Prueba de Concepto ELK + Snort

Seguridad en Redes

Docente: Ing. Gonzalo Vilanova

Alumna: Flavia De Rosa

Legajo: 158.739-0

Comisión: K4571

Objetivos

- Dar conceptos básicos de sistemas de detección de intrusos
- Dar nociones de ELK teniendo en cuenta eventos y logs
- Demostrar la simplicidad en la creación de distintos tipos de dashboard.
- Realizar una demo que permita unificar los conceptos presentados.
- Realizar configuraciones y creaciones de índices adecuadas que permitan captura y visualización de tráfico malicioso por medio de alertas de snort.

Introducción

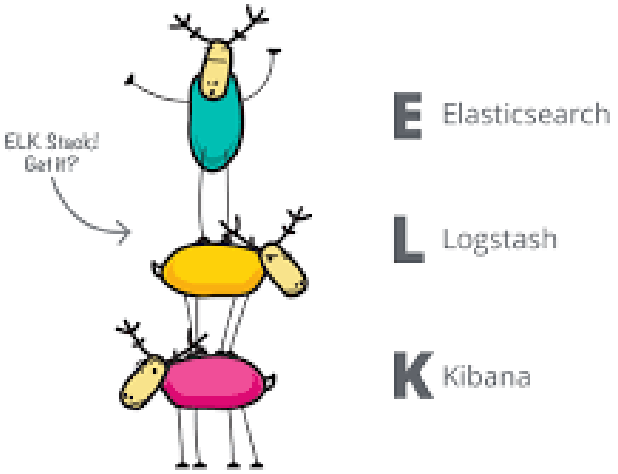
En la actualidad, conocer el comportamiento del usuario al usar las distintas aplicaciones, es información muy útil. Por medio de los reportes de fallas del sistema y su rendimiento, se puede mejorar mucho la calidad de los servicios utilizados.
Este trabajo detalla el desarrollo de un generador de alertas Snort y la configuración de un Elasticsearch, Logstash y Kibana (**SELK**) para analizar, almacenar, visualizar y analizar alertas de Snort por medio de tableros de control desde Kibana
Realizamos una prueba de concepto que consiste en la instalacion y configuracion de los servicios Elasticsearch, Kibana y Logstash (ELK), con el agregado de Snort, un IDS con el cual se crean reglas que por medio de alertas permite la captura de logs locales, los cuales se visualizan desde la consola de la máquina virtual.
También se crea un dashboard personalizado en Kibana con los logs de Logstash y las capturas de las alertas de Snort, con los que se pueden ver los datos en distintas formas configurables desde un dashboard en Kibana.

ELK

ELK funciona como un Sistema distribuido donde los datos se almacenan en diferentes sistemas que colaboran entre sí y muestran los resultados que se demandan en cada momento en una sola petición.

ELK combina Elasticsearch, Logstach y Kibana que se utiliza para proporcionar un enfoque integral en la consolidación, gestión y análisis de registros de las aplicaciones.

Una de las ventajas ELK Stack es que simplifica la búsqueda y el análisis de datos al proporcionar información en tiempo real a partir de los datos de registro.



Permite operar sobre distintos índices al mismo tiempo y así potenciar las búsquedas..
Elasticsearch tiene una poderosa DSL basada en JSON, que permite a los equipos de desarrollo construir consultas complejas y afinarlas para recibir los resultados más precisos de una búsqueda. También proporciona una forma de clasificar y agrupar los resultados.
Elasticsearch permite escalar horizontalmente, y permite extender los recursos y equilibrar la carga entre los nodos de un cluster. Además, registra cualquier cambio

