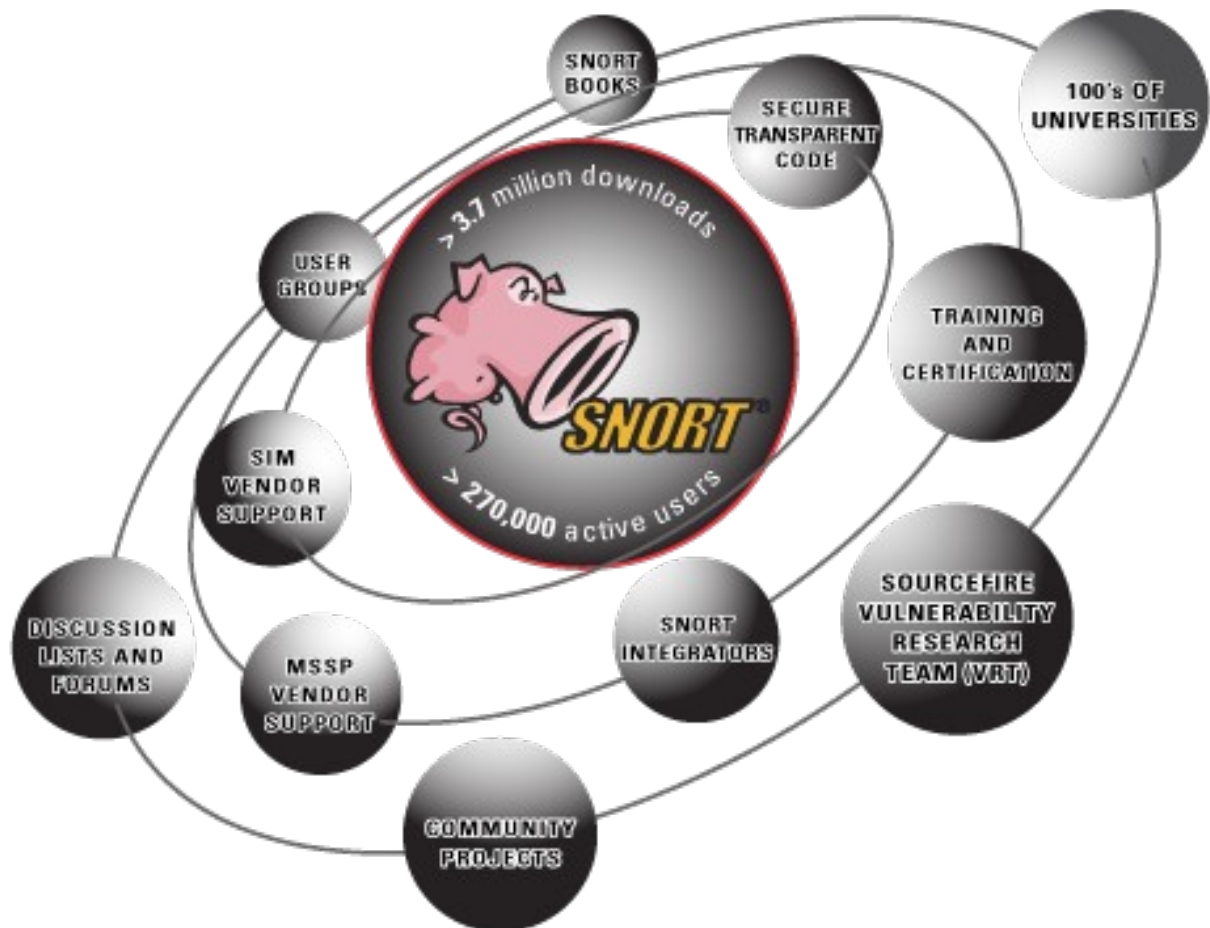


IMPLEMENTACIÓN DE SNORT



ÍNDICE

INTRODUCCIÓN.....	2
Funcionamiento interno de Snort.....	3
INSTALACIÓN.....	4
Montaje de la red.....	4
Instalación de Snort, barnyard2 y Snorby.....	5
Snort.....	6
barnyard2.....	9
PulledPork.....	10
Snorby.....	11
USO Y EJEMPLOS.....	12
Puesta en funcionamiento.....	12
La interfaz de Snorby.....	13
Ataque man in the middle con ettercap.....	18
Escaneo de puertos desde Windows con Zenmap.....	20
Ataque de diccionario a un servidor SSH.....	21
BIBLIOGRAFÍA.....	24

INTRODUCCIÓN

Snort es un sniffer de paquetes y un detector de intrusos basado en red. Fue creado por Martin Roesch en 1998 y es de código abierto, aunque desde 2013 pertenece a Cisco.

Tiene un motor de detección de ataques muy potente, que es totalmente configurable mediante la creación de nuevas reglas. Snort permite mostrar los datos por pantalla o almacenarlos en archivos de texto plano o tipo Unified2, que es un formato estándar para IDS (siglas de *sistema de detección de intrusos* en inglés). Estos datos son procesados por otros programas como **barnyard2**, que se encargan de llevar los registros a una base de datos MySQL. Existen también multitud de aplicaciones de terceros que permiten recoger el contenido de esta base de datos y mostrarlo por pantalla en un formato más agradable, además de proporcionar estadísticas de los datos. Ejemplos de programas de este tipo son **ACID**, **BASE**, **Snorby**, **Sguil** y **Aanval**.

Las capacidades de un IDS como snort son las siguientes:

- Recopilar información sobre ataques que nos hayan realizado.
- Inspeccionar cosas que no deberían estar presentes, como archivos infectados, paquetes que no deberían existir en nuestra red o existencia de patrones extraños.
- Reconocer ataques y contenerlos antes de que se expandan demasiado.
- Reconocer patrones que reflejen ataques conocidos.
- Análisis estadístico de los datos.

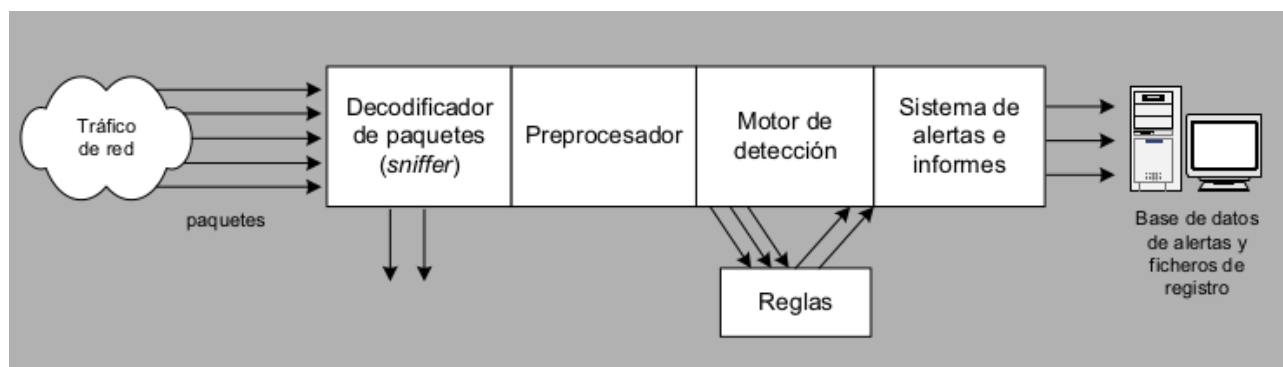
Sin embargo, un IDS por muy bien configurado que esté nunca podrá:

- Compensar sistemas de autenticación débiles en nuestra red.
- Investigar ataques sin la intervención de técnicos de red.

- Compensar por debilidades en protocolos de red o servicios que tengamos mal configurados.

