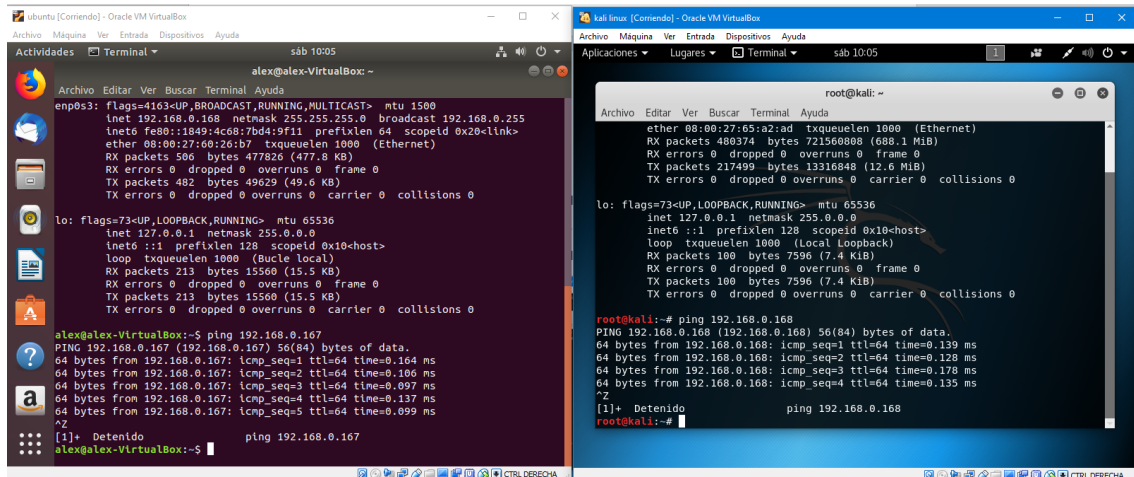
alex@alex-VirtualBox:~$

root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.167 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a00:27ff:fe65:a2ad prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:65:a2:ad txqueuelen 1000 (Ethernet)
    RX packets 480374 bytes 721560808 (688.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 217499 bytes 13316848 (12.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 100 bytes 7596 (7.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 100 bytes 7596 (7.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#
```

Verificamos la conexión entre ambas con un ping.

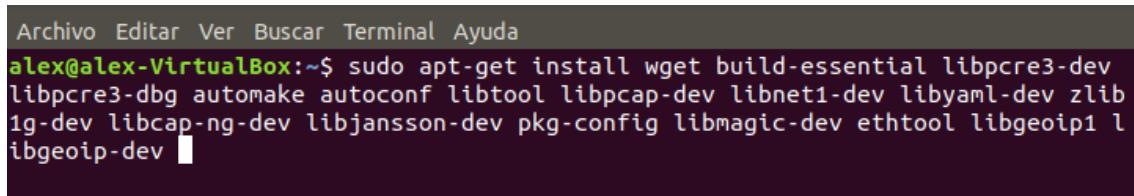


The image shows two side-by-side terminal windows. The left window is titled 'Ubuntu [Comando] - Oracle VM VirtualBox' and shows the output of the 'ping 192.168.0.167' command. The right window is titled 'kali linux [Comando] - Oracle VM VirtualBox' and shows the output of the 'ping 192.168.0.168' command.

```
alex@alex-VirtualBox:~$ ping 192.168.0.167
PING 192.168.0.167 (192.168.0.167) 56(84) bytes of data.
 64 bytes from 192.168.0.167: icmp_seq=1 ttl=64 time=0.164 ms
 64 bytes from 192.168.0.167: icmp_seq=2 ttl=64 time=0.106 ms
 64 bytes from 192.168.0.167: icmp_seq=3 ttl=64 time=0.097 ms
 64 bytes from 192.168.0.167: icmp_seq=4 ttl=64 time=0.137 ms
 64 bytes from 192.168.0.167: icmp_seq=5 ttl=64 time=0.099 ms
^Z
[1]+  Detenido                  ping 192.168.0.167
alex@alex-VirtualBox:~$

root@kali:~# ping 192.168.0.168
PING 192.168.0.168 (192.168.0.168) 56(84) bytes of data.
 64 bytes from 192.168.0.168: icmp_seq=1 ttl=64 time=0.139 ms
 64 bytes from 192.168.0.168: icmp_seq=2 ttl=64 time=0.128 ms
 64 bytes from 192.168.0.168: icmp_seq=3 ttl=64 time=0.178 ms
 64 bytes from 192.168.0.168: icmp_seq=4 ttl=64 time=0.135 ms
^Z
[1]+  Detenido                  ping 192.168.0.168
root@kali:~#
```

Y instalamos todas las dependencias de Suricata en Ubuntu con:



The image shows a terminal window titled 'Ubuntu [Comando] - Oracle VM VirtualBox' with the command 'sudo apt-get install wget build-essential libpcrc3-dev libpcrc3-dbg automake autoconf libtool libcap-dev libnet1-dev libyaml-dev zlib1g-dev libcap-ng-dev libjansson-dev pkg-config libmagic-dev ethtool libgeoip1 libgeoip-dev' being executed.