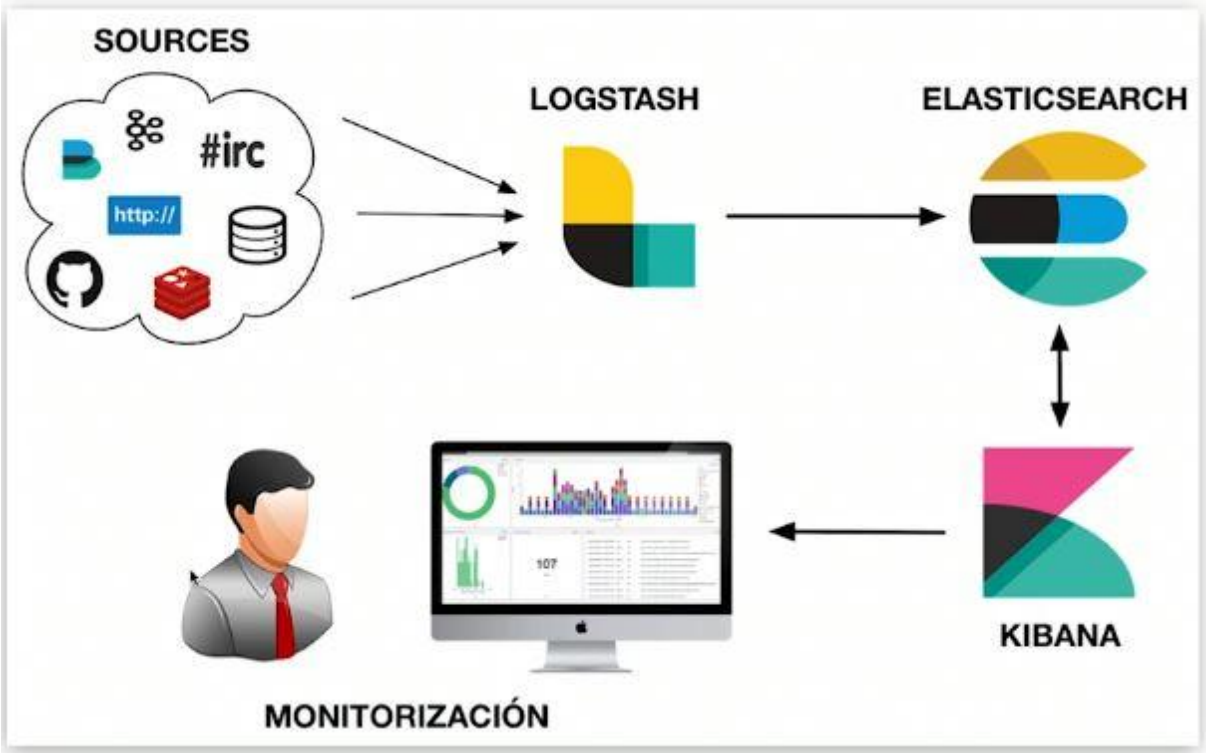
realizado en registros de transacciones en múltiples nodos en el clúster para minimizar la posibilidad de pérdida de datos. Por otro lado, estos cluster pueden detectar aquellos nodos que fallan y reorganizarlos para que los datos siempre sean accesibles.

Funcionamiento

Principalmente ELK Stack satisface una necesidad específica en el espacio de análisis y gestión de registros.

Para infraestructuras basadas en la nube, la consolidación de salidas de registro a una ubicación central desde diferentes fuentes como servidores web, servidores de correo, servidores de bases de datos y dispositivos de red es útil (sobre todo cuando se trata de tomar mejores decisiones basadas en distintas fuentes de datos).

ELK es especialmente bueno para aprovechar al máximo sus registros de Snort.

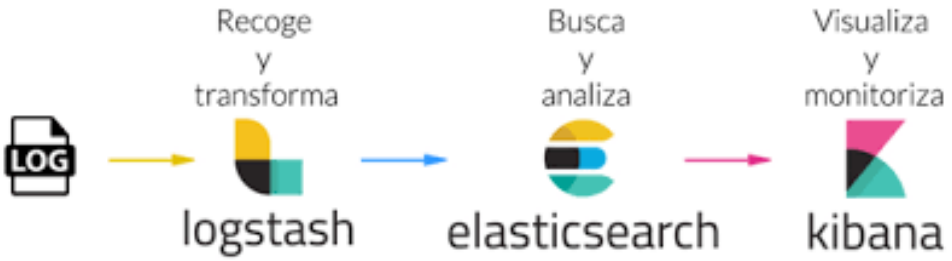


En general, ELK Stack se ejecuta completo, no cada componente individual por separado. **Cada uno de estos servicios desempeña un papel importante y, para desempeñarse bajo una gran demanda, es más ventajoso implementar cada servicio en su propio servidor.**

Pero hacerlo, se introducen otros problemas relacionados con implementaciones multinodos, redes, seguridad y administración.