Funcionamiento interno de Snort

La arquitectura de Snort se basa en cuatro componentes principales que se muestran en la siguiente imagen:

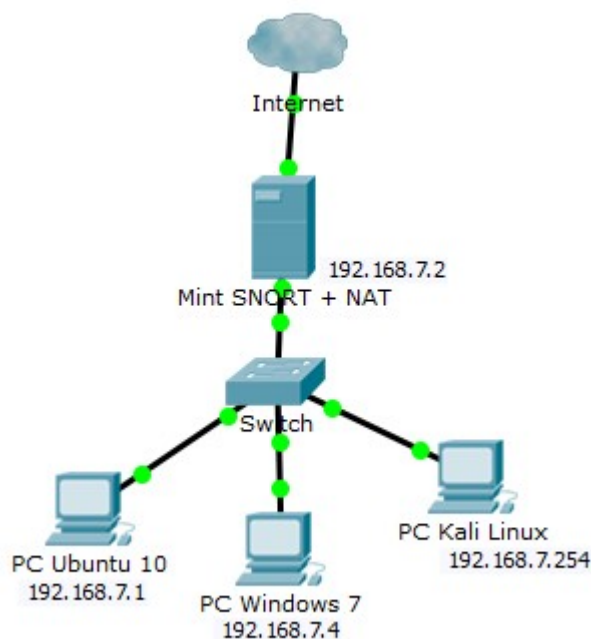


- **Decodificador de paquetes** o **sniffer**: recoge los paquetes de red y los prepara para ser procesados.
- **Preprocesadores**: preparan los datos antes de que actúe el motor de detección. Algunos también generan alertas si detectan anomalías en las cabeceras de los paquetes.
- **Motor de detección**: compara los paquetes con las **reglas** y si encuentra una coincidencia, genera una alerta. Las reglas pueden ser definidas fácilmente por el usuario. Esto es uno de los mayores éxitos de Snort y lo que provoca que tenga una cantidad tan grande de usuarios. Gracias a la comunidad, Snort tiene un sistema de reglas más potente que cualquier IDS de pago (en los que, como es lógico, los usuarios no pueden ver las reglas), y además están más actualizadas.
- Si se detecta una coincidencia con una regla, Snort lleva a cabo las acciones que le hayamos indicado. Para esta práctica queremos que almacene la información en archivos de formato Unified2.

INSTALACIÓN

Montaje de la red

Nuestra red tendrá el aspecto siguiente:



Instalaremos Snort en una máquina Linux Mint que además hará de router NAT para los ordenadores de la red interna. Lo configuraremos para que analice solamente el tráfico de la interfaz de la red interna, que en nuestro caso es eth1. Si se tuviera Snort en una máquina diferente, habría que conectarlo a un puerto espejo del switch o poner un hub antes del switch, para que capte todo el tráfico. También se puede poner el IDS fuera del router/firewall, pero es menos recomendable ya que de esta forma daría muchas alertas de tráfico que en realidad no entra en nuestra red interna.

Utilizaremos una máquina Kali Linux para realizar diversos ataques sobre el resto de ordenadores de la red y comprobar la efectividad de Snort.

Instalación de Snort, barnyard2 y Snorby

Vamos a instalar y configurar Snort para que actúe como sistema Sistema de Detección de Intrusos, aunque también puede actuar como sniffer. Nuestra instalación se basa en varios componentes que funcionan a la vez:

- **Snort**, versión 2.9.8: el IDS que analizará el tráfico de nuestra red. Lo configuraremos para que la salida de las alertas sea en formato Unified2. Necesita de una biblioteca llamada DAQ (*Data Acquisition Library*), que nos descargaremos de la misma página de Snort.

<https://www.snort.org/downloads>

- **barnyard2**, versión 2.1.14: es el programa que se encargará de llevar estos registros en formato Unified2 a una base de datos MySQL. URL del proyecto:

<https://github.com/firnsy/barnyard2>

- **PulledPork**, versión 0.7.2: herramienta que nos permitirá descargar automáticamente las últimas reglas creadas por la comunidad para Snort.

<https://github.com/finchy/pulledpork>

- **Snorby**, versión 2.6.2: una de las interfaces gráficas web disponibles para Snort. Recogerá los informes de la base de datos y los presentará de manera legible y ordenada.

<https://github.com/Snorby/snorby>

He realizado la instalación siguiendo paso a paso [esta excelente guía](#) de Noah Dietrich, escrita originalmente para Ubuntu pero que me ha funcionado en Linux Mint. A continuación voy a dar una idea general de la instalación y explicar los pasos más importantes:

Snort

- Nos bajamos tanto Snort 2.9.8 como el DAQ, en su versión 2.0.6, de la página web de Snort. Hay que descomprimirlos, compilarlos (*./configure, make*) e instalarlos (*sudo make install*). Instalamos las dependencias necesarias que se indican en la guía. El archivo binario del programa es **/usr/local/bin/snort** pero la carpeta donde estará la configuración y todas las reglas será **/etc/snort**. Los logs y los archivos donde se guarda el output de snort están en **/var/log/snort**.
- Hay que crear el usuario snort en el sistema, así como toda la estructura de directorios de **/etc/snort** y **/var/log/snort**, y asignar los permisos adecuados. Así es como queda el directorio **/etc/snort**, podemos ver dónde se van a guardar las distintas reglas:

```
user@snortserver:~$ tree /etc/snort
/etc/snort
|-- attribute_table.dtd
|-- classification.config
|-- file_magic.conf
|-- gen-msg.map
|-- preproc_rules
|-- reference.config
|-- rules
|   |-- iplists
|   |   |-- black_list.rules
|   |   |-- white_list.rules
|   |-- local.rules
|-- snort.conf
|-- so_rules
|-- threshold.conf
|-- unicode.map
```

En nuestro caso, como utilizaremos PulledPork, todas las reglas se guardarán en un único archivo, **/etc/snort/snort.rules**, que no se muestra en la imagen.

- Editamos el archivo de configuración de Snort, **/etc/snort/snort.conf**, para indicarle cuál es nuestra red interna y dónde estarán las reglas.

- Comprobamos que funciona ejecutándolo con la opción -v, que nos indica la versión

```
root@debian:~/snort_src/snort-2.9.8.0# snort -V

o''~
  ''~
    ~

-*> Snort! <*-
Version 2.9.8.0 GRE (Build 229)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2015 Cisco and/or its affiliates. All rights reserved.

Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.6.2
Using PCRE version: 8.35 2014-04-04
Using ZLIB version: 1.2.8

root@debian:~/snort_src/snort-2.9.8.0# _
```

Nota: las capturas corresponden a una instalación previa en Debian 8 que no me terminó funcionando. De la instalación en Mint no hice capturas.

- Escribimos una regla simple para comprobar la detección. Añadimos a **/etc/snort/rules/local.rules** la siguiente línea

```
alert icmp any any -> $ HOME_NET any (msg:"ICMP test detected"; GID:1;
sid:10000001; rev:001; classtype:icmp-event;)
```

Es fácil entender lo que hace esta regla: nos lanza un mensaje cuando enviamos un ping a algún ordenador de la red interna. 10000001 es el identificador único de la regla. Se pone un número tan elevado para que no interfiera con las demás reglas que vamos a instalar.

La siguiente imagen muestra la estructura básica de una regla en Snort:

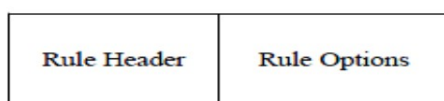


Figure 3. Snort Rule Structure

Action	Protocol	Source Address	Source Port	Direction	Destination Address	Destination Port
--------	----------	----------------	-------------	-----------	---------------------	------------------