```
alex@alex-VirtualBox:~$ sudo apt-get install wget build-essential libpcrc3-dev libpcrc3-dbg automake autoconf libtool libcap-dev libnet1-dev libyaml-dev zlib1g-dev libcap-ng-dev libjansson-dev pkg-config libmagic-dev ethtool libgeoip1 libgeoip-dev
```

Descargamos el Suricata

```
alex@alex-VirtualBox:~$ wget http://www.openinfosecfoundation.org/download/suricata-2.0.8.tar.gz
```

Una vez descargado lo descomprimos

```
2018-11-03 10:28:53 (2,32 MB/s) - "suricata-2.0.8.tar.gz"
4209]

alex@alex-VirtualBox:~$ tar -xvf suricata-2.0.8.tar.gz
```

Y entramos con cd suricata-2.0.8

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
alex@alex-VirtualBox:~$ cd suricata-2.0.8
alex@alex-VirtualBox:~/suricata-2.0.8$
```

Ahora lo vamos a configurar con `./configure --sysconfdir=/etc --localstatedir=/var --disable-gccmarch-native --enable-geoip`

Sysconfdir: Establece el directorio para los archivos de configuración.

Localstatedir: determina el directorio para los archivos log (los que contienen los mensajes del sistema) donde se mostrarán las alertas.

Disable-gccmarch-native: Esto es porque lo estamos ejecutando en una máquina virtual.

Enable-geoip: Para habilitar el uso del paquete de geolocalización de direcciones IP.

```
alex@alex-VirtualBox:~/suricata-2.0.8$ ./configure --sysconfdir=/etc --localstatedir=/var --disable-gccmarch-native --enable-geoip
checking for a BSD-compatible install... /usr/bin/install -c
```

Utilizaremos el comando make que será el que se encargue de leer todos los makefiles instalados de suricata (o cualquier otro programa).

```
make[1]: se sale del directorio '/home/alex/suricata-2.0.8'
alex@alex-VirtualBox:~/suricata-2.0.8$ sudo make install
```

```
make[1]: se sale del directorio '/home/alex/suricata-2.0.8'
alex@alex-VirtualBox:~/suricata-2.0.8$ sudo make install-conf
```

Estas reglas servirán para saber si los paquetes son sospechosos.

Ahora hacemos make install-rules

```
alex@alex-VirtualBox:~/suricata-2.0.8$ sudo make install-rules
```

Para verificar que toda la configuración ha sido exitosa hacemos `ls /etc/suricata/rules`.

```
alex@alex-VirtualBox:~/suricata-2.0.8$ ls /etc/suricata/rules/
botcc.portgrouped.rules      emerging-netbios.rules
botcc.rules                  emerging-p2p.rules
BSD-License.txt              emerging-policy.rules
ciarmy.rules                  emerging-pop3.rules
classification.config         emerging-rpc.rules
compromised-ips.txt           emerging-scada.rules
compromised.rules             emerging-scan.rules
decoder-events.rules          emerging-shellcode.rules
dns-events.rules              emerging-smtp.rules
drop.rules                    emerging-snmp.rules
dshield.rules                 emerging-sql.rules
emerging-activex.rules        emerging-telnet.rules
emerging-attack_response.rules emerging-tftp.rules
emerging-chat.rules           emerging-trojan.rules
emerging-current_events.rules emerging-user_agents.rules
emerging-deleted.rules        emerging-voip.rules
emerging-dns.rules            emerging-web_client.rules
emerging-dos.rules            emerging-web_server.rules
emerging-exploit.rules        emerging-web_specific_apps.rules
emerging-ftp.rules            emerging-worm.rules
emerging-games.rules          gpl-2.0.txt
emerging-icmp_info.rules      http-events.rules
emerging-icmp.rules           LICENSE
emerging-imap.rules           sid-msg.map
emerging-inappropriate.rules  smtp-events.rules
emerging-info.rules           stream-events.rules
emerging-malware.rules        suricata-2.0-enhanced-open.txt
emerging-misc.rules           tor.rules
```

Tenemos que configurar el archivo `suricata.yaml`

Y tenemos que poner donde se guardara el log

```
GNU nano 2.9.3 /etc/suricata/suricata.yaml

# The default logging directory. Any log or output file will be
# placed here if its not specified with a full path name. This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/
```

Nos vamos a HTTP y ponemos el puerto que examinara el tráfico.

Miramos que este nuestro rango de IP

```
# These would be retrieved during the Signature address parsing stage.
address-groups:

  HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"

  EXTERNAL_NET: "!$HOME_NET"

  HTTP_SERVERS: "$HOME_NET"
```


Y aquí que tengamos puesto el puerto 80.

```
# Holds the port group vars that would be passed in a Signature.  
# These would be retrieved during the Signature port parsing stage.  
port-groups:  
  
    HTTP_PORTS: "80"  
  
    SHELLCODE_PORTS: "!80"
```

En sesión nominada Threading debe establecer los parámetros relacionados al soporte de suricata. Añadimos detect-thread-ratio = N en el que N será el número de procesadores del host.

```
threading:  
# On some cpu's/architectures it is beneficial to tie individual threads  
# to specific CPU's/CPU cores. In this case all threads are tied to CPU0,  
# and each extra CPU/core has one "detect" thread.  
#  
# On Intel Core2 and Nehalem CPU's enabling this will degrade performance.  
#  
set-cpu-affinity: no  
detect-thread-ratio: 1.5  
# Tune cpu affinity of suricata threads. Each family of threads can be bound
```

Para realizar la detención de intrusos debemos deshabilitar los paquetes ofload en la tarjeta de red que es donde estará escuchando suricata.

```
Archivo Editor Ver Buscar Terminal Ayuda  
alex@alex-VirtualBox:~/suricata-2.0.8$ suricata -c /etc/suricata/suricata.yaml  
-i enp0s3
```

Ahora nos vamos al kali Linux y ponemos nmap -PS (IP del Ubuntu).

```
root@kali:~# nmap -PS 192.168.0.168  
Starting Nmap 7.70 ( https://nmap.org ) at 2018-11-04 22:02 CET  
█
```

Ahora revisamos las alertas con tail -f /var/log/suricata/eve.json (Debemos estar como root)

Ahí podremos que tenemos señal de la Kali.