En la demostración que elaboramos, nos quedamos con el despliegue de un solo stack en el cual están todos los servicios, con la finalidad de que podamos aprender los conceptos básicos de ELK y utilizarlo para fines de desarrollo y pruebas.

Componentes



Elasticsearch:

Es un motor de búsqueda RESTful distribuido —creado para la nube— que almacena los mensajes que se van registrando.

Logstash:

Se encarga de recopilar, procesar y reenviar eventos y registrar los mensajes que va procesando. Estos mensajes los retransmite a ElasticSearch.

Kibana:

Consiste en un panel de búsqueda y análisis de código abierto basado en navegador. Proporciona capacidades de visualización sobre el contenido indexado en un clúster específico de Elasticsearch.

Beats:

Es un agente de cliente el cual se encarga de empujar (hacer un setup) de cualquier archivo nuevo o modificación de la fuente de datos.

Instalación

Se puede realizar en cualquier sistema operativo, teniendo en cuenta que debe estar instalada la última versión de jdk de java. Esto último, hace necesario que la máquina disponga por lo menos de 4GB de memoria RAM, para el correcto funcionamiento del resto de los servicios.

Para la prueba de concepto se cuenta con un ambiente virtual Lubuntu 18.04 LTS en la plataforma VirtualBox. Se le otorga una RAM de 4GB, cumpliendo con los requisitos mínimos para evitar problemas de funcionamiento.

Se elaboró un documento para la instalación (**Instalación y Configuración ELK + Snort.pdf**), donde se explica el paso a paso de la instalación de cada uno de los servicios.

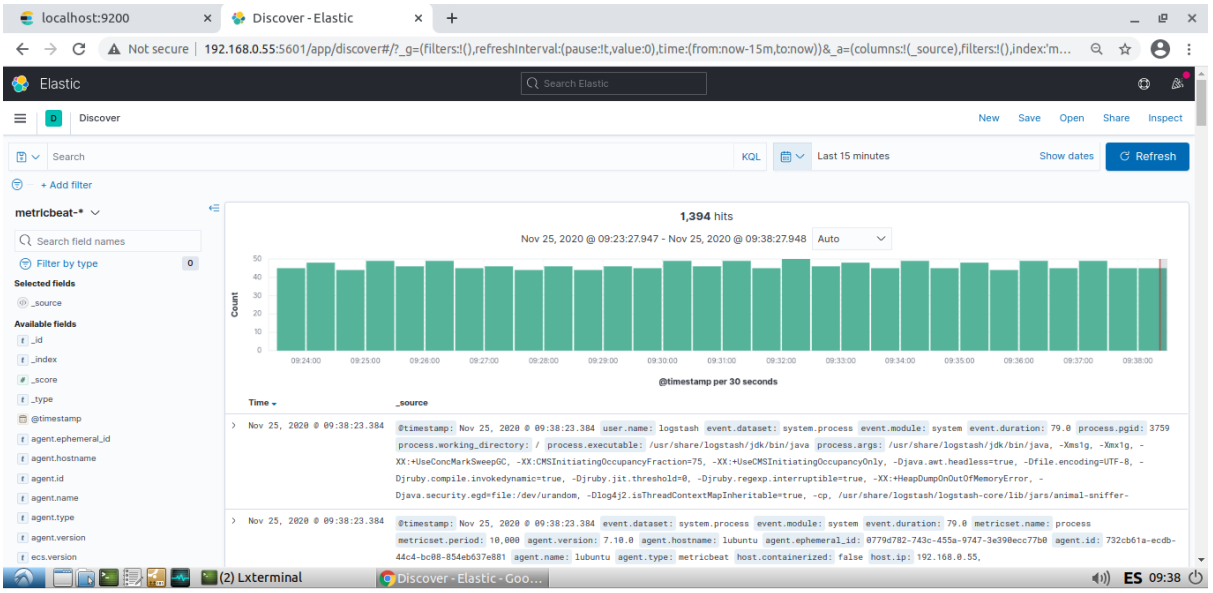
Inicio de Servicios ELK

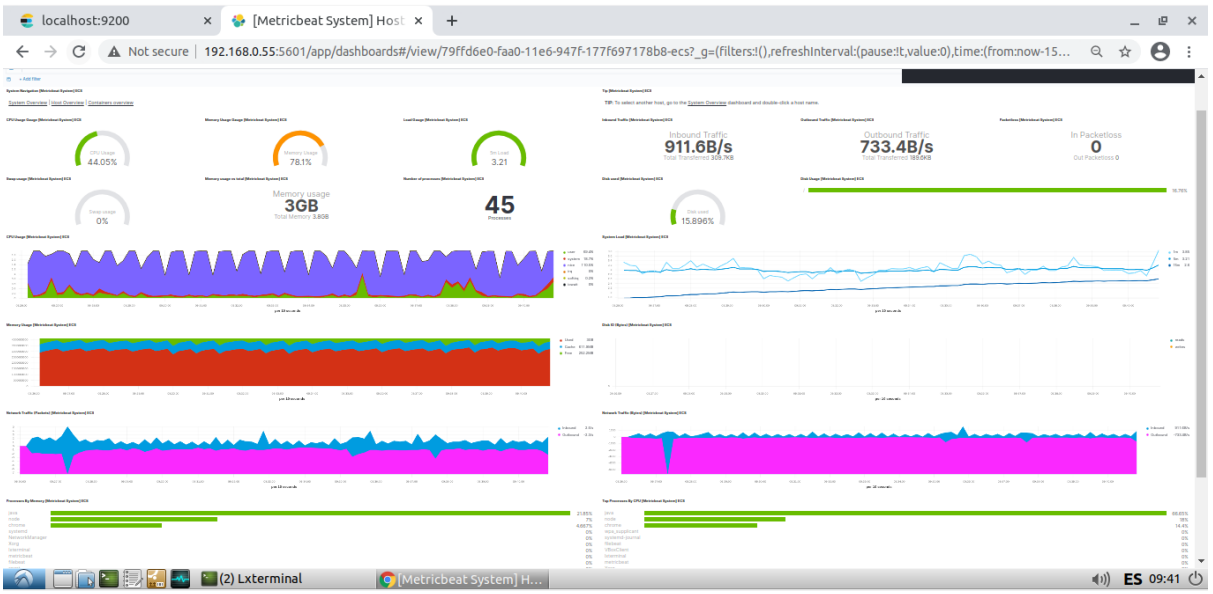
Desde la terminal inicio de a uno los servicios

- systemctl start elasticsearch
- systemctl start kibana
- systemctl start logstash

Verifico por web que se encuentren funcionando y se ejecuta por terminal **bash comprobar.sh** (un script generado para ver el status de cada servicio).

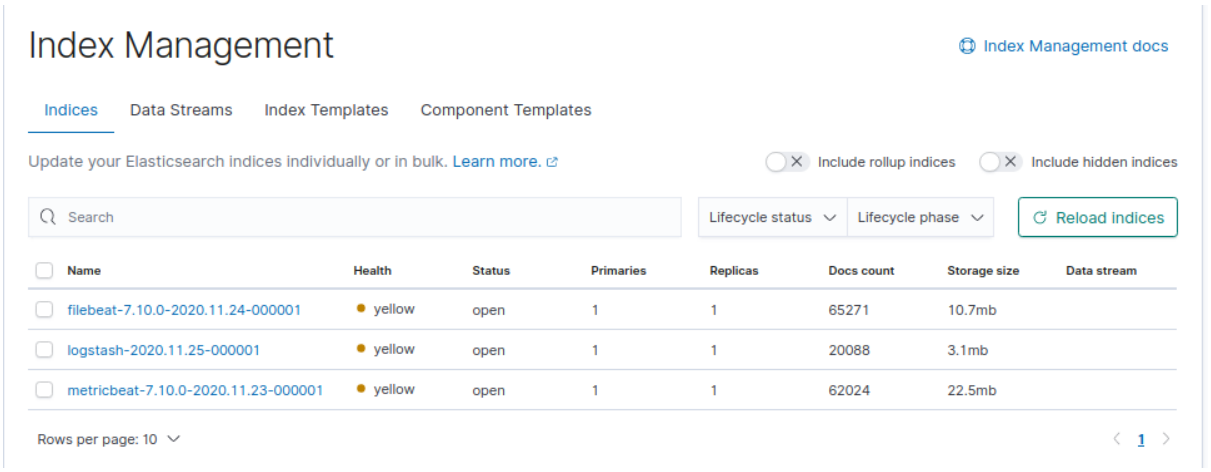
Además **se instaló metricbeat para probar el funcionamiento de elasticsearch y kibana** con las visualizaciones y dashboard para ver las métricas del sistema en tiempo real.



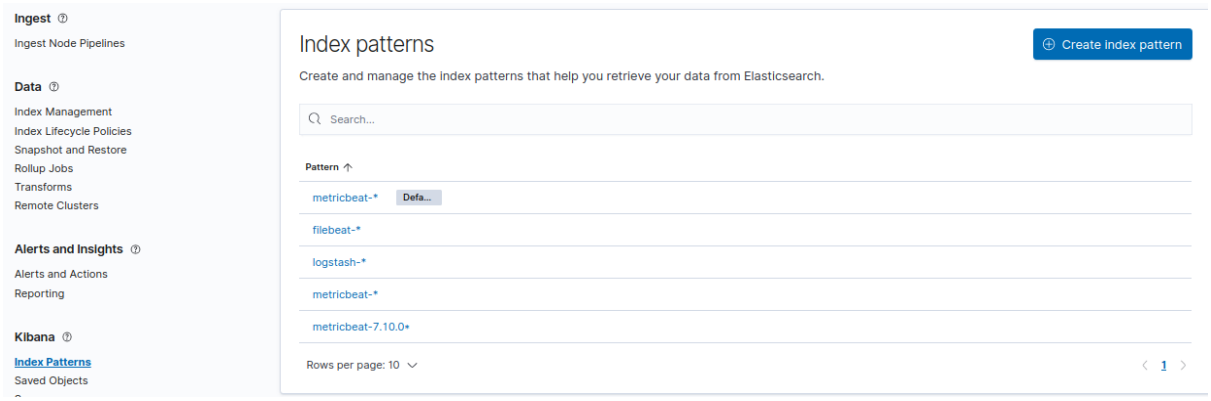


Se encontraron dificultades para lograr el funcionamiento de Logstash, por lo cual **se instaló filebeat, con output a Kibana,** logrando que se visualizaran en tiempo real los logs de snort, según la ejecución de las reglas creadas para detección de comportamiento extraño.

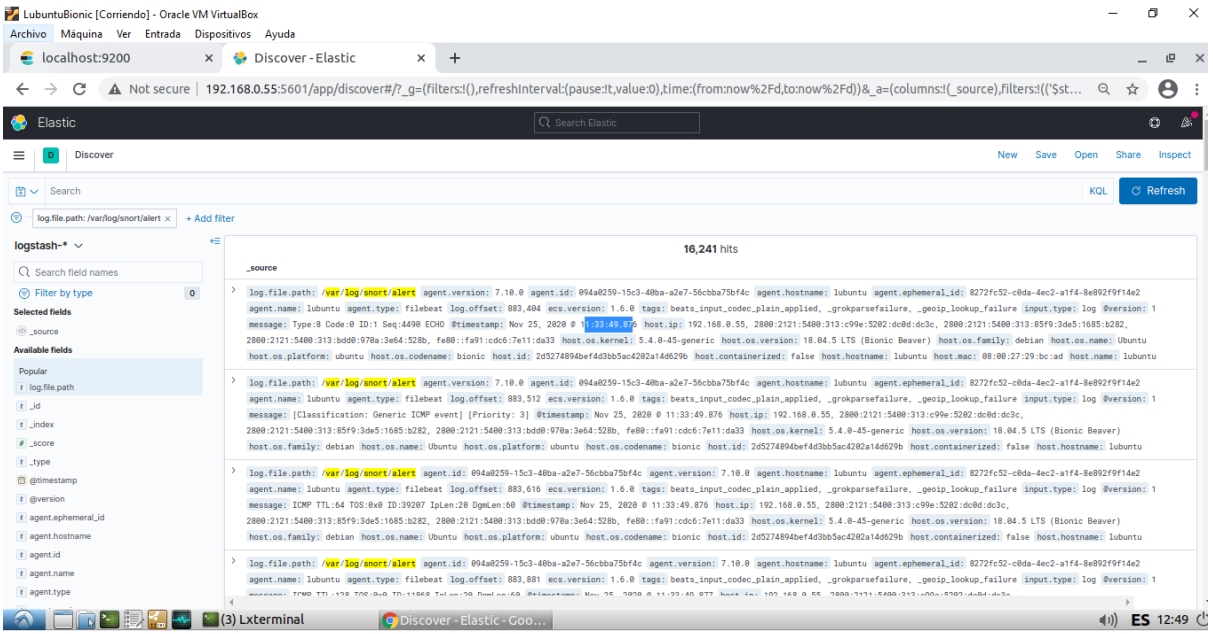
Luego de varias modificaciones y pruebas en los archivos logstash.yml y ag2.conf, se logró que el índice aparezca dentro de la administración de índices disponibles:



Se crea el parámetro Logstash-* asociado a ese índice, desde Kibana.



Desde Kibana -> Discover, se filtra por el índice Logstash y para que muestre los resultados de los logs del archivo /var/log/snort/alert.



Snort

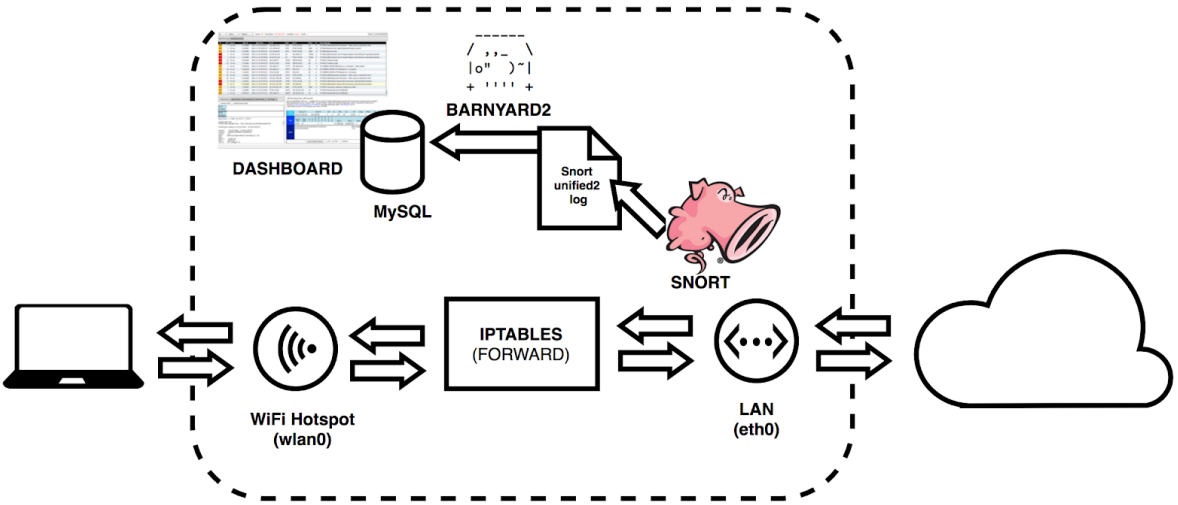
Es un Sistema de Detección de Intrusos (IDS) basado en red digital de servicios integrados.

Consiste en un sistema basado en una red que monitoriza todo un dominio de colisión y funciona detectando usos indebidos y maliciosos.

Cuenta con un lenguaje de creación de reglas en el que se pueden definir los patrones que se utilizarán a la hora de monitorizar el sistema.

Funcionamiento

- **Modo sniffer,** en el que se motoriza por pantalla en tiempo real toda la actividad en la red en que Snort está configurado.
- **Modo packet logger** (registro de paquetes), en el que se almacena en un sistema de log toda la actividad de la red en que se ha configurado Snort para un posterior análisis.
- **Modo IDS**, en el que se motoriza por pantalla o en un sistema basado en log, toda la actividad de la red a través de un fichero de configuración en el que se especifican las reglas y patrones a filtrar para estudiar los posibles ataques. **(Es el modo en que lo usaremos en la prueba de concepto)**



Instalación

Una vez hemos instalado correctamente el programa y lo ponemos en funcionamiento, debemos introducir en la base de patrones de ataques los que queremos utilizar para detectar actividades sospechosas contra nuestra red.

Se elaboró un documento para la instalación (**Instalación y Configuración ELK + Snort.pdf**), donde se explica el paso a paso en la instalación de Snort.

Reglas

Snort ya viene con reglas predeterminadas, creadas en la instalación. Estos archivos .rules contienen alertas predeterminadas que pueden ser utilizadas para optimizar el ids.

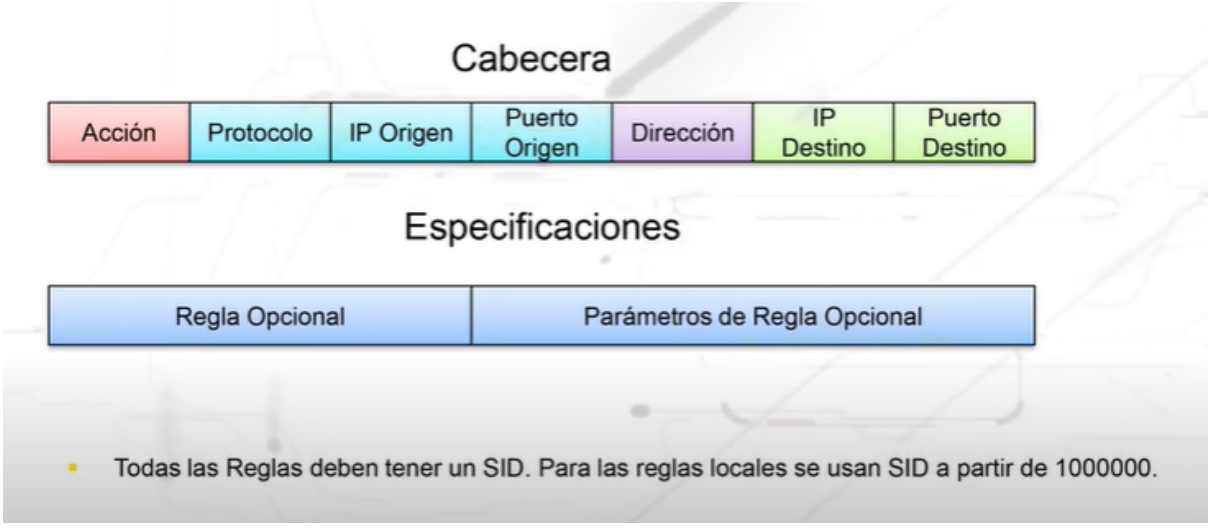
```
root@lubuntu:/etc/snort/rules# ls
attack-responses.rules          community-nntp.rules           deleted.rules       netbios.rules    sql.rules
backdoor.rules                  community-oracle.rules         dns.rules           nntp.rules       telnet.rules
bad-traffic.rules               community-policy.rules         dos.rules           oracle.rules     tftp.rules
chat.rules                      community-sip.rules            experimental.rules  other-ids.rules  virus.rules
community-bot.rules             community-smtp.rules           exploit.rules       p2p.rules        web-attacks.rules
community-deleted.rules         community-sql-injection.rules  finger.rules        policy.rules     web-cgi.rules
community-dos.rules             community-virus.rules          ftp.rules           pop2.rules       web-client.rules
community-exploit.rules         community-web-attacks.rules    icmp-info.rules     pop3.rules       web-coldfusion.rules
community-ftp.rules             community-web-cgi.rules        icmp.rules          porn.rules       web-frontpage.rules
community-game.rules            community-web-client.rules     imap.rules          rpc.rules        web-iis.rules
community-icmp.rules            community-web-dos.rules        info.rules          rservices.rules  web-misc.rules
community-imap.rules            community-web-iis.rules        local.rules         scan.rules       web-php.rules
community-inappropriate.rules   community-web-misc.rules       misc.rules          shellcode.rules  x11.rules
community-mail-client.rules     community-web-php.rules        multimedia.rules    smtp.rules
community-misc.rules            ddos.rules                     mysql.rules         snmp.rules
root@lubuntu:/etc/snort/rules# nano local.rules
```

Configuración de reglas (local.rules). en snort con las cual va a realizar el escaneo de los datos.

Para la creación de reglas, se accede al archivo de reglas locales de snort

```
root@lubuntu:/home/lubuntu# nano /etc/snort/rules/local.rules
```

Se debe tener en cuenta que cada regla tiene un formato estándar de manera que pueda ser entendible por cualquier usuario con conocimiento de su sintaxis o administrador de red.



Se crea una regla para el protocolo ICMP, con un mensaje desde cualquier puerto.

```
  GNU nano 2.9.3                                  /etc/snort/rules/local.rules

# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp 192.168.0.1/24 any -> any any (msg:"Alguien esta haciendo ping";sid:19910316;rev:1;)
```

Se accede al archivo de configuración de Snort, para modificar algunos parámetros y verificar la regla creada

```
root@lubuntu:/home/lubuntu# nano /etc/snort/snort.conf
```

Se modifica la ipvar HOME_NET para que apunte a la ip del gateway y el ipvar EXTERNAL_NET con la ip de la pc anfitriona con windows.

```
  GNU nano 2.9.3                        snort.conf                          Modified

# Step #1: Set the network variables.  For more information, see README.variabl$
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
ipvar HOME_NET 192.168.0.1/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you c
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET
```

Se deberán agregar los conjuntos de reglas que se quieran utilizar para detectar las actividades sospechosas que afecten en la red. Se puede pensar que agregando todos los conjuntos de reglas que existan, se detectarán más ataques, pero la realidad es que no. **Cuantas más reglas se añadan, más se sobrecarga el programa, pudiendo llegar a tasas de pérdida de paquetes no analizados muy elevadas, con lo que habrá ataques que el IDS no detecte.**

Para elegir distintos tipos de classtypes, en /etc/snort/classification.config

```
root@lubuntu: /etc/snort
File  Edit  Tabs  Help
  GNU nano 2.9.3                    classification.config

config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1


# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
config classification: suspicious-login,An attempted login using a suspicious username was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: inappropriate-content,Inappropriate Content was Detected,1

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Comprobación de regla creada

Se abre una terminal en la pc anfitriona y se ejecuta un ping para ver si Snort lo detecta por la regla creada en el archivo.

```
root@lubuntu:/home/lubuntu# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.55  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 2800:2121:5400:43b:ea57:4b5c:85b3:79ff  prefixlen 128  scopeid 0x0
```

```
C:\Windows\system32>ping 192.168.0.55

Haciendo ping a 192.168.0.55 con 32 bytes de datos:
Respuesta desde 192.168.0.55: bytes=32 tiempo=1ms TTL=64
Respuesta desde 192.168.0.55: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.55: bytes=32 tiempo<1m TTL=64
Respuesta desde 192.168.0.55: bytes=32 tiempo<1m TTL=64

Estadísticas de ping para 192.168.0.55:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 1ms, Media = 0ms
```

Desde la pc Linux vemos la captura por consola de los paquetes enviados, con los parámetros y el mensaje que agregamos en la regla. Identificando la ip de la cual recibe el ping,

```
root@lubuntu: /etc/snort
File  Edit  Tabs  Help
rity: 0] {ICMP} 192.168.0.55 -> 192.168.0.117
11/14-00:30:55.095397  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
 activity] [Priority: 3] {ICMP} 192.168.0.55 -> 192.168.0.117
11/14-00:30:56.101219  [**] [1:382:7] ICMP PING Windows [**] [Classification: Mi
sc activity] [Priority: 3] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:56.101219  [**] [1:19910316:1] Alguien esta haciendo ping [**] [Prio
rity: 0] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:56.101219  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
ity] [Priority: 3] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:56.101247  [**] [1:19910316:1] Alguien esta haciendo ping [**] [Prio
rity: 0] {ICMP} 192.168.0.55 -> 192.168.0.117
11/14-00:30:56.101247  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
 activity] [Priority: 3] {ICMP} 192.168.0.55 -> 192.168.0.117
11/14-00:30:57.110907  [**] [1:382:7] ICMP PING Windows [**] [Classification: Mi
sc activity] [Priority: 3] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:57.110907  [**] [1:19910316:1] Alguien esta haciendo ping [**] [Prio
rity: 0] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:57.110907  [**] [1:384:5] ICMP PING [**] [Classification: Misc activ
ity] [Priority: 3] {ICMP} 192.168.0.117 -> 192.168.0.55
11/14-00:30:57.110934  [**] [1:19910316:1] Alguien esta haciendo ping [**] [Prio
rity: 0] {ICMP} 192.168.0.55 -> 192.168.0.117
11/14-00:30:57.110934  [**] [1:408:5] ICMP Echo Reply [**] [Classification: Misc
 activity] [Priority: 3] {ICMP} 192.168.0.55 -> 192.168.0.117
```

Inicio de servicio Snort