Figure 4. Rule Header Format

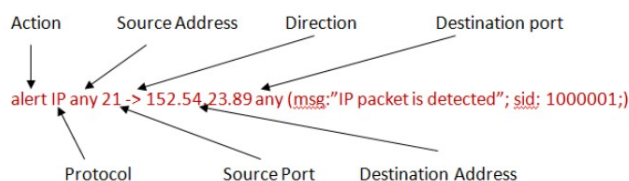
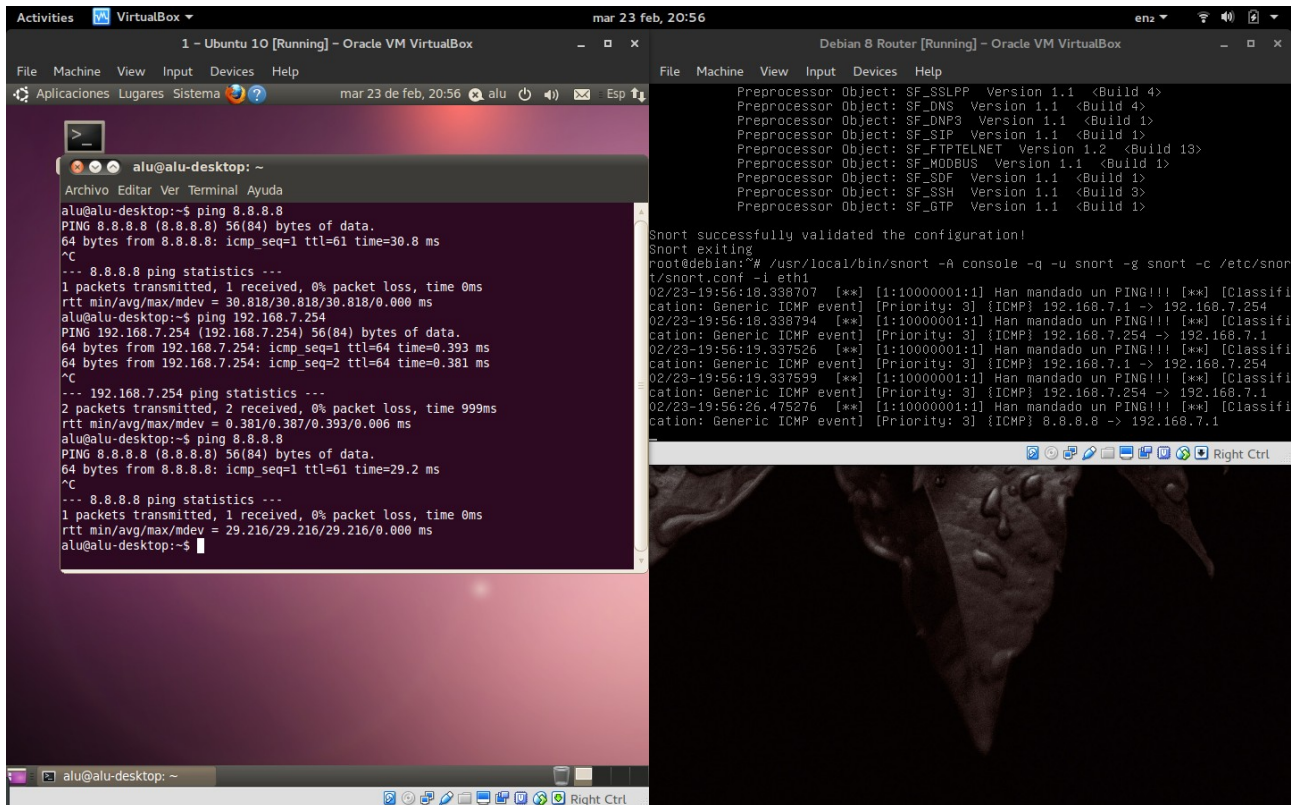


Figure 5. A Snort rule example.

Implementación de Snort

Para comprobar que la regla funciona, iniciamos Snort en modo IDS con la opción *-A console*, que muestra los resultados por pantalla, y hacemos que se ejecute con el usuario *snort* y el grupo *snort*:



```
alu@alu-desktop: ~  
Archivo Editor Ver Terminal Ayuda  
alu@alu-desktop:~$ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=61 time=30.8 ms  
^C  
--- 8.8.8.8 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 30.818/30.818/30.818/0.000 ms  
alu@alu-desktop:~$ ping 192.168.7.254  
PING 192.168.7.254 (192.168.7.254) 56(84) bytes of data:  
64 bytes from 192.168.7.254: icmp_seq=1 ttl=64 time=0.393 ms  
64 bytes from 192.168.7.254: icmp_seq=2 ttl=64 time=0.381 ms  
^C  
--- 192.168.7.254 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 999ms  
rtt min/avg/max/mdev = 0.381/0.387/0.393/0.006 ms  
alu@alu-desktop:~$ ping 8.8.8.8  
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:  
64 bytes from 8.8.8.8: icmp_seq=1 ttl=61 time=29.2 ms  
^C  
--- 8.8.8.8 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 29.216/29.216/29.216/0.000 ms  
alu@alu-desktop:~$
```

```
Debian 8 Router [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>  
Preprocessor Object: SF_DNS Version 1.1 <Build 4>  
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>  
Preprocessor Object: SF_SIP Version 1.1 <Build 1>  
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>  
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>  
Preprocessor Object: SF_SDF Version 1.1 <Build 1>  
Preprocessor Object: SF_SSH Version 1.1 <Build 3>  
Preprocessor Object: SF_GTP Version 1.1 <Build 1>  
Snort successfully validated the configuration!  
Snort exiting  
root@debian:~# /usr/local/bin/snort -A console -q -u snort -g snort -c /etc/snort.conf -i eth1  
02/23-19:56:18.338707 [**] [1:10000001:1] Han mandado un PING!!! [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 192.168.7.1 -> 192.168.7.254  
02/23-19:56:18.338794 [**] [1:10000001:1] Han mandado un PING!!! [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 192.168.7.254 -> 192.168.7.1  
02/23-19:56:19.337526 [**] [1:10000001:1] Han mandado un PING!!! [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 192.168.7.1 -> 192.168.7.254  
02/23-19:56:19.337599 [**] [1:10000001:1] Han mandado un PING!!! [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 192.168.7.254 -> 192.168.7.1  
02/23-19:56:26.475276 [**] [1:10000001:1] Han mandado un PING!!! [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 8.8.8.8 -> 192.168.7.1
```

- Para probar que una configuración de Snort es válida, lo ejecutamos con la opción *-T*, y le indicamos con *-i* la interfaz por la que va a escuchar y con *-c* el archivo de configuración:

```
sudo snort -T -i eth1 -c /etc/snort/snort.conf
```

- Al crear una regla local, añadiremos siempre una línea en */etc/snort/sid-msg.map*, para que barnyard2 se entere de que existe y la cargue. Este archivo tiene un formato específico y la línea a añadir en este caso sería así:

```
1 || 10000001 || 001 || icmp-event || 0 || ICMP Test detected ||  
url,tools.ietf.org/html/rfc792
```

barnyard2

- Nos bajamos la versión 2.1.14-336 de la página del proyecto, y la descomprimos, compilamos e instalamos. Instalamos dependencias.
- En este momento crearemos el usuario *snort* y la base de datos *snort* en MySQL.
- Configuramos snort para que deje las alertas en archivos de formato u2. Configuramos también barnyard2 para que recoja correctamente estos archivos y los guarde en la base de datos que acabamos de crear.
- Comprobamos que todo funciona correctamente arrancando ambos programas, lanzando unos cuantos pings para hacer saltar la regla y comprobando la tabla *events* de la base de datos, que debería tener unas cuantas filas. Podemos ver también los archivos .u2 en `/var/log/snort`.

```
sudo /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w
/var/log/snort/barnyard2.waldo -g snort -u snort
```

```
root@debian:~# mysql -u snort -p -D snort -e "select count(*) from event"
Enter password:
+-----+
| count(*) |
+-----+
|      34 |
+-----+
root@debian:~# _
```

```
root@debian:~# ls -l /var/log/snort
total 12
drwsrwxr-t 2 snort snort 4096 Feb 23 19:01 archived_logs
-rw-r--r-- 1 snort snort   0 Feb 23 20:44 barnyard2.waldo
-rw----- 1 snort snort  594 Feb 23 19:56 snort.log.1456257307
-rw----- 1 snort snort 1940 Feb 23 20:56 snort.u2.1456260970
root@debian:~# _
```


Este es el aspecto que tienen las reglas editadas por la comunidad:

```
# -- Begin GID:3 Based Rules -- #

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"FILE-OFFICE Microsoft Word ole stream memory corruption attempt"; sid:13469; gid:3; rev:11; classtype:attempted-user; flowbits:isset,file.doc; reference:cve,2008-0109; reference:url,technet.microsoft.com/en-us/security/bulletin/ms08-009; metadata: engine shared, soid 3|13469, service http, policy max-detect-ips drop;)
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"FILE-OFFICE Microsoft Excel sst record arbitrary code execution attempt"; sid:13582; gid:3; rev:13; classtype:attempted-user; flowbits:isset,file.xls; reference:cve,2008-0116; reference:url,technet.microsoft.com/en-us/security/bulletin/MS08-014; metadata: engine shared, soid 3|13582, service http, policy max-detect-ips drop;)
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"FILE-OFFICE Microsoft Word malformed css remote code execution attempt"; sid:13790; gid:3; rev:11; classtype:attempted-user; flowbits:isset,file.doc; reference:cve,2008-1434; reference:url,technet.microsoft.com/en-us/security/bulletin/MS08-026; metadata: engine shared, soid 3|13790, service http, policy max-detect-ips drop;)
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"FILE-OFFICE RTF control word overflow attempt"; sid:13803; gid:3; rev:11; classtype:attempted-user; flowbits:isset,file.rtf; reference:cve,2008-1091; reference:url,technet.microsoft.com/en-us/security/bulletin/ms08-026; metadata: engine shared, soid 3|13803, service http;)
root@debian:~# cat /etc/snort/rules/snort.rules | wc -l
31930
root@debian:~# _
```

El programa PulledPork se puede añadir al **crontab** para que las reglas se actualicen automáticamente.

Snorby

Sólo queda instalar Snorby y configurarlo para que lea la base de datos. Es una aplicación web con lo que tenemos que tener un servidor Apache funcionando. La instalación se hace sin problemas siguiendo los pasos de la guía, aunque no entiendo algunos pasos ya que nunca habíamos instalado una aplicación del framework Ruby on Rails.

USO Y EJEMPLOS

Puesta en funcionamiento

Nos centraremos en el uso de Snort como IDS. Se pueden iniciar los 3 programas (snort, barnyard2 y el servicio de Snorby que leerá la base de datos, **snorby-worker**) automáticamente al arrancar el sistema, como demonios. Yo elegí no hacerlo porque barnyard2 tarda más de 15 minutos en ponerse en marcha debido a la cantidad de reglas que tiene que cargar, con lo cual me viene bien ver en terminal cuándo ha terminado de arrancar. Los comandos para poner todo en marcha serían:

```
| /usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i eth1
```

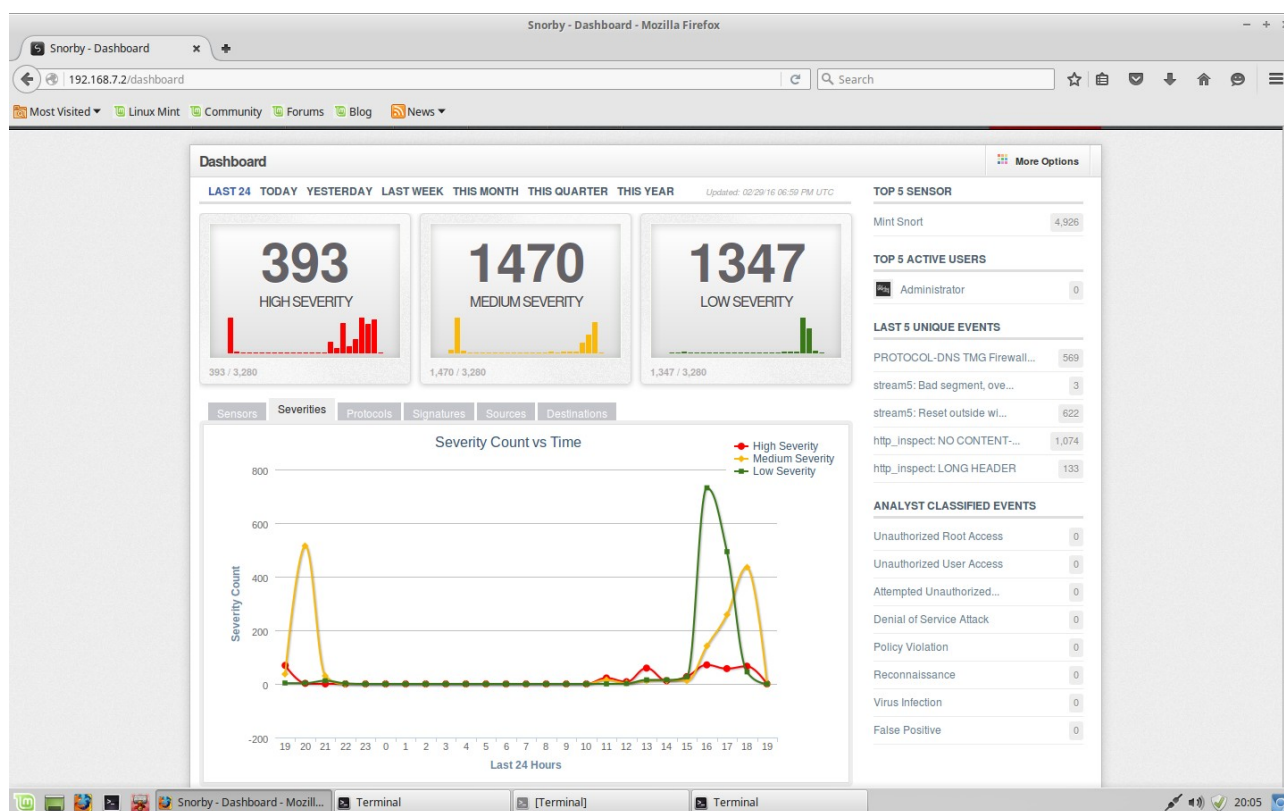
```
| /usr/local/bin/barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f  
snort.u2 -q -w /var/log/snort/barnyard2.waldo -g snort -u snort -a  
/var/log/snort/archived_logs
```

```
| cd /var/www/html/snorby  
/usr/bin/ruby script/delayed_job start
```

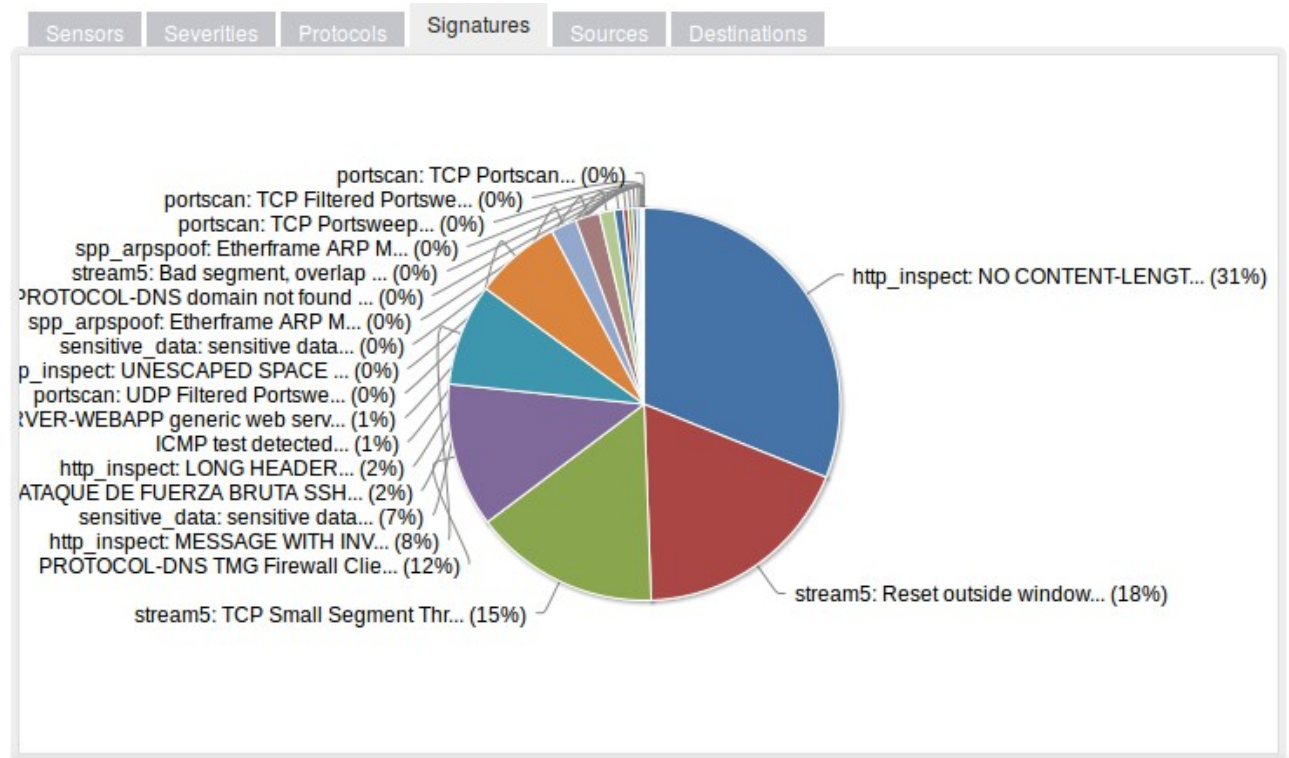
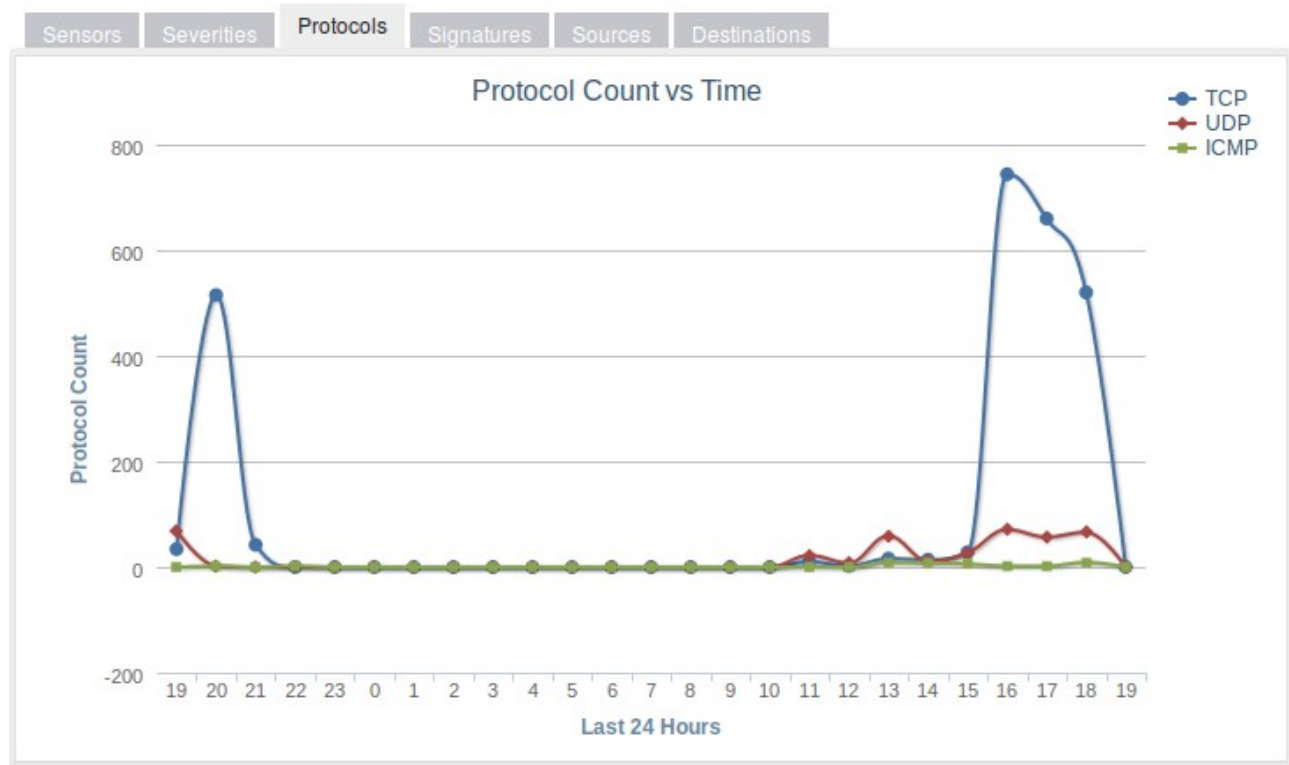
Ejecuto estos comandos en 3 terminales distintas como root, y no con *sudo* porque si no se abren dos procesos (uno siendo ejecutado por *snort* y otro por *alu*) por cada comando, con lo que hay conflictos.

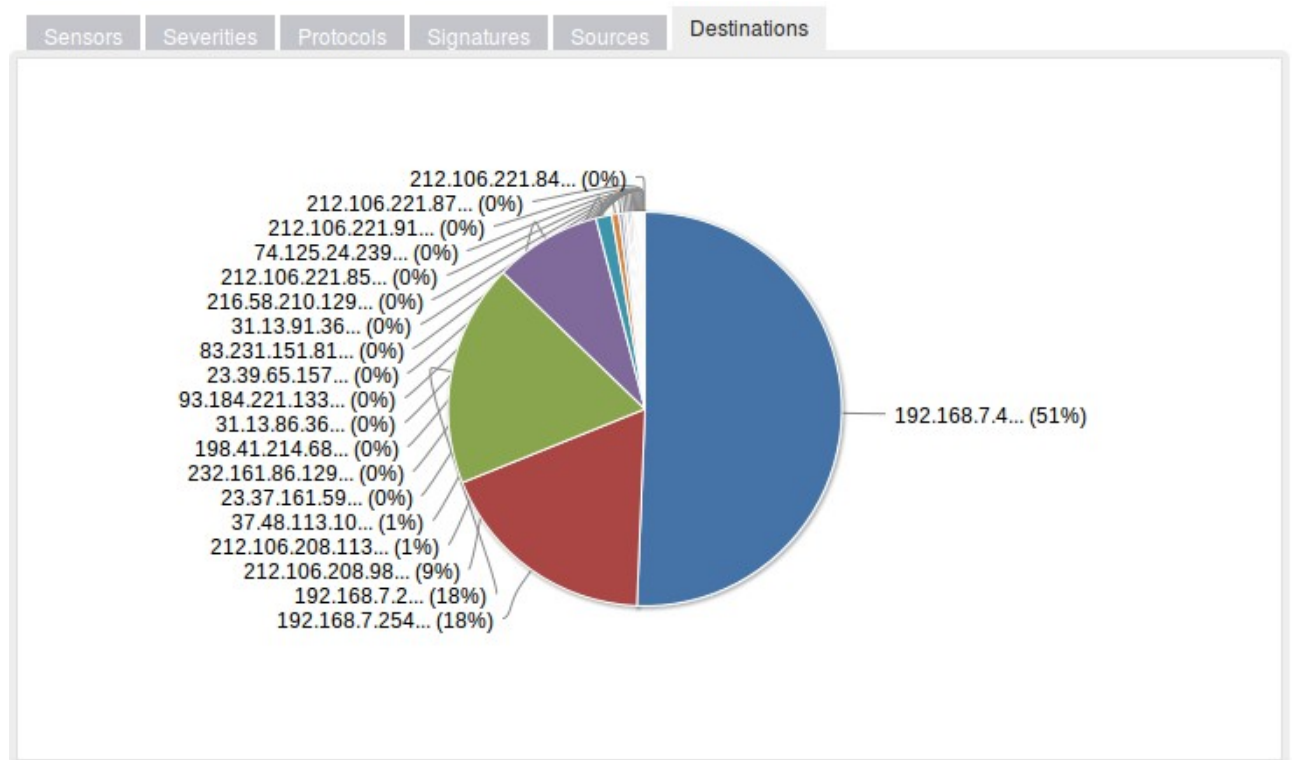
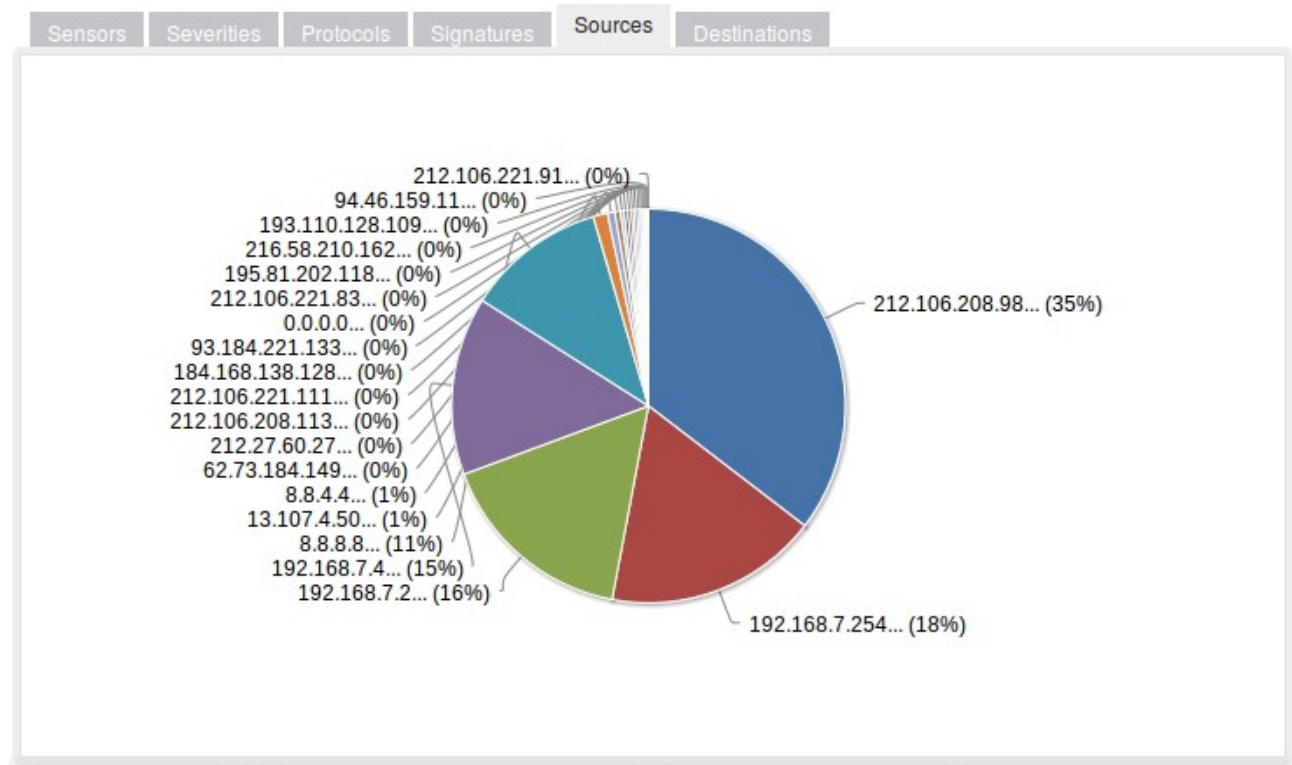
La interfaz de Snorby

Al acceder al servidor web de nuestro Linux Mint aparecerá una página de login, en la que debermos indicar usuario *snorby@snorby.org*, y contraseña *snorby*. Después accederemos a la interfaz principal, en la que podemos ver todas las alertas ordenadas por prioridad. También podemos ver diferentes gráficos con información diversa.



Algunas de las alertas que veremos al principio serán falsos positivos que da Snort. Otra información cambiando las pestañas de abajo:





Otras utilidades que nos proporciona Snorby:

- Ver todos los datos del paquete que ha generado la alerta. Como ejemplo un paquete generado por un escaneo de puertos con Zenmap.

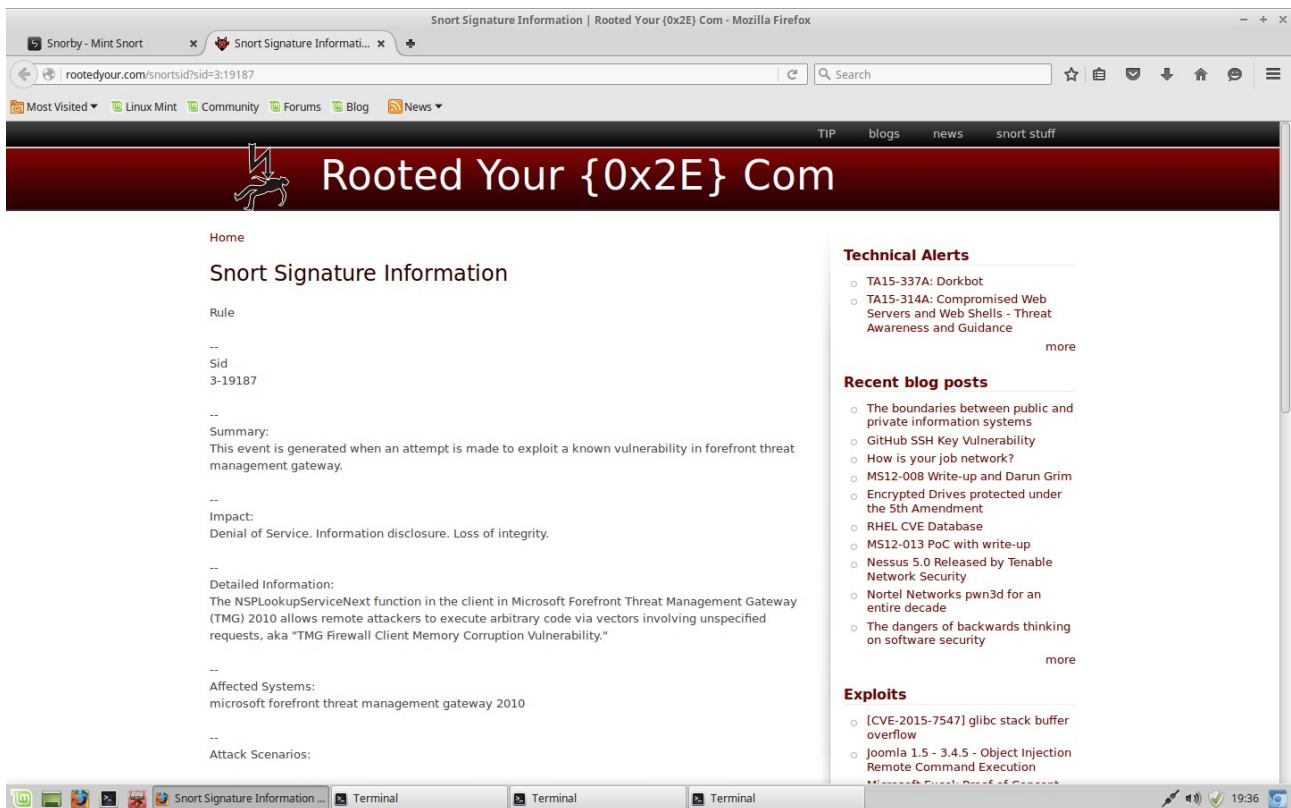
The screenshot shows the Snorby web interface for a specific event. At the top, it displays the event details: 'Mint Snort', source IP '192.168.7.4', destination IP '192.168.7.2', and event signature 'portscan: TCP Portswep'. Below this, the 'IP Header Information' section shows a table with fields like Source, Destination, Ver, Hlen, Tos, Len, ID, Flags, Off, TTL, Proto, and Csum. The 'Signature Information' section shows the Generator ID (122), Sig. ID (3), and Category (attempted-recon). The 'Payload' section displays the raw packet data in hexadecimal and ASCII. The 'Notes' section at the bottom indicates that there are currently no notes for this event.

- Hacer una búsqueda *whois* de la IP que ha provocado la alerta, con un solo click:

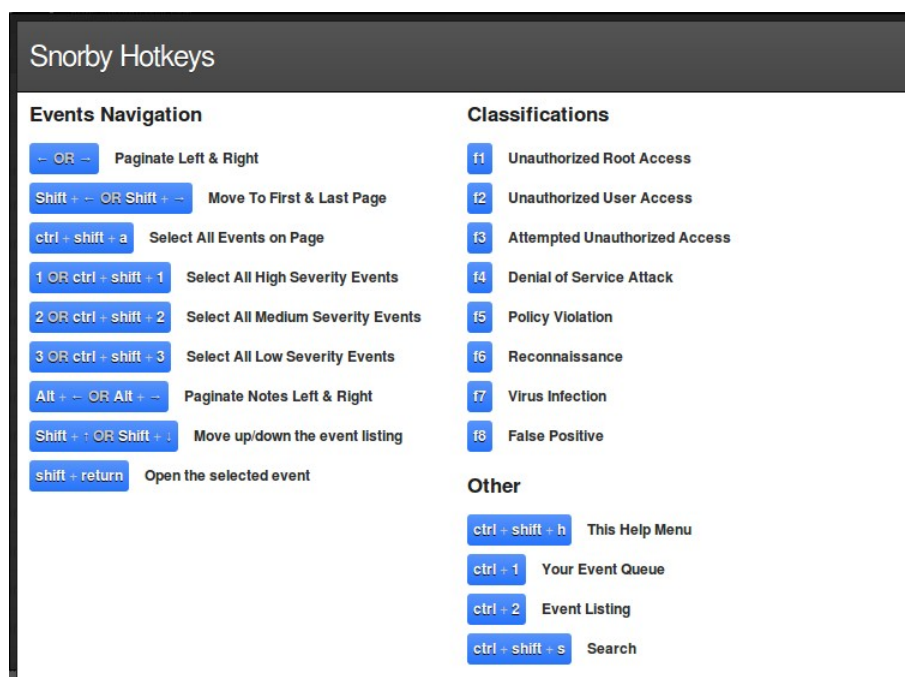
The screenshot shows a Whois lookup result for the IP address 87.221.106.212. The title of the page is 'Lookup: 87.221.106.212.static.jazztel.es'. The main content area displays the Whois information in a text format, including details about the RIPE Database query service, the object's name (JAZZNET), and contact information for Jazz Telecom S.A. The output is filtered to show information related to the IP range 212.106.194.0 - 212.106.223.255. The contact information includes the role of JAZZTEL-RIPE, the address of Jazz Telecom S.A. in Madrid, and the email address abuse@jazztel.com.

Implementación de Snort

- Consultar la base de datos oficial para una alerta de Snort, en un solo click



- Teclas rápidas, para un acceso inmediato a la información



Ataque man in the middle con ettercap

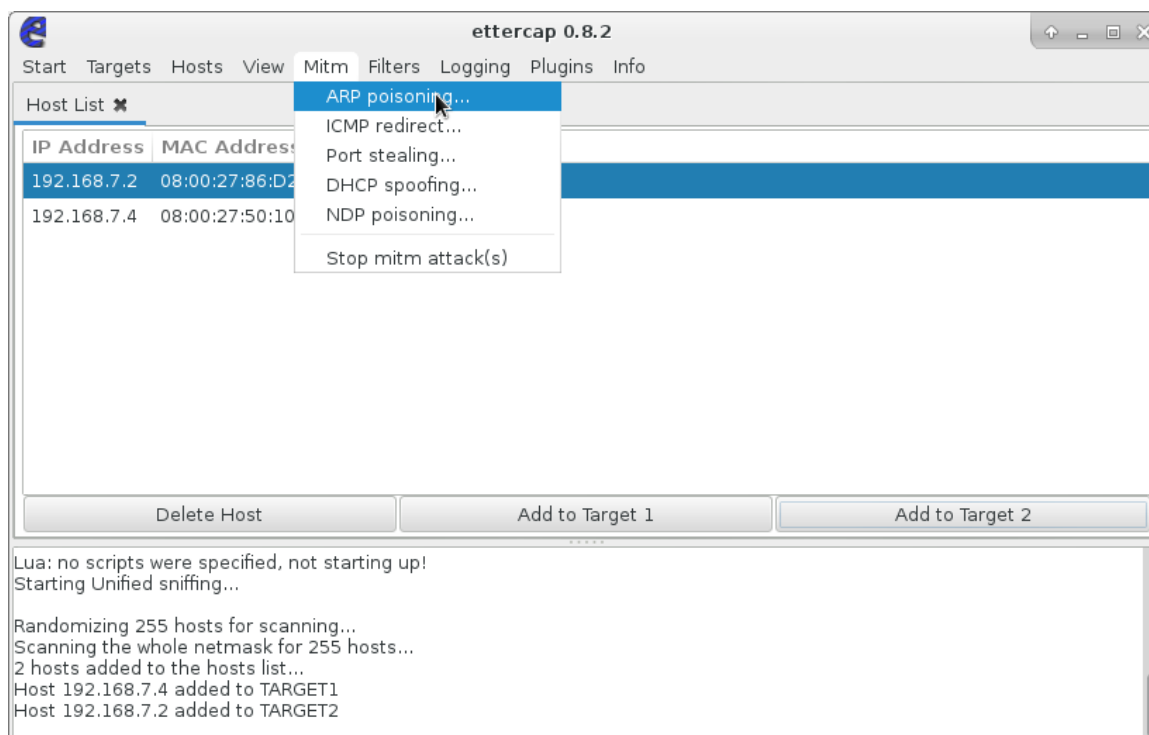
Para que Snort detecte este tipo de ataques debemos activar el preprocesador **arpspoof**.

```
# ARP spoof detection. For more information, see the Snort Manual - Configuring Snort -  
Preprocessors - ARP Spoof Preprocessor  
preprocessor arpspoof  
# preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
```

Descomentando la línea '*preprocessor arpspoof*', snort revisará todos los paquetes para inconsistencias entre direcciones IP y MAC. Para más información acerca de la configuración de este preprocesador:

<http://manual.snort.org/node17.html#SECTION00321500000000000000>

Vamos a realizar un ataque de ARP spoofing con nuestra máquina con Kali Linux. Para ello utilizaremos **ettercap**. Realizar un ataque de este tipo es muy simple, basta escanear la red interna con el programa, seleccionar nuestros objetivos y seleccionar '*ARP poisoning...*'



Podemos comprobar que se ha realizado con éxito porque en la tabla ARP de la víctima aparece tanto la máquina atacante como el router con la misma MAC (la del atacante). En el router ocurriría lo mismo. De esta manera todo el tráfico entre el router y la víctima pasa a través de la máquina atacante.

```
C:\Users\alu>arp -a

Interfaz: 192.168.7.4 --- 0xa
Dirección de Internet      Dirección física      Tipo
192.168.7.2                08-00-27-4c-8e-d4     dinámico
192.168.7.254              08-00-27-4c-8e-d4     dinámico
192.168.7.255              ff-ff-ff-ff-ff-ff     estático
224.0.0.22                  01-00-5e-00-00-16     estático
224.0.0.252                 01-00-5e-00-00-fc     estático
239.255.255.250            01-00-5e-7f-ff-fa     estático

C:\Users\alu>
```

Una vez realizado el ataque, podemos ver cómo snort nos avisa de una discordancia en la capa Ethernet:

```
02/29-15:58:07.412524  [**] [112:2:1] spp_arpspoof: Etherframe ARP Mismatch SRC [**]
02/29-15:58:08.426200  [**] [112:2:1] spp_arpspoof: Etherframe ARP Mismatch SRC [**]
02/29-15:58:09.437297  [**] [112:2:1] spp_arpspoof: Etherframe ARP Mismatch SRC [**]
```

En la interfaz de Snorby también aparecen aunque no clasifica la alerta en ningún nivel de severidad, debido a que la regla no debe estar correctamente escrita para ello:



Escaneo de puertos desde Windows con Zenmap

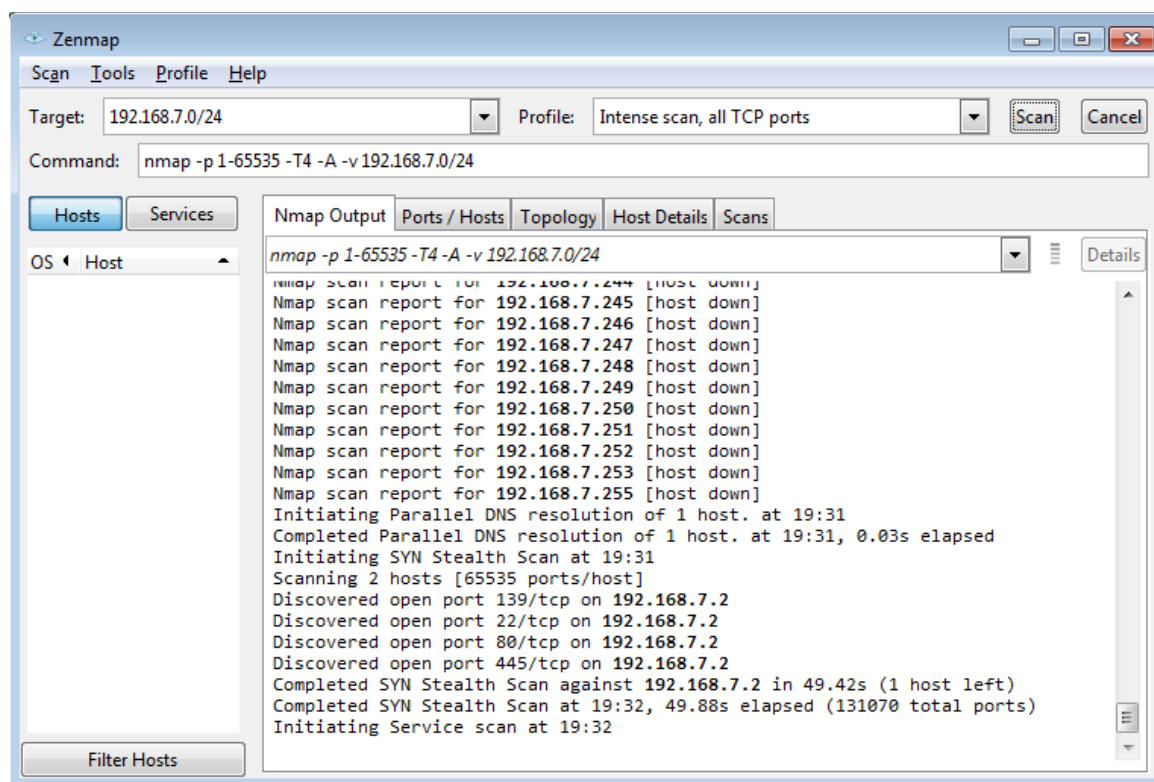
De nuevo hay que activar un preprocesador que venía desactivado por defecto, en este caso es el **sfportscan**. Este preprocesador está desarrollado por Sourcefire y su misión es detectar si estamos siendo víctimas de cualquier procedimiento de reconocimiento de nuestra red interna (es el primer paso en cualquier ataque a través de la red).

```
# Portscan detection. For more information, see README.sfportscan
# preprocessor sfportscan: proto { all } memcap { 10000000 } sense_level { low }
preprocessor sfportscan: proto { all } scan_type { all } sense_level { high }
```

La que vemos en la imagen es una configuración muy básica, para más información se puede consultar el manual:

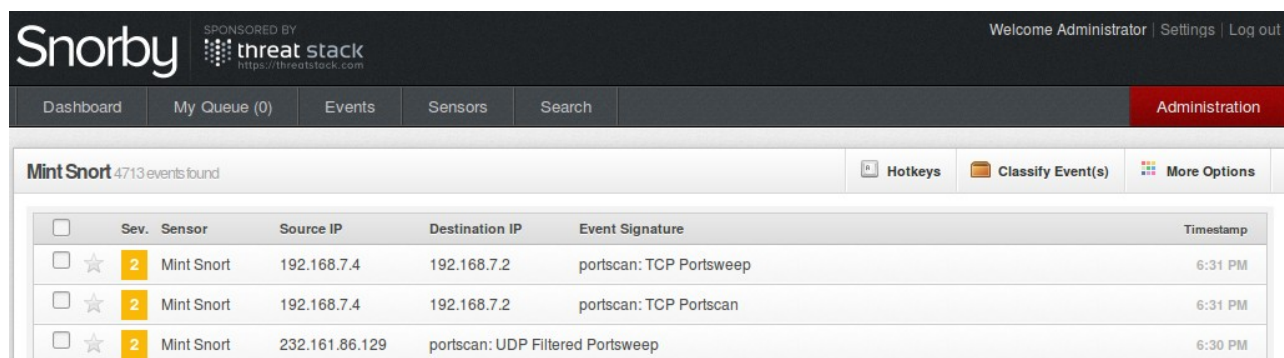
<http://manual.snort.org/node17.html#SECTION00324000000000000000>

Descargamos **Zenmap** en nuestra máquina Windows 7 y realizamos un ataque de escaneo intensivo de puertos al ordenador con Linux Mint en el que tenemos Snort.



Vemos las alertas reflejadas inmediatamente en Snort, tanto en la terminal como en Snorby:

```
02/29-19:30:39.386455 [**] [122:23:1] portscan: UDP Filtered Portsweep [**] [Classification: Attempted Information Leak] [Priority: 2] {PROTO:128} 0.0.0.0 -> 232.161.86.129
02/29-19:31:37.803206 [**] [122:3:1] portscan: TCP Portsweep [**] [Classification: Attempted Information Leak] [Priority: 2] {PROTO:255} 192.168.7.4 -> 192.168.7.2
02/29-19:31:37.825555 [**] [122:1:1] portscan: TCP Portscan [**] [Classification: Attempted Information Leak] [Priority: 2] {PROTO:255} 192.168.7.4 -> 192.168.7.2
```



The screenshot shows the Snorby web interface. At the top, there's a navigation bar with 'Dashboard', 'My Queue (0)', 'Events', 'Sensors', 'Search', and 'Administration'. Below this, a table displays events found by 'Mint Snort'. The table has columns for 'Sev.', 'Sensor', 'Source IP', 'Destination IP', 'Event Signature', and 'Timestamp'. Three events are listed, all with a severity of 2 and sensor 'Mint Snort'. The first two events are 'portscan: TCP Portsweep' and 'portscan: TCP Portscan', both from 192.168.7.4 to 192.168.7.2 at 6:31 PM. The third event is 'portscan: UDP Filtered Portsweep' from 232.161.86.129 at 6:30 PM.

Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
2	Mint Snort	192.168.7.4	192.168.7.2	portscan: TCP Portsweep	6:31 PM
2	Mint Snort	192.168.7.4	192.168.7.2	portscan: TCP Portscan	6:31 PM
2	Mint Snort	232.161.86.129		portscan: UDP Filtered Portsweep	6:30 PM

Ataque de diccionario a un servidor SSH

Para este ataque instalamos el paquete *openssh-server* en la máquina Linux Mint, con lo cual tenemos un servidor SSH funcionando con la configuración básica. Cambiamos esta configuración para que permita acceder al usuario *root* con contraseña *123*.

Snort tiene un preprocesador que solamente controla el tráfico SSH y detecta diferentes vulnerabilidades conocidas. Sin embargo esta vez vamos a escribir una regla personalizada. Editaremos el archivo `/etc/snort/rules/local.rules` para añadir la siguiente regla:

```
alert tcp any any -> 192.168.7.2 22 (msg:"ATAQUE DE FUERZA BRUTA SSH";
    flow:established,to_server; content:"SSH"; nocase; offset:0; depth:4;
    detection_filter:track by_src, count 30, seconds 60; sid:100000002; rev:1;)
```

Esto hará saltar una alarma cuando intentemos acceder más de 30 veces en 60 segundos a nuestro servidor SSH (puerto 22 del ordenador 192.168.7.2).

No nos tenemos que olvidar de actualizar el archivo `/etc/snort/sid-msg.map`, ya que si no barnyard2 no se enterará de que existe esta nueva regla:

Implementación de Snort

The screenshot shows the Snorby web interface in a Mozilla Firefox browser. The address bar displays the URL: `192.168.7.2/results?match_all=true&search[sensor][column]=signature&search[sensor][operator]=is&search[sensor][value]=28010&title=`. The page title is "Snorby - ATAQUE DE FUERZA BRUTA SSH". The interface includes a navigation bar with links to Dashboard, My Queue (0), Events, Sensors, Search, and Administration. The main content area displays a table of events titled "ATAQUE DE FUERZA BRUTA SSH" with 70 events bound. The table has columns for checkboxes, severity, sensor, source IP, destination IP, event signature, and timestamp. All events show a severity of 0, sensor "Mint Snort", source IP "192.168.7.254", destination IP "192.168.7.2", and event signature "ATAQUE DE FUERZA BRUTA SSH". The timestamp for all events is "6:27 PM".

<input type="checkbox"/>	Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM
<input type="checkbox"/>	0	Mint Snort	192.168.7.254	192.168.7.2	ATAQUE DE FUERZA BRUTA SSH	6:27 PM

BIBLIOGRAFÍA

- [Guía de instalación](#)
- [Manual de usuario oficial de Snort](#)
- <https://www.snort.org/documents>

Otros tutoriales e información útil:

<http://www.seren.net/documentation/unix%20utilities/Snort.pdf>

<http://resources.infosecinstitute.com/snort-rule-writing-for-the-it-professional/>

En español:

<https://seguridadinformaticaufps.wikispaces.com/file/view/1150214.pdf>

<http://www.maestrosdelweb.com/snort/>

Vídeos:

<https://www.youtube.com/watch?v=RUmYojxy3Xw>

https://www.youtube.com/watch?v=cQeeko9J_Yw