1. Configuración previa

En mi caso, X = 18.
Las IPs que tiene cada maquina son las siguientes:
--Mi pc: 10.100.18.10
--Kali1: eth0: 10.100.18.1 , eth1: 10.100.19.1
--Kali2: eth0: 10.100.19.2

++
+

1.1. Copia la distribución kali en 2 tarjetas de memoria

$ Primero: nos bajamos la imagen de Kali a nuestro pc (lo dejamos en el home por ejemplo): https://images.offensive-security.com/arm-images/kali-linux-2018.1a-rpi3-nexmon.img.xz

$ Segundo: una vez lo hemos copiado en nuestro pc, lo descomprimimos: xz -d kali-linux-2018.1a-rpi3-nexmon.img.xz (en el caso de no tener instalado xz, lo buscamos con apt-cache search xz, e instalamos el paquete que necesitemos)

$ Tercero: Mete la tarjeta microSD en tu portátil. Para saber el nombre del dispositivo que se corresponde con la tarjeta miniSD en tu ordenador ejecuta el siguiente comando:
df -h

Este comando mostrará todas las particiones que hay montadas en la máquina, hay que localizar la que se corresponde con la memoria miniSD. En particular, en Linux será algo como: /dev/sdX1 montada en /media/nombreUsuario. Si la tarjeta ya tenıá grabado algo previamente, pueden aparecer varios puntos de montaje: /dev/sdX1, /dev/sdX2, etc. Primero es necesario desmontar todas ellas, por ejemplo:
umount /dev/sdX1 (en mi caso no se llamaba sdX1 o sdX2. Era elgo estilo mmcblk0p1 y mmcblk0p2)

Para copiar esta distribución ejecuta en tu portátil el siguiente comando, teniendo en cuenta que <DIR> es el nombre de la carpeta donde está almacenada la imagen de kali y <DEV> es el nombre del dispositivo sdX en tu ordenador sin el número. Es importante que seas especialmente cuidadoso con estas instrucciones para no borrar alguna de las particiones de tu ordenador:
sudo dd bs=4M if=./kali-linux-2018.1a-rpi3-nexmon.img of=/dev/mmcblk0p

Repetir el proceso para la segunda tarjeta microSD. No es necesario volver a descargar la imagen de kali porque ya la tenemos descargada.

++

1.2. Configuración inicial kali1

conectamos la raspberry al pc a traves del cable USB TTL serial

lanzamos en un terminal: sudo screen /dev/ttyUSB0 115200

(en el caso de no tener instalado screen lo instalamos con apt-get install)

Este programa abrirá una consola serie con la raspberry pi, entra con nombre de usuario root, contraseña toor.

ponemos la siguiente configuracion en kali1 (usar un editor: nano, vi, etc):

```
root@kali1:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
 address 10.100.18.1
 netmask 255.255.255.0

auto eth1
iface eth1 inet static
 address 10.100.19.1
 netmask 255.255.255.0

allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

añadimos la configuracion para las redes inalambricas. No pongo las contraseñas por razones obvias:

```
root@kali1:~# cat /etc/wpa_supplicant/wpa_supplicant.conf
network={
 ssid="eduroam"
 scan_ssid=1
 key_mgmt=WPA-EAP
 pairwise=CCMP
 group=CCMP TKIP
 eap=PEAP
 ca_cert="/etc/ssl/certs/ca.pem"
 identity="u.usuario@alumnos.urjc.es"
 domain_suffix_match="urjc.es"
 phase1="peaplabel=0"
 phase2="auth=MSCHAPV2"
 password="contraseña"
}

 network={
 ssid="reddetucasa"
 psk="contraseña"
key_mgmt=WPA-PSK
}
```

descargamos el fichero.pem de la autoridad de certificacion de la urjc (el fichero aparece en el campus):

```
root@kali1:~# cat /etc/ssl/certs/ca.pem
# subject: CN=TERENA SSL High Assurance CA 3, O=TERENA, L=Amsterdam,
ST=Noord-Holland, C=NL
# issuer:  CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com,
O=DigiCert Inc, C=US
# SHA256 Fingerprint:
BE:6A:0D:9E:1D:11:5F:22:93:F6:AB:F1:1B:3E:C8:E8:82:E2:44:26:EE:EB:09:AA:A
5:03:59:79:93:E7:7A:25
-----BEGIN CERTIFICATE-----
MIIE4DCCA8igAwIBAgIQC1w0NWdbJGfA1zI3+Q1flDANBgkqhkiG9w0BAQsFADBs
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSswKQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5j
ZSBFViBSb290IENBMB4XDTE0MTExODEyMDAwMFoXDTI0MTExODEyMDAwMFowczEL
MAkGA1UEBhMCTkwxFjAUBgNVBAgTDU5vb3JkLUhvbGxhbmQxEjAQBgNVBAcTCUFt
c3RlcmRhbTEPMA0GA1UEChMGVEVSRU5BMScwJQYDVQQDEx5URVJFTkEgU1NMIEhp
Z2ggQXNzdXJhbmNlIENBIDMwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQChNsmK4gfxr6c9j2OMBRo3gOA7z5keoaPHiX4rUX+1fF1Brmvf7Uo83sRiXRYQ
RJrD79hzJrulDtdihxgS5HgvIQHqGrp3NRRDUlq/4bItLTp9QCHzLhRQSrSYaFkI
zztYezwb3ABzNiVciqQFk7WR9ebh9ZaCxaXfebcg7LodgQQ4XDvkW2Aknkb1J8NV
nlbKen6PLlNSL4+MLV+uF1e87aTgOxbM9sxZ1/1LRqrOu28z9WA8qUZn2Av+hcP2
TQIBoMPMQ8dT+6Yx/0Y+2J702OU//dS0pi8gMe7FtYVcZrlcSy/C40I7EFYHEjTm
zH4EGvG6t9wZua2atFKvP/7HAgMBAAGjggF1MIIBcTASBgNVHRMBAf8ECDAGAQH/
AgEAMA4GA1UdDwEB/wQEAwIBhjB/BggrBgEFBQcBAQRzMHEwJAYIKwYBBQUHMAGG
GGh0dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBJBggrBgEFBQcwAoY9aHR0cDovL2Nh
Y2VydHMuZGlnaWNlcnQuY29tL0RpZ2lDZXJ0SGlnaEFzc3VyYW5jZUVWUm9vdENB
LmNydDBLBgNVHR8ERDBCMECgPqA8hjpodHRwOi8vY3JsNC5kaWdpY2VydC5jb20v
RGlnaUNlcnRIaWdoQXNzdXJhbmNlRVZSb290Q0EuY3JsMD0GA1UdIAQ2MDQwMgYE
VR0gADAqMCgGCCsGAQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5jb20vQ1BT
MB0GA1UdDgQWBBTCuIXX4bkTvdFIvP1e3H2QQnqKqTAfBgNVHSMEGDAWgBSxPsNp
A/i/RwHUmCYaCALvY2QrwzANBgkqhkiG9w0BAQsFAAOCAQEAsCq7NTey6NjZHqT4
kjZBNU3sItnD+RYAMWx4ZyaELcy7XhndQzX88TYSCYxl/YWB6lCjxx0dL3wTZUbX
r+WRDzz5xX+98kdYrwNCT7fmT4eenv6cCS1sC9hc4sIl5dkb1pguY3ViV5D8/yEB
hadOpw3TwI8xkqe2j/H5fp4Oaf9cFdpf9C85mQgZJwsvtvmmDTQTPcGPRFTgdGtY
2xbWxDah6HjKpX6iI4BTBQhhpX6TJl6/GEaYK07s2Kr8BFPhrgmep9vrepWv61x7
dnnqz5SeAs6cbSm551qG7Dj8+6f/8e33oqLC5Ldnbt0Ou6PjtZ4O02dN9cnicemR
1B0/YQ==
-----END CERTIFICATE-----
# subject: CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com,
O=DigiCert Inc, C=US
# issuer:  CN=DigiCert High Assurance EV Root CA, OU=www.digicert.com,
O=DigiCert Inc, C=US
# SHA256 Fingerprint:
74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:A
B:D7:80:6E:D3:B1:18:CF
-----BEGIN CERTIFICATE-----
MIIDxTCCAq2gAwIBAgIQAqxcJmoLQJuPC3nyrkYldzANBgkqhkiG9w0BAQUFADBs
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSswKQYDVQQDEyJEaWdpQ2VydCBIaWdoIEFzc3VyYW5j
ZSBFViBSb290IENBMB4XDTA2MTExMDAwMDAwMFoXDTMxMTExMDAwMDAwMFowbDEL
MAkGA1UEBhMCVVMxFTATBgNVBAoTDERpZ2lDZXJ0IEluYzEZMBcGA1UECxMQd3d3
LmRpZ2ljZXJ0LmNvbTErMCkGA1UEAxMiRGlnaUNlcnQgSGlnaCBBc3N1cmFuY2Ug
RVYgUm9vdCBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMbM5XPm
+9S75S0tMqbf5YE/yc0lSbZxKsPVlDRnogocsF9ppkCxxLeyj9CYpKlBWTrT3JTW
```

```
PNt0OKRKzE0lgvdKpVMSOO7zSW1xkX5jtqumX8OkhPhPYlG++MXs2ziS4wblCJEM
xChBVfvLWokVfnHoNb9Ncgk9vjo4UFt3MRuNs8ckRZqnrG0AFFoEt7oT61EKmEFB
Ik5lYYeBQVCmeVyJ3hlKV9Uu5l0cUyx+mM0aBhakaHPQNAQTXKFx01p8VdteZOE3
hzBWBOURtCmAEvF5OYiiAhF8J2a3iLd48soKqDirCmTCv2ZdlYTBoSUeh10aUAsg
EsxBu24LUTi4S8sCAwEAAaNjMGEwDgYDVR0PAQH/BAQDAgGGMA8GA1UdEwEB/wQF
MAMBAf8wHQYDVR0OBBYEFLE+w2kD+L9HAdSYJhoIAu9jZCvDMB8GA1UdIwQYMBaA
FLE+w2kD+L9HAdSYJhoIAu9jZCvDMA0GCSqGSIb3DQEBBQUAA4IBAQAcGgaX3Nec
nzyIZgYIVyHbIUf4KmeqvxgydkAQV8GK83rZEWWONfqe/EW1ntlMMUu4kehDLI6z
eM7b41N5cdblIZQB2lWHmiRk9opmzN6cN82oNLFpmyPInngiK3BD41VHMWEZ71jF
hS9OMPagMRYjyOfiZRYzy78aG6A9+MpeizGLYAiJLQwGXFK3xPkKmNEVX58Svnw2
Yzi9RKR/5CYrCsSXaQ3pjOLAEFe4yHYSkVXySGnYvCoCWw9E1CAx2/S6cCZdkGCe
vEsXCS+0yx5DaMkHJ8HSXPfqIbloEpw8nL+e/IBcm2PN7EeqJSdnoDfzAIJ9VNep
+OkuE6N36B9K
-----END CERTIFICATE-----
```

editamos el fichero /etc/hostname para renombrar a la raspberry como kali1:

```
root@kali1:~# cat /etc/hostname
kali1
```

como queremos que kali1 sea un router, tenemos que editar el siguiente fichero:

/etc/sysctl.conf

añadimos al final la siguiente linea: net.ipv4.ip_forward = 1

Hay un problema con la hora, la distribución kali tiene configurada la hora en la que se creó la distribución, diciembre de 2017 y el certificado para conectarse a la red inalámbrica aún no es válido en esa fecha, por tanto hay que cambiar la hora a kali1. Para ello vamos a configurar que cada vez que se reinicie la máquina se configure una hora más actual:

```
crontab -e
@reboot date --set "04/17/2018 11:00"
```

Ejecuta reboot para que la configuración tenga efecto y vuelve a entrar a través del puerto serie. La raspberry se habrá conectado a la red inalámbrica eduroam, en su interfaz wlan0. Apunta la dirección IP que te han asignado por DHCP. Esta dirección IP te permitirá conectarte a la raspberry pi, de forma más cómoda y con tantos terminales como necesites, a través de ssh desde tu portátil que también deberá estará conectado a la red eduroam. Ten en cuenta que cada vez que inicies la raspberry le podrán asignar una dirección IP diferente a su interfaz inalámbrica.
Ahora que ya tienes conectada kali1 a Internet, ejecuta:

```
apt-get update
apt-get install tcpdump
apt-get install snort
apt-get install netcat
apt-get install nmap
```

para cambiar la contraseña a la raspberry:
passwrd root
nos pediran la nueva contraseña, y que la volvamos a meter y ya estaria lista la nueva contraseña.

ATENCION: SI NO APARECE ETH1 EN KALI1 AL CONECTAR POR ETHERNET A KALI2, AÑADIR ESTA LINEA EN KALI1: ifconfig eth1 10.100.19.1 netmask 255.255
++

1.3. Configuración inicial kali2

```
root@kali2:~# cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
 address 10.100.19.2
 netmask 255.255.255.0

allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

up route add -net 10.100.18.0/24 gw 10.100.19.1
```

Realiza la misma configuración en /etc/wpa_supplicant/wpa_supplicant.conf que en kali1.
Cambia el nombre a esta segunda raspberry pi para identificarla como kali2. Cambia el passwd de root
y la hora.
Ejecuta reboot para que la configuración tenga efecto y vuelve a entrar a través del puerto serie. La raspberry se habrá conectado a la red inalámbrica eduroam, en su interfaz wlan0. Apunta la dirección IP que te han asignado por DHCP. Esta dirección IP te permitirá conectarte a la raspberry pi, de forma más cómoda y con tantos terminales como necesites, a través de ssh desde tu portátil que también deberá estará conectado a la red eduroam. Ten en cuenta que cada vez que inicies la raspberry le podrán asignar una dirección IP diferente a su interfaz inalámbrica.
El comando para conectarse por ssh:
ssh -l root <dir-ip-wlan-raspberry>
nos pedira la contraseña de root y listo.

==========

1.4. Configuración en tu ordenador
Configura la dirección IP 10.100.X.10/24 en tu ordenador a través de la interfaz cableada que tendrás conectada a kali1. También debes configurar una ruta a la subred 10.100.(X+1).0/24 a través del router 10.100.X.1 (kali1), esta ruta te permitirá alcanzar la máquina kali2. Comprueba que puedes hacer un ping a la máquina kali2.

Si todo los pasos anteriores se han hecho bien, hacer la siguiente modificacion no deberia suponer ningun problema.

Nos vamos a system settings-->network-->wired-->options-->ipv4Settings

ahora añadimos una nueva direccion: 10.100.X.10. Esta es la direccion ip de nuestro pc.
en routes añadimos lo siguiente:
en address: 10.100.X+1.0
netmask: 255.255.255.0
gateway: 10.100.X.1

++

2. Snort
La máquina kali1 va a ejecutar una herramienta IDS, snort, que detecta accesos potencialmente maliciosos y los registra en un fichero de log. Cuando snort descubra tráfico potencialmente malicioso escribirá una alerta en un fichero de logs y almacenará el tráfico malicioso en un fichero de captura. Estos ficheros se encontrarán en la carpeta /var/log/snort.

2.1. Reglas Snort
En la carpeta /etc/snort/rules/ se describen reglas predefinidas en snort para la inspección de tráfico. Dependiendo de la configuración del fichero /etc/snort/snort.conf se podrán cargar las reglas que se desean aplicar al tráfico que el IDS examine. A continuación responde a las siguientes preguntas en la memoria de la práctica:

1. El fichero /etc/snort/snort.conf es el que contiene la configuración de snort. Este fichero está dividido en varias partes, las lıńeas que comienzan por # son comentarios. Nosotros nos vamos a fijar en la parte de la configuración de variables (parte 1 ó Step #1) y en la parte de configuración de reglas (parte 7 ó Step #7). Escribe en la memoria el contenido de las variable HOME_NET y EXTERNAL_NET y explica qué crees que significa ese valor.

Lo primero, comentamos estas lineas de la seccion 7 del fichero /etc/snort/snort.conf:

```
#include $RULE_PATH/voip.rules
#include $RULE_PATH/web-activex.rules
#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
#include $RULE_PATH/community-web-php.rules
```

haciendo un cat -n /etc/snort/snort.conf | grep HOME_NET, podemos ver en que lineas tenemos la palabra HOME_NET. Segun esto, podemos ver que se trata de todas aquellas direcciones de nuestra subred. En cambio las EXTERNAL_NET, son todas aquellas direcciones externas.

---

2. En la sección Step #7 se incluyen la reglas escritas en diversos ficheros que se encuentran en la carpeta /etc/snort/rules, en particular comprueba que se inlcuye el fichero /etc/snort/rules/icmp.rules. Abre este fichero. Las lıńeas que comienzan por # son comentarios, las reglas están escritas cada una en
una única lıńea. Incluye la última regla de ese fichero en la memoria y explica el contenido.

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Large ICMP
Packet"; dsize:>800; reference:arachnids,246; classtype:bad-unknown;
sid:499; rev:4;)
```

esta incluyendo esta linea para avisar de todo el trafico del exterior hacia las redes internas que pueda ser perjudicial(classtype:bad-unknown).

---

3. Busca en el fichero icmp.rules la regla que es una alerta que escribe el mensaje "ICMP PING NMAP" 2, inclúyela en la memoria y explica todo lo que puedes saber de su contenido.

```
root@kali1:~# cat /etc/snort/rules/icmp.rules | grep 'ICMP PING NMAP'
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING NMAP";
dsize:0; itype:8; reference:arachnids,162; classtype:attempted-recon;
sid:469; rev:3;)
```

va a avisar de todo el trafico externo de algun nmap. Nivel de prioridad medio: Attempted Information Leak

---

4. Busca en el fichero icmp-info.rules la regla que es una alerta que escribe el mensaje ICMP PING *NIX", inclúyela en la memoria y explica todo lo que puedes saber de su contenido.

```
root@kali1:~# cat /etc/snort/rules/icmp-info.rules | grep 'ICMP PING'
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP PING *NIX";
itype:8; content:"|10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F|";
depth:32; classtype:misc-activity; sid:366; rev:7;)
```

vemos que se va a avisar de toda misc-activity, proveniente de redes externas. Nivel de prioridad: bajo: Misc activity

---

5. Explica cuál es la diferencia entre ambas reglas y el nivel de prioridad de cada una de ellas. ¿Por qué una tiene mayor prioridad que otra?

Un ping hecho desde nmap esta hecho para ver la topologia de la red. En cambio el otro puede provenir de un usuario que simplemente quiere comprobar si funciona o no internet. Es mas peligroso el primero debido a que es mas probable que intente algo ilegitimo.

---

6. Lanza snort en la máquina kali1 (snort -A console -c /etc/snort/snort.conf -i eth1) para que comience a detectar tráfico potencialmente peligroso y déjalo lanzado para que te vaya mostrando las alertas que detecte en los apartados sucesivos. Una vez arrancado informará de que la inicialización se ha completado e indicará el número de proceso (pid), apúntalo para que, en caso de que sea necesario, puedas matar el proceso.

```
root@kali1:~# snort -A console -c /etc/snort/snort.conf -i eth1
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 4848 5250 6988 7000:7001
7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118
8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091
9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
```

```
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901
1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988
7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088
8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080
9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
    Split Any/Any group = enabled
    Search-Method-Optimizations = enabled
    Maximum pattern length = 20
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort_dynamicengine/libsf_engine.so...
done
Loading all dynamic detection libs from /usr/lib/snort_dynamicrules...
WARNING: No dynamic libraries found in directory
/usr/lib/snort_dynamicrules.
  Finished Loading all dynamic detection libs from
/usr/lib/snort_dynamicrules
Loading all dynamic preprocessor libs from
/usr/lib/snort_dynamicpreprocessor/...
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_dns_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_ftptelnet_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_dce2_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_ssl_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_imap_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_pop_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_ssh_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_dnp3_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_sdf_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_reputation_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_modbus_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_gtp_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_smtp_preproc.so... done
  Loading dynamic preprocessor library
/usr/lib/snort_dynamicpreprocessor//libsf_sip_preproc.so... done
  Finished Loading all dynamic preprocessor libs from
/usr/lib/snort_dynamicpreprocessor/
Log directory = /var/log/snort
WARNING: ip4 normalizations disabled because not inline.
WARNING: tcp normalizations disabled because not inline.
```

```
WARNING: icmp4 normalizations disabled because not inline.
WARNING: ip6 normalizations disabled because not inline.
WARNING: icmp6 normalizations disabled because not inline.
Frag3 global config:
    Max frags: 65536
    Fragment memory cap: 4194304 bytes
Frag3 engine config:
    Bound Address: default
    Target-based policy: WINDOWS
    Fragment timeout: 180 seconds
    Fragment min_ttl:   1
    Fragment Anomalies: Alert
    Overlap Limit:     10
    Min fragment Length:     100
      Max Expected Streams: 768
Stream global config:
    Track TCP sessions: ACTIVE
    Max TCP sessions: 262144
    TCP cache pruning timeout: 30 seconds
    TCP cache nominal timeout: 3600 seconds
    Memcap (for reassembly packet storage): 8388608
    Track UDP sessions: ACTIVE
    Max UDP sessions: 131072
    UDP cache pruning timeout: 30 seconds
    UDP cache nominal timeout: 180 seconds
    Track ICMP sessions: INACTIVE
    Track IP sessions: INACTIVE
    Log info if session memory consumption exceeds 1048576
    Send up to 2 active responses
    Wait at least 5 seconds between responses
    Protocol Aware Flushing: ACTIVE
        Maximum Flush Point: 16000
Stream TCP Policy config:
    Bound Address: default
    Reassembly Policy: WINDOWS
    Timeout: 180 seconds
    Limit on TCP Overlaps: 10
    Maximum number of bytes to queue per session: 1048576
    Maximum number of segs to queue per session: 2621
    Options:
        Require 3-Way Handshake: YES
        3-Way Handshake Timeout: 180
        Detect Anomalies: YES
    Reassembly Ports:
      21 client (Footprint)
      22 client (Footprint)
      23 client (Footprint)
      25 client (Footprint)
      42 client (Footprint)
      53 client (Footprint)
      79 client (Footprint)
      80 client (Footprint) server (Footprint)
      81 client (Footprint) server (Footprint)
      109 client (Footprint)
```

```
      110 client (Footprint)
      111 client (Footprint)
      113 client (Footprint)
      119 client (Footprint)
      135 client (Footprint)
      136 client (Footprint)
      137 client (Footprint)
      139 client (Footprint)
      143 client (Footprint)
      161 client (Footprint)
      additional ports configured but not printed.
Stream UDP Policy config:
    Timeout: 180 seconds
HttpInspect Config:
    GLOBAL CONFIG
      Detect Proxy Usage:       NO
      IIS Unicode Map Filename: /etc/snort/unicode.map
      IIS Unicode Map Codepage: 1252
      Memcap used for logging URI and Hostname: 150994944
      Max Gzip Memory: 104857600
      Max Gzip Sessions: 319687
      Gzip Compress Depth: 65535
      Gzip Decompress Depth: 65535
    DEFAULT SERVER CONFIG:
      Server profile: All
      Ports (PAF): 80 81 311 383 591 593 901 1220 1414 1741 1830 2301
2381 2809 3037 3128 3702 4343 4848 5250 6988 7000 7001 7144 7145 7510
7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180 8181
8243 8280 8300 8800 8888 8899 9000 9060 9080 9090 9091 9443 9999 11371
34443 34444 41080 50002 55555
      Server Flow Depth: 0
      Client Flow Depth: 0
      Max Chunk Length: 500000
      Small Chunk Length Evasion: chunk size <= 10, threshold >= 5 times
      Max Header Field Length: 750
      Max Number Header Fields: 100
      Max Number of WhiteSpaces allowed with header folding: 200
      Inspect Pipeline Requests: YES
      URI Discovery Strict Mode: NO
      Allow Proxy Usage: NO
      Disable Alerting: NO
      Oversize Dir Length: 500
      Only inspect URI: NO
      Normalize HTTP Headers: NO
      Inspect HTTP Cookies: YES
      Inspect HTTP Responses: YES
      Extract Gzip from responses: YES
      Decompress response files:
      Unlimited decompression of gzip data from responses: YES
      Normalize Javascripts in HTTP Responses: YES
      Max Number of WhiteSpaces allowed with Javascript Obfuscation in
HTTP responses: 200
      Normalize HTTP Cookies: NO
      Enable XFF and True Client IP: NO
```

```
      Log HTTP URI data: NO
      Log HTTP Hostname data: NO
      Extended ASCII code support in URI: NO
      Ascii: YES alert: NO
      Double Decoding: YES alert: NO
      %U Encoding: YES alert: YES
      Bare Byte: YES alert: NO
      UTF 8: YES alert: NO
      IIS Unicode: YES alert: NO
      Multiple Slash: YES alert: NO
      IIS Backslash: YES alert: NO
      Directory Traversal: YES alert: NO
      Web Root Traversal: YES alert: NO
      Apache WhiteSpace: YES alert: NO
      IIS Delimiter: YES alert: NO
      IIS Unicode Map: GLOBAL IIS UNICODE MAP CONFIG
      Non-RFC Compliant Characters: 0x00 0x01 0x02 0x03 0x04 0x05 0x06
0x07
      Whitespace Characters: 0x09 0x0b 0x0c 0x0d
rpc_decode arguments:
    Ports to decode RPC on: 111 32770 32771 32772 32773 32774 32775 32776
32777 32778 32779
    alert_fragments: INACTIVE
    alert_large_fragments: INACTIVE
    alert_incomplete: INACTIVE
    alert_multiple_requests: INACTIVE
FTPTelnet Config:
    GLOBAL CONFIG
      Inspection Type: stateful
      Check for Encrypted Traffic: YES alert: NO
      Continue to check encrypted data: YES
    TELNET CONFIG:
      Ports: 23
      Are You There Threshold: 20
      Normalize: YES
      Detect Anomalies: YES
    FTP CONFIG:
      FTP Server: default
        Ports (PAF): 21 2100 3535
        Check for Telnet Cmds: YES alert: YES
        Ignore Telnet Cmd Operations: YES alert: YES
        Ignore open data channels: NO
      FTP Client: default
        Check for Bounce Attacks: YES alert: YES
        Check for Telnet Cmds: YES alert: YES
        Ignore Telnet Cmd Operations: YES alert: YES
        Max Response Length: 256
SMTP Config:
    Ports: 25 465 587 691
    Inspection Type: Stateful
    Normalize: ATRN AUTH BDAT DATA DEBUG EHLO EMAL ESAM ESND ESOM ETRN
EVFY EXPN HELO HELP IDENT MAIL NOOP ONEX QUEU QUIT RCPT RSET SAML SEND
STARTTLS SOML TICK TIME TURN TURNME VERB VRFY X-EXPS XADR XAUTH XCIR
```

```
XEXCH50 XGEN XLICENSE X-LINK2STATE XQUE XSTA XTRN XUSR CHUNKING X-ADAT X-
DRCP X-ERCP X-EXCH50
    Ignore Data: No
    Ignore TLS Data: No
    Ignore SMTP Alerts: No
    Max Command Line Length: 512
    Max Specific Command Line Length:
       ATRN:255 AUTH:246 BDAT:255 DATA:246 DEBUG:255
       EHLO:500 EMAL:255 ESAM:255 ESND:255 ESOM:255
       ETRN:246 EVFY:255 EXPN:255 HELO:500 HELP:500
       IDENT:255 MAIL:260 NOOP:255 ONEX:246 QUEU:246
       QUIT:246 RCPT:300 RSET:246 SAML:246 SEND:246
       SIZE:255 STARTTLS:246 SOML:246 TICK:246 TIME:246
       TURN:246 TURNME:246 VERB:246 VRFY:255 X-EXPS:246
       XADR:246 XAUTH:246 XCIR:246 XEXCH50:246 XGEN:246
       XLICENSE:246 X-LINK2STATE:246 XQUE:246 XSTA:246 XTRN:246
       XUSR:246
    Max Header Line Length: 1000
    Max Response Line Length: 512
    X-Link2State Alert: Yes
    Drop on X-Link2State Alert: No
    Alert on commands: None
    Alert on unknown commands: No
    SMTP Memcap: 838860
    MIME Max Mem: 838860
    Base64 Decoding: Enabled
    Base64 Decoding Depth: Unlimited
    Quoted-Printable Decoding: Enabled
    Quoted-Printable Decoding Depth: Unlimited
    Unix-to-Unix Decoding: Enabled
    Unix-to-Unix Decoding Depth: Unlimited
    Non-Encoded MIME attachment Extraction: Enabled
    Non-Encoded MIME attachment Extraction Depth: Unlimited
    Log Attachment filename: Enabled
    Log MAIL FROM Address: Enabled
    Log RCPT TO Addresses: Enabled
    Log Email Headers: Enabled
    Email Hdrs Log Depth: 1464
SSH config:
    Autodetection: ENABLED
    Challenge-Response Overflow Alert: ENABLED
    SSH1 CRC32 Alert: ENABLED
    Server Version String Overflow Alert: ENABLED
    Protocol Mismatch Alert: ENABLED
    Bad Message Direction Alert: DISABLED
    Bad Payload Size Alert: DISABLED
    Unrecognized Version Alert: DISABLED
    Max Encrypted Packets: 20
    Max Server Version String Length: 100
    MaxClientBytes: 19600 (Default)
    Ports:
      22
DCE/RPC 2 Preprocessor Configuration
  Global Configuration
```

```
    DCE/RPC Defragmentation: Enabled
    Memcap: 102400 KB
    Events: co
    SMB Fingerprint policy: Disabled
  Server Default Configuration
    Policy: WinXP
    Detect ports (PAF)
      SMB: 139 445
      TCP: 135
      UDP: 135
      RPC over HTTP server: 593
      RPC over HTTP proxy: None
    Autodetect ports (PAF)
      SMB: None
      TCP: 1025-65535
      UDP: 1025-65535
      RPC over HTTP server: 1025-65535
      RPC over HTTP proxy: None
    Invalid SMB shares: C$ D$ ADMIN$
    Maximum SMB command chaining: 3 commands
    SMB file inspection: Disabled
DNS config:
    DNS Client rdata txt Overflow Alert: ACTIVE
    Obsolete DNS RR Types Alert: INACTIVE
    Experimental DNS RR Types Alert: INACTIVE
    Ports: 53
SSLPP config:
    Encrypted packets: not inspected
    Ports:
      443      465      563      636      989
      992      993      994      995     7801
     7802     7900     7901     7902     7903
     7904     7905     7906     7907     7908
     7909     7910     7911     7912     7913
     7914     7915     7916     7917     7918
     7919     7920
    Server side data is trusted
    Maximum SSL Heartbeat length: 0
Sensitive Data preprocessor config:
    Global Alert Threshold: 25
    Masked Output: DISABLED
SIP config:
    Max number of sessions: 40000
    Max number of dialogs in a session: 4 (Default)
    Status: ENABLED
    Ignore media channel: DISABLED
    Max URI length: 512
    Max Call ID length: 80
    Max Request name length: 20 (Default)
    Max From length: 256 (Default)
    Max To length: 256 (Default)
    Max Via length: 1024 (Default)
    Max Contact length: 512
    Max Content length: 2048
```

```
    Ports:
      5060  5061  5600
    Methods:
        invite cancel ack bye register options refer subscribe update
join info message notify benotify do qauth sprack publish service
unsubscribe prack
IMAP Config:
    Ports: 143
    IMAP Memcap: 838860
    MIME Max Mem: 838860
    Base64 Decoding: Enabled
    Base64 Decoding Depth: Unlimited
    Quoted-Printable Decoding: Enabled
    Quoted-Printable Decoding Depth: Unlimited
    Unix-to-Unix Decoding: Enabled
    Unix-to-Unix Decoding Depth: Unlimited
    Non-Encoded MIME attachment Extraction: Enabled
    Non-Encoded MIME attachment Extraction Depth: Unlimited
POP Config:
    Ports: 110
    POP Memcap: 838860
    MIME Max Mem: 838860
    Base64 Decoding: Enabled
    Base64 Decoding Depth: Unlimited
    Quoted-Printable Decoding: Enabled
    Quoted-Printable Decoding Depth: Unlimited
    Unix-to-Unix Decoding: Enabled
    Unix-to-Unix Decoding Depth: Unlimited
    Non-Encoded MIME attachment Extraction: Enabled
    Non-Encoded MIME attachment Extraction Depth: Unlimited
Modbus config:
    Ports:
      502
DNP3 config:
    Memcap: 262144
    Check Link-Layer CRCs: ENABLED
    Ports:
      20000

++++++++++++++++++++++++++++++++++++++++++++++++++
Initializing rule chains...
WARNING: /etc/snort/rules/chat.rules(33) threshold (in rule) is
deprecated; use detection_filter instead.

1731 Snort rules read
    1731 detection rules
    0 decoder rules
    0 preprocessor rules
1731 Option Chains linked into 250 Chain Headers
0 Dynamic rules
++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]-------------------------------------
----
```

```
|             tcp     udp    icmp      ip
|     src     108      18       0       0
|     dst     932     126       0       0
|     any     381      48     145      22
|      nc      25       8      94      20
|     s+d      12       5       0       0
+-------------------------------------------------------------------------
----

+----------------------[detection-filter-config]-----------------------
------
| memory-cap : 1048576 bytes
+----------------------[detection-filter-rules]------------------------
------
| none
--------------------------------------------------------------------------
------

+----------------------[rate-filter-config]----------------------------
------
| memory-cap : 1048576 bytes
+----------------------[rate-filter-rules]-----------------------------
------
| none
--------------------------------------------------------------------------
------

+----------------------[event-filter-config]---------------------------
------
| memory-cap : 1048576 bytes
+----------------------[event-filter-global]---------------------------
------
| none
+----------------------[event-filter-local]----------------------------
------
| gen-id=1      sig-id=3152       type=Threshold tracking=src count=5
seconds=2
| gen-id=1      sig-id=1991       type=Limit     tracking=src count=1
seconds=60
| gen-id=1      sig-id=2923       type=Threshold tracking=dst count=10
seconds=60
| gen-id=1      sig-id=3273       type=Threshold tracking=src count=5
seconds=2
| gen-id=1      sig-id=2496       type=Both      tracking=dst count=20
seconds=60
| gen-id=1      sig-id=2275       type=Threshold tracking=dst count=5
seconds=60
| gen-id=1      sig-id=2494       type=Both      tracking=dst count=20
seconds=60
| gen-id=1      sig-id=2523       type=Both      tracking=dst count=10
seconds=10
| gen-id=1      sig-id=2495       type=Both      tracking=dst count=20
seconds=60
```

```
| gen-id=1      sig-id=2924       type=Threshold tracking=dst count=10
seconds=60
+-----------------------[suppression]-----------------------------------
------
| none
-------------------------------------------------------------------------
------
Rule application order: activation->dynamic->pass->drop->sdrop->reject-
>alert->log
Verifying Preprocessor Configurations!
WARNING: flowbits key 'sslv2.server_hello.request' is checked but not
ever set.
WARNING: flowbits key 'tlsv1.server_hello.request' is checked but not
ever set.
WARNING: flowbits key 'realplayer.playlist' is set but not ever checked.
WARNING: flowbits key 'smb.tree.create.llsrpc' is set but not ever
checked.
WARNING: flowbits key 'tlsv1.client_hello.request' is checked but not
ever set.
WARNING: flowbits key 'ms_sql_seen_dns' is checked but not ever set.
WARNING: flowbits key 'sslv2.client_hello.request' is checked but not
ever set.
31 out of 1024 flowbits in use.

[ Port Based Pattern Matching Memory ]
+- [ Aho-Corasick Summary ] ------------------------------------
| Storage Format    : Full-Q
| Finite Automaton  : DFA
| Alphabet Size     : 256 Chars
| Sizeof State      : Variable (1,2,4 bytes)
| Instances         : 210
|     1 byte states : 204
|     2 byte states : 6
|     4 byte states : 0
| Characters        : 18500
| States            : 8055
| Transitions       : 104342
| State Density     : 5.1%
| Patterns          : 1686
| Match States      : 1011
| Memory (MB)       : 3.28
|   Patterns        : 0.11
|   Match Lists     : 0.15
|   DFA
|     1 byte states : 1.21
|     2 byte states : 1.59
|     4 byte states : 0.00
+-------------------------------------------------------------
[ Number of patterns truncated to 20 bytes: 380 ]
pcap DAQ configured to passive.
Acquiring network traffic from "eth1".
Reload thread starting...
Reload thread started, thread 0x5da19460 (778)
Decoding Ethernet
```

```
        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.7.0 GRE (Build 149)
   ''''    By Martin Roesch & The Snort Team:
http://www.snort.org/contact#team
           Copyright (C) 2014 Cisco and/or its affiliates. All rights
reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.8.1
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.2.8

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 2.4  <Build
1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
Commencing packet processing (pid=773)
```

==========

2.2. Alertas Snort
Vamos a realizar algunas pruebas sencillas para ver cómo se activan las notificaciones en snort.

1. Desde kali2 ejecuta un ping a 10.100.X.10 con el envıó de un único paquete. Observa las alertas que muestra snort y ve al fichero de definición de esa/s regla/s, copia la/s regla/s y explica qué condiciones se han cumplido para que se activen.

nos vamos a kali2, y hacemos ping -c 1 10.100.18.10, ya que solo queremos enviar un unico paquete a nuestro pc:

```
root@kali2:~# ping -c 1 10.100.18.10
PING 10.100.18.10 (10.100.18.10) 56(84) bytes of data.
64 bytes from 10.100.18.10: icmp_seq=1 ttl=63 time=1.42 ms

--- 10.100.18.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.428/1.428/1.428/0.000 ms
```

ahora en kali1 vemos lo siguiente:

```
04/22-12:13:32.281581  [**] [1:368:6] ICMP PING BSDtype [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/22-12:13:32.281581  [**] [1:366:7] ICMP PING *NIX [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/22-12:13:32.281581  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/22-12:13:32.282351  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
```

Vemos que se ha activado el mismo tipo de mensaje: Misc activity.
El primero se debe a que lo ha provocado un sistema BSD, este caso, Kali.
El segundo es debido a que ha detectado un ping de cualquier tipo.
El tercero es porque ha detectado un mensaje icmp, del echo request de kali2.
El cuarto es debido al icmp de respuesta de mi pc.

---

2. Desde kali2 vuelve a ejecutar el mismo ping pero con tamaño de paquete 1000 bytes (-s 1000). Observa
las alertas que muestra snort y ve al fichero de definición de esa/s regla/s, copia la/s regla/s y explica
qué condiciones se han cumplido para que se activen.

En kali2 lanzamos el ping hacia el pc:

```
root@kali2:~# ping -c 1 -s 1000 10.100.18.10
PING 10.100.18.10 (10.100.18.10) 1000(1028) bytes of data.
1008 bytes from 10.100.18.10: icmp_seq=1 ttl=63 time=1.79 ms

--- 10.100.18.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.797/1.797/1.797/0.000 ms
```

miramos en kali1 para ver los mensajes:

```
04/22-12:32:49.354601  [**] [1:368:6] ICMP PING BSDtype [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/22-12:32:49.354601  [**] [1:366:7] ICMP PING *NIX [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/22-12:32:49.354601  [**] [1:480:5] ICMP PING speedera [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/22-12:32:49.354601  [**] [1:499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
10.100.19.2 -> 10.100.18.10
```

```
04/22-12:32:49.354601  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/22-12:32:49.355457  [**] [1:499:4] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2] {ICMP}
10.100.18.10 -> 10.100.19.2
04/22-12:32:49.355457  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
```

Ha mostrado los mismos mensajes que en el apartado anterior pero esta vez, añadiendo un par de mensajes de tipo: Potentially Bad Traffic, ya que el tamaño del mensaje icmp es demasiado grande.

---

3. Desde kali2 vuelve a ejecutar el mismo ping pero con tamaño de paquete 0 bytes (-s 0). Observa las alertas que muestra snort y ve al fichero de definición de esa/s regla/s, copia la/s regla/s y explica qué condiciones se han cumplido para que se activen.

en kali2:

```
root@kali2:~# ping -c 1 -s 0 10.100.18.10
PING 10.100.18.10 (10.100.18.10) 0(28) bytes of data.
8 bytes from 10.100.18.10: icmp_seq=1 ttl=63

--- 10.100.18.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

en kali1:

```
04/22-12:37:14.976617  [**] [1:469:3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2] {ICMP}
10.100.19.2 -> 10.100.18.10
04/22-12:37:14.976617  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/22-12:37:14.977416  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
```

Al detectar el tamaño de los paquetes, ha pensado que se trata de un mensaje enviado con nmap.

---

4. En la carpeta /var/log/snort se quedan almacenados ficheros snort.log.*. Estos ficheros contienen los
paquetes que han generado las alertas que se han mostrado en snort. Interpreta estos ficheros cargándolos
con tcpdump y la opción -r <nombreFichero>.

```
root@kali1:/var/log/snort# tcpdump -r snort.log.1524399205
reading from file snort.log.1524399205, link-type EN10MB (Ethernet)
12:13:32.281581 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 956,
seq 1, length 64
12:13:32.281581 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 956,
seq 1, length 64
12:13:32.281581 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 956,
seq 1, length 64
12:13:32.282351 IP 10.100.18.10 > 10.100.19.2: ICMP echo reply, id 956,
seq 1, length 64
12:32:49.354601 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 978,
seq 1, length 1008
12:32:49.354601 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 978,
seq 1, length 1008
12:32:49.354601 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 978,
seq 1, length 1008
12:32:49.354601 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 978,
seq 1, length 1008
12:32:49.354601 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 978,
seq 1, length 1008
12:32:49.355457 IP 10.100.18.10 > 10.100.19.2: ICMP echo reply, id 978,
seq 1, length 1008
12:32:49.355457 IP 10.100.18.10 > 10.100.19.2: ICMP echo reply, id 978,
seq 1, length 1008
12:37:14.976617 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 980,
seq 1, length 8
12:37:14.976617 IP 10.100.19.2 > 10.100.18.10: ICMP echo request, id 980,
seq 1, length 8
12:37:14.977416 IP 10.100.18.10 > 10.100.19.2: ICMP echo reply, id 980,
seq 1, length 8
```

Vemos que las 4 primeras lineas se corresponden con el primer ping que hemos hecho. Las 6 siguientes se corres
ponden con el segundo ping que hemos realizado, de tamaño 1000. Por ultimo, las tres ultimas lineas son del
ultimo ping, de tamaño cero.

++

3. Pentesting con nmap

Lanzamos en kali1 lo siguiente, si habiamos parado snort: snort -A console -c /etc/snort/snort.conf -i eth1

3.1. Sondeo de equipos

1. Desde kali2 ejecuta nmap para sondear qué equipos están activos en la subredes 10.100.(X+1).0/24. Realiza una captura en kali2(eth0) con la opción -n 3 y guarda el contenido en un fichero nmap-01.cap y después arranca el sondeo en kali2.

Lanzamos la captura en kali2:

```
root@kali2:~# tcpdump -i eth0 -n -w nmap-01.cap

En otro terminal de kali2, lanzamos el nmap:
root@kali2:~# nmap -sP -PR -n 10.100.19.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 14:40 UTC
Nmap scan report for 10.100.19.1
Host is up (0.0011s latency).
MAC Address: 00:14:5C:97:72:62 (Intronics)
Nmap scan report for 10.100.19.2
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 2.31 seconds
```

a) Explica la salida que muestra nmap.

vemos que nos avisa de que hay dos equipos activos, kali1 (10.100.19.1) y la propia kali2
(10.100.19.2)

b) Interrumpe la captura y explica qué paquetes se han enviado y cuáles tienen respuesta.

Copiamos la captura desde el terminal de nuestro pc hacia kali2:
$ scp root@10.0.0.10:nmap-01.cap . (el punto indica el directorio actual desde el que lanzamos el comando)

vemos que hay un total de 512 paquetes.

si nos fijamos, los mensajes son de tipo ARP. Nos avisa de que en 10.100.19.1 hay una raspberry (kali1)

```
11    0.001029    Intronic_97:72:62Raspberr_65:a8:e4ARP   60
      10.100.19.1 is at 00:14:5c:97:72:62
```

c) ¿Se muestra alguna alerta en snort? Explica tu respuesta.

No, no aparece ninguna alerta. No debe de haber ninguna regla en snort kali1.

---

2. Desde kali2 ejecuta nmap para sondear un único equipo de su misma subred, kali1. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-02.cap y después arranca el sondeo en kali2.

a) Explica la salida que muestra nmap.

```
root@kali2:~# nmap -sP -n 10.100.19.2  10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 15:07 UTC
Nmap scan report for 10.100.19.2
Host is up.
Nmap scan report for 10.100.19.1
Host is up (0.00065s latency).
```

```
MAC Address: 00:14:5C:97:72:62 (Intronics)
Nmap done: 2 IP addresses (2 hosts up) scanned in 0.34 seconds
```

vemos que identifica si una maquina esta o no levantada. En este caso, nos dice que, de las dos direcciones que le hemos pasado a nmap, las dos estan activas.

b) Interrumpe la captura y explica qué paquetes se han enviado y cuáles tienen respuesta.

aparecen solamente los paquetes ARP contestando a nmap kali2. Es como la version reducida de la captura anterior (ya que hemos ido directamente a una maquina en concreto en no a toda la subred)

c) ¿Se muestra alguna alerta en snort? Explica tu respuesta.

Si antes no hemos visto nada con snort por el tipo de mensajes ARP, ahora tampoco ya que tambien se han empleado este tipo de mensajes.

---

3. Desde kali2 ejecuta nmap para sondear un único equipo de otra subred diferente, tu máquina. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-03.cap y después arranca el sondeo en kali2.

Lanzamos la tercera captura en kali2:

```
root@kali2:~# tcpdump -i eth0 -n -s 0 -w nmap-03.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size
262144 bytes
^C12 packets captured
12 packets received by filter
0 packets dropped by kernel
```

a) Explica la salida que muestra nmap.

Ahora lanzamos nmap hacia la direccion 10.100.18.10 (la de nuestro PC):

```
root@kali2:~# nmap -sP -n 10.100.18.10
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 15:23 UTC
Nmap scan report for 10.100.18.10
Host is up (0.0021s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

vemos que aparece que el host esta conectado (nuestto pc)

b) Interrumpe la captura y explica qué paquetes se han enviado y cuáles tienen respuesta.

Vemos que se han enviados mensajes ICMP echo request por parte de kali2. Ahora snort si detecta este tipo de mensajes, a diferencia de los ARP (dentro de la subred entre kali1 y kali2)
Si nos fijamos, nuestro pc le envia mensajes ICMP con el flag de RST activado. Esto diria que se debe a que como no hay ningun puerto abierto justo donde esta intentando hacer la peticion, mi pc le responde con un RST.

c) ¿Se muestra alguna alerta en snort? Explica tu respuesta.

Ahora si vemos que aparecen mensajes en snort kali1:

```
04/27-15:23:30.520224  [**] [1:469:3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2] {ICMP}
10.100.19.2 -> 10.100.18.10
04/27-15:23:30.520224  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/27-15:23:30.520513  [**] [1:453:5] ICMP Timestamp Request [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/27-15:23:30.521603  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/27-15:23:30.521770  [**] [1:451:5] ICMP Timestamp Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
```

Vemos que como los mensajes son ICMP, snort si los detecta, a diferencia de los mensajes ARP que se generaban en la subred entre kali1 y kali2.

==========

3.2. Sondeo TCP SYN

Realiza una captura en lkali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-04.cap. Utiliza nmap desde kali2 de la siguiente forma para que se envıén segmentos TCP con el flag de SYN activo con el objetivo de determinar si hay un servicio esperando recibir paquetes entre los puertos 1 a 50 de kali1.

1. Explica la salida que muestra nmap

```
root@kali2:~# nmap -sS -Pn -p 1-50 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 15:59 UTC
Initiating ARP Ping Scan at 15:59
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 15:59, 0.09s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 15:59
Scanning 10.100.19.1 [50 ports]
Discovered open port 22/tcp on 10.100.19.1
Completed SYN Stealth Scan at 15:59, 0.09s elapsed (50 total ports)
Nmap scan report for 10.100.19.1
Host is up (0.0016s latency).
```

```
Not shown: 49 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:14:5C:97:72:62 (Intronics)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.92 seconds
           Raw packets sent: 51 (2.228KB) | Rcvd: 51 (2.032KB)
```

nos esta avisando de que en kali1, el puerto 22 esta abierto.

---

2. Interrumpe la captura y explica los paquetes intercambiados. Explica las diferencias del sondeo del puerto 22 y el resto de puertos.

Vemos que a diferencia del resto de puertos, con el 22, como esta abierto, pero nmap no quiere establecer una comunicacion, le envia de vuelta un mensaje TCP con el flag RST. El resto de mensajes llevan ademas del RST, el flag ACK.

---

3. Explica si snort ha detectado alertas e indica cuáles y por qué

Como se tratan de mensajes con el flag SYN, snort no los identifica como posibles amenazas, ya que si eso fuese asi, cualquier intento de conexion no malicioso podria ser tratado como tal.

---

4. Consulta los servicios TCP activos (los servidores que se encuentran esperando paquetes TCP) en la máquina kali1 a la que estabas realizando el sondeo, utilizando el comando netstat -nt4l. Relaciona el resultado de la ejecución de este comando con el resultado del sondeo.

comando: netstat -nt4l
Al meter el comando en kali1 nos sale lo siguiente:

```
root@kali1:~# netstat -nt4l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN
```

lo cual no nos llega de primeras ya que con nmap hemos descubierto que efectivamente el puerto 22 esta esperando a recibir peticiones.

---

5. Una vez encontrado un puerto abierto, puede ser útil obtener información de la versión del servicio que se está ejecutando. Prueba a realizar el sondeo anterior únicamente en el puerto que has encontrado abierto y añadiendo la opción -sV. Previamente a realizar el sondeo realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-05.cap. Estudia el resultado de nmap y el contenido de la captura y explićalos.

lanzamos la quinta captura en kali2:

```
root@kali2:~# tcpdump -i eth0 -n -s 0 -w nmap-05.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size
262144 bytes
^C13 packets captured
13 packets received by filter
0 packets dropped by kernel
```

y lanzamos en kali2 lo siguiente:

```
root@kali2:~# nmap -sS -Pn -sV -p 22 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 17:17 UTC
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 17:17
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 17:17, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 17:17
Scanning 10.100.19.1 [1 port]
Discovered open port 22/tcp on 10.100.19.1
Completed SYN Stealth Scan at 17:17, 0.08s elapsed (1 total ports)
Initiating Service scan at 17:17
Scanning 1 service on 10.100.19.1
Completed Service scan at 17:17, 0.08s elapsed (1 service on 1 host)
NSE: Script scanning 10.100.19.1.
Initiating NSE at 17:17
Completed NSE at 17:17, 0.02s elapsed
Initiating NSE at 17:17
Completed NSE at 17:17, 0.00s elapsed
Nmap scan report for 10.100.19.1
Host is up (0.00067s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Debian 3 (protocol 2.0)
MAC Address: 00:14:5C:97:72:62 (Intronics)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.28 seconds
           Raw packets sent: 2 (72B) | Rcvd: 2 (72B)
```

vemos que kali2 le envia a kali1, un primer SYN, pero no quiere establecer una conexion con el (es lo mismo de antes; esta descubriendo puertos)

justo despues podemos ver que vuelve a hacer lo mismo, pero ahora si con intencion de conectarse al puerto 22. Vemos el intercambio de mensajes, con el SYN, SYN + ACK, ACK. Podemos observar inmediatamente despues que se ve el tipo de servicio que esta ofreciendo:
9 1.211359 10.100.19.2 SSH 98 10.100.19.1 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-3)

vemos que es justo lo que ya nos adelantaba nmap.

---

6. Vamos a arrancar un servidor de web en la máquina kali1 escuchando peticiones HTTP en el puerto 80.
Para que este servidor sólo use IPv4 hay que modificar el siguiente fichero /etc/apache2/ports.conf y cambiar la línea Listen 80 por Listen 0.0.0.0:80. Inicia el servidor con el siguiente comando:
/etc/init.d/apache2 start

una vez editamos el fichero (nano /etc/apache2/ports.conf), y arrancado el servicio(/etc/init.d/apache2 start), hacemos netstat -nt4l, en kali1, donde hemos lanzado el servicio.

```
root@kali1:/etc/apache2# netstat -nt4l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN
```

vemos que ahora aparece el nuevo servicio.

si se volviese a sondear los puertos, apareceria el puerto 80, junto con el 22 como puertos que estan esperando trafico.

---

7. Prueba a realizar el sondeo anterior en el puerto 80 añadiendo la opción -sV. Previamente a realizar el sondeo realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-06.cap. Estudia el resultado de nmap y el contenido de la captura y explícalos.

lanzamos la captura en kali2:

```
root@kali2:~# tcpdump -i eth0 -n -s 0 -w nmap-06.cap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size
262144 bytes
^C279 packets captured
279 packets received by filter
0 packets dropped by kernel
```

lanzamos el sondeo en kali2:

```
root@kali2:~# nmap -sS -Pn -sV -p 20-100 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 17:51 UTC
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 17:51
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 17:51, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 17:51
Scanning 10.100.19.1 [81 ports]
Discovered open port 22/tcp on 10.100.19.1
Discovered open port 80/tcp on 10.100.19.1
Completed SYN Stealth Scan at 17:51, 0.09s elapsed (81 total ports)
Initiating Service scan at 17:51
Scanning 2 services on 10.100.19.1
Completed Service scan at 17:51, 6.19s elapsed (2 services on 1 host)
NSE: Script scanning 10.100.19.1.
Initiating NSE at 17:51
Completed NSE at 17:51, 0.08s elapsed
Initiating NSE at 17:51
Completed NSE at 17:51, 0.00s elapsed
Nmap scan report for 10.100.19.1
Host is up (0.0016s latency).
Not shown: 79 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Debian 3 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.29 ((Debian))
MAC Address: 00:14:5C:97:72:62 (Intronics)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.29 seconds
           Raw packets sent: 82 (3.592KB) | Rcvd: 82 (3.276KB)
```

analizando las capturas, vemos que aparece el mismo mensaje(de entre la gran cantidad que hay) que indica que en el puerto 22 hay un servicio ssh.

```
173   1.229515    10.100.19.2 SSH   98    10.100.19.1 Server: Protocol
(SSH-2.0-OpenSSH_7.6p1 Debian-3)

Frame 173: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: Intronic_97:72:62 (00:14:5c:97:72:62), Dst:
Raspberr_65:a8:e4 (b8:27:eb:65:a8:e4)
Internet Protocol Version 4, Src: 10.100.19.1, Dst: 10.100.19.2
Transmission Control Protocol, Src Port: 22, Dst Port: 38594, Seq: 1,
Ack: 1, Len: 32
    Source Port: 22
    Destination Port: 38594
    [Stream index: 81]
    [TCP Segment Len: 32]
```

```
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 33    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 32 bytes
    Flags: 0x018 (PSH, ACK)
    Window size value: 227
    [Calculated window size: 29056]
    [Window size scaling factor: 128]
    Checksum: 0x1ec6 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP),
Timestamps
    [SEQ/ACK analysis]
SSH Protocol
```

y tambien vemos que hay otro servicio, en este caso http, de apache, en el puerto 80.
hemos visto que se ha establecido la conexion (SYN, SYN + ACK, ACK):

```
178   7.167807    10.100.19.1 HTTP  84    10.100.19.2 GET / HTTP/1.0

Frame 178: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
Ethernet II, Src: Raspberr_65:a8:e4 (b8:27:eb:65:a8:e4), Dst:
Intronic_97:72:62 (00:14:5c:97:72:62)
Internet Protocol Version 4, Src: 10.100.19.2, Dst: 10.100.19.1
Transmission Control Protocol, Src Port: 48482, Dst Port: 80, Seq: 1,
Ack: 1, Len: 18
    Source Port: 48482
    Destination Port: 80
    [Stream index: 82]
    [TCP Segment Len: 18]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 19    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Header Length: 32 bytes
    Flags: 0x018 (PSH, ACK)
    Window size value: 229
    [Calculated window size: 29312]
    [Window size scaling factor: 128]
    Checksum: 0x3b03 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
    Options: (12 bytes), No-Operation (NOP), No-Operation (NOP),
Timestamps
    [SEQ/ACK analysis]
Hypertext Transfer Protocol
```

==========

3.3. Sondeo UDP

Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-07.cap.

Utiliza nmap desde kali2 de la siguiente forma para que se envıén paquetes UDP con el objetivo de determinar si hay un servicio esperando recibir paquetes entre los puertos 60 a 70 de kali1.

1. Explica la salida que muestra nmap.

```
root@kali2:~# nmap -Pn -sU -p 60-70 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-27 18:03 UTC
Initiating ARP Ping Scan at 18:03
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 18:03, 0.09s elapsed (1 total hosts)
Initiating UDP Scan at 18:03
Scanning 10.100.19.1 [11 ports]
Completed UDP Scan at 18:03, 4.59s elapsed (11 total ports)
Nmap scan report for 10.100.19.1
Host is up (0.00085s latency).

PORT   STATE         SERVICE
60/udp closed        unknown
61/udp closed        ni-mail
62/udp closed        acas
63/udp closed        via-ftp
64/udp closed        covia
65/udp closed        tacacs-ds
66/udp closed        sqlnet
67/udp closed        dhcps
68/udp open|filtered dhcpc
69/udp closed        tftp
70/udp closed        gopher
MAC Address: 00:14:5C:97:72:62 (Intronics)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.38 seconds
           Raw packets sent: 25 (700B) | Rcvd: 11 (588B)
```

vemos que muestra que todos los puertos estan cerrados, salvo el puerto 68, con servicio dhcpc.

---

2. Interrumpe la captura y explica los paquetes intercambiados. Explica las diferencias del sondeo del puerto
68 y el resto de puertos.

Si nos fijamos en los mensajes de port unreachable, nos fijamos aqui:
User Datagram Protocol, Src Port: 35137, Dst Port: XX
donde XX significa el puerto en cuestion. Si nos fijamos, el unico puerto que no aparece como unreachable es el 68.

---

3. Explica si snort ha detectado alertas e indica cuáles y por qué.

```
[**] [Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.1 ->
10.100.19.2
04/27-18:03:50.409560  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:50.409655  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:50.409773  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:50.409873  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:50.409970  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:51.510625  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:52.612192  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:53.713346  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/27-18:03:54.814703  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
```

Si nos fijamos, snort ha detectado mensajes ICMP, como suele ser habitual, en este caso de destino inalcanzable.

4. Consulta los servicios UDP activos (los servidores que se encuentran esperando paquetes UDP) en la máquina kali1 a la que estabas realizando el sondeo, utilizando el comando netstat -nu4l. Relaciona el resultado de la ejecución de este comando con el resultado del sondeo.

```
root@kali1:/etc/apache2# netstat -nu4l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:68              0.0.0.0:*
```

como vemos, hay un servicio udp en el puerto 68. Es por eso que no salia destino inalcanzable, en ese puerto.

==========

3.4. Sondeo TCP FIN, Xmas

1. Desde kali2 ejecuta nmap para realizar un ataque FIN TCP. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un

fichero nmap-08.cap y después arranca el sondeo en kali2 para los puertos 20-25.
a) Explica la salida que muestra nmap.

lanzamos el siguiente comando en kali2:

```
root@kali2:~# nmap -sF -Pn -sV -p 20-25 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-28 10:54 UTC
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 10:54
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 10:54, 0.08s elapsed (1 total hosts)
Initiating FIN Scan at 10:54
Scanning 10.100.19.1 [6 ports]
Completed FIN Scan at 10:55, 1.31s elapsed (6 total ports)
Initiating Service scan at 10:55
Scanning 1 service on 10.100.19.1
Discovered open port 22/tcp on 10.100.19.1
Discovered open|filtered port 22/tcp on 10.100.19.1 is actually open
Completed Service scan at 10:55, 0.07s elapsed (1 service on 1 host)
NSE: Script scanning 10.100.19.1.
Initiating NSE at 10:55
Completed NSE at 10:55, 0.02s elapsed
Initiating NSE at 10:55
Completed NSE at 10:55, 0.00s elapsed
Nmap scan report for 10.100.19.1
Host is up (0.0011s latency).

PORT   STATE  SERVICE   VERSION
20/tcp closed ftp-data
21/tcp closed ftp
22/tcp open   ssh       OpenSSH 7.6p1 Debian 3 (protocol 2.0)
23/tcp closed telnet
24/tcp closed priv-mail
25/tcp closed smtp
MAC Address: 00:14:5C:97:72:62 (Intronics)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.65 seconds
           Raw packets sent: 8 (308B) | Rcvd: 6 (228B)
```

Nos muestra que el puerto 22 esta abierto: 22/tcp open ssh OpenSSH 7.6p1 Debian 3 (protocol 2.0)

b) Interrumpe la captura y explica qué paquetes se han enviado y cuáles tienen respuesta.

Nos fijamos que aquellos puertos que estan cerrados, han recibido un mensaje con el flag RST activado; esto nos dice que el puerto en cuestion esta cerrado.

El unico que no recibe respuesta es el 22, por lo que nos lleva a pensar que el puerto esta abierto.

c) ¿Se muestra alguna alerta en snort? Explica tu respuesta.

```
04/28-10:54:59.118385  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:22
04/28-10:54:59.118422  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:23
04/28-10:54:59.118519  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:21
04/28-10:54:59.118598  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:25
04/28-10:54:59.118672  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:20
04/28-10:54:59.118745  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57375 ->
10.100.19.1:24
04/28-10:55:00.218809  [**] [1:621:7] SCAN FIN [**] [Classification:
Attempted Information Leak] [Priority: 2] {TCP} 10.100.19.2:57376 ->
10.100.19.1:22
```

Si, snort detecta como Attempted Information Leak, de prioridad 2, todos los mensajes que han sido enviados con el flag de FIN activado.

---

2. Desde kali2 ejecuta nmap para realizar un ataque XMAS TCP. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-09.cap y después arranca el sondeo en kali2 para los puertos 20-25.

a) Explica la salida que muestra nmap.

```
root@kali2:~# nmap -sX -Pn -sV -p 20-25 -n -v 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-28 12:03 UTC
NSE: Loaded 43 scripts for scanning.
Initiating ARP Ping Scan at 12:03
Scanning 10.100.19.1 [1 port]
Completed ARP Ping Scan at 12:03, 0.09s elapsed (1 total hosts)
Initiating XMAS Scan at 12:03
Scanning 10.100.19.1 [6 ports]
Completed XMAS Scan at 12:03, 1.29s elapsed (6 total ports)
Initiating Service scan at 12:03
Scanning 1 service on 10.100.19.1
```

```
Discovered open port 22/tcp on 10.100.19.1
Discovered open|filtered port 22/tcp on 10.100.19.1 is actually open
Completed Service scan at 12:03, 0.08s elapsed (1 service on 1 host)
NSE: Script scanning 10.100.19.1.
Initiating NSE at 12:03
Completed NSE at 12:03, 0.02s elapsed
Initiating NSE at 12:03
Completed NSE at 12:03, 0.00s elapsed
Nmap scan report for 10.100.19.1
Host is up (0.00068s latency).

PORT   STATE  SERVICE   VERSION
20/tcp closed ftp-data
21/tcp closed ftp
22/tcp open   ssh       OpenSSH 7.6p1 Debian 3 (protocol 2.0)
23/tcp closed telnet
24/tcp closed priv-mail
25/tcp closed smtp
MAC Address: 00:14:5C:97:72:62 (Intronics)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.15 seconds
           Raw packets sent: 8 (308B) | Rcvd: 6 (228B)
```

vemos que nos muestra como abierto el puerto 22, al igual que en el apartado anterior.

b) Interrumpe la captura y explica qué paquetes se han enviado y cuáles tienen respuesta.

Nos ocurre igual que en apartado anterior. Los puertos cerrados son respondidos con un RST.

c) ¿Se muestra alguna alerta en snort? Explica tu respuesta.

```
04/28-12:03:06.665412  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:23
04/28-12:03:06.665670  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:25
04/28-12:03:06.665736  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:22
04/28-12:03:06.665784  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:21
```

```
04/28-12:03:06.665836  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:24
04/28-12:03:06.665909  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42303 -> 10.100.19.1:20
04/28-12:03:07.766974  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:42304 -> 10.100.19.1:22
```

Al detectar los flags que tiene activado, y al ver que no se ha establecido una comunicacion previa, muestra mensajes de SCAN nmap XMAS.

==========

3.5. Sondeos exhaustivos

1. Desde kali2 vamos a realizar un análisis exhaustivo de kali1. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-10.cap y después arranca el sondeo en kali2 hacia la máquina kali1. Explica el resultado que muestra nmap. Carga la captura y comenta algún aspecto
relevante que veas.

```
root@kali2:~# nmap -A -n 10.100.19.1
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-28 14:16 UTC
Nmap scan report for 10.100.19.1
Host is up (0.00081s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Debian 3 (protocol 2.0)
| ssh-hostkey:
|   2048 e0:03:bc:49:8c:7c:db:1c:69:27:d9:5e:0a:a3:0d:70 (RSA)
|   256 d9:67:5d:c1:cf:d7:8d:8e:94:ad:87:b5:96:e4:85:67 (ECDSA)
|_  256 3b:2f:d1:6d:61:c6:a0:d2:9d:85:0b:3a:47:f4:7e:a2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Debian))
|_http-server-header: Apache/2.4.29 (Debian)
|_http-title: Apache2 Debian Default Page: It works
MAC Address: 00:14:5C:97:72:62 (Intronics)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.81 ms 10.100.19.1

OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.45 seconds
```

Respecto a las otras capturas, se nota que ne esta, se ha hecho un sondeo exhaustivo, ya que aparecen muchisimos paquetes. La mayoria son de intentos de conexion hacia los puertos de kali1, el cual responde con un RST.
Podemos ver que los puertos 80, 22 tienen algun tipo de servicio levantado (http (apache), ssh)

```
04/28-14:16:48.574782  [**] [1:1418:11] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:50678 -> 10.100.19.1:161
04/28-14:16:48.597838  [**] [1:1421:11] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:50678 -> 10.100.19.1:705
04/28-14:16:56.364955  [**] [1:365:8] ICMP PING undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.19.1
04/28-14:16:56.365024  [**] [1:409:7] ICMP Echo Reply undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.1 ->
10.100.19.2
04/28-14:16:56.390028  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.19.1
04/28-14:16:56.390066  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.1 ->
10.100.19.2
04/28-14:16:56.415203  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
04/28-14:16:56.590723  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:44193 -> 10.100.19.1:1
```

En snort podemos ver diferentes tipos de mensajes, como es normal, debido al tipo de examen al que se ha sometido a la kali1.

. Desde kali2 vamos a realizar un análisis exhaustivo de tu ordenador en la interfaz 10.100.X.10. Realiza
una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-11.cap y después
arranca el sondeo en kali2 hacia tu máquina. Explica el resultado que muestra nmap. Carga la captura
y comenta algún aspecto relevante que veas.

---

2. Desde kali2 ejecuta nmap para sondear un único equipo de su misma subred, kali1. Realiza una captura en kali2(eth0) con la opción -n y guarda el contenido en un fichero nmap-02.cap y después arranca el sondeo en kali2.

a) Explica la salida que muestra nmap.

```
root@kali2:~# nmap -A -n 10.100.18.10
Starting Nmap 7.70 ( https://nmap.org ) at 2018-04-28 14:35 UTC
Nmap scan report for 10.100.18.10
Host is up (0.0017s latency).
All 1000 scanned ports on 10.100.18.10 are closed
Too many fingerprints match this host to give specific OS details
Network Distance: 2 hops

TRACEROUTE (using port 1723/tcp)
HOP RTT     ADDRESS
1   0.72 ms 10.100.19.1
2   1.83 ms 10.100.18.10

OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.10 seconds


04/28-14:36:00.335020  [**] [1:469:3] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2] {ICMP}
10.100.19.2 -> 10.100.18.10
04/28-14:36:00.335020  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/28-14:36:00.335266  [**] [1:453:5] ICMP Timestamp Request [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/28-14:36:00.335891  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/28-14:36:00.336071  [**] [1:451:5] ICMP Timestamp Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/28-14:36:00.595327  [**] [1:1421:11] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:61354 -> 10.100.18.10:705
04/28-14:36:00.668823  [**] [1:1418:11] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:61354 -> 10.100.18.10:161
04/28-14:36:01.725050  [**] [1:365:8] ICMP PING undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/28-14:36:01.726039  [**] [1:409:7] ICMP Echo Reply undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/28-14:36:01.750077  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/28-14:36:01.750597  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
```

```
04/28-14:36:01.775910  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.18.10 -> 10.100.19.2
04/28-14:36:01.850408  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:61573 -> 10.100.18.10:1
04/28-14:36:02.950224  [**] [1:365:8] ICMP PING undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.19.2 ->
10.100.18.10
04/28-14:36:02.950810  [**] [1:409:7] ICMP Echo Reply undefined code [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/28-14:36:02.975246  [**] [1:384:5] ICMP PING [**] [Classification:
Misc activity] [Priority: 3] {ICMP} 10.100.19.2 -> 10.100.18.10
04/28-14:36:02.975781  [**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3] {ICMP} 10.100.18.10 ->
10.100.19.2
04/28-14:36:03.000963  [**] [1:402:7] ICMP Destination Unreachable Port
Unreachable [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.18.10 -> 10.100.19.2
04/28-14:36:03.075795  [**] [1:1228:7] SCAN nmap XMAS [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.100.19.2:61573 -> 10.100.18.10:1
04/28-14:36:03.485648  [**] [1:449:6] ICMP Time-To-Live Exceeded in
Transit [**] [Classification: Misc activity] [Priority: 3] {ICMP}
10.100.19.1 -> 10.100.19.2
```

Podemos ver una gran cantidad de mensajes que nos muestra snort en kali2 (eth0);
Pero a diferencia del caso anterior, como mi ordenador no tiene lanzado ningun servidor, en cuanto al sondeo de puerto solo aparecen mensajes de respuesta con el flag RST activado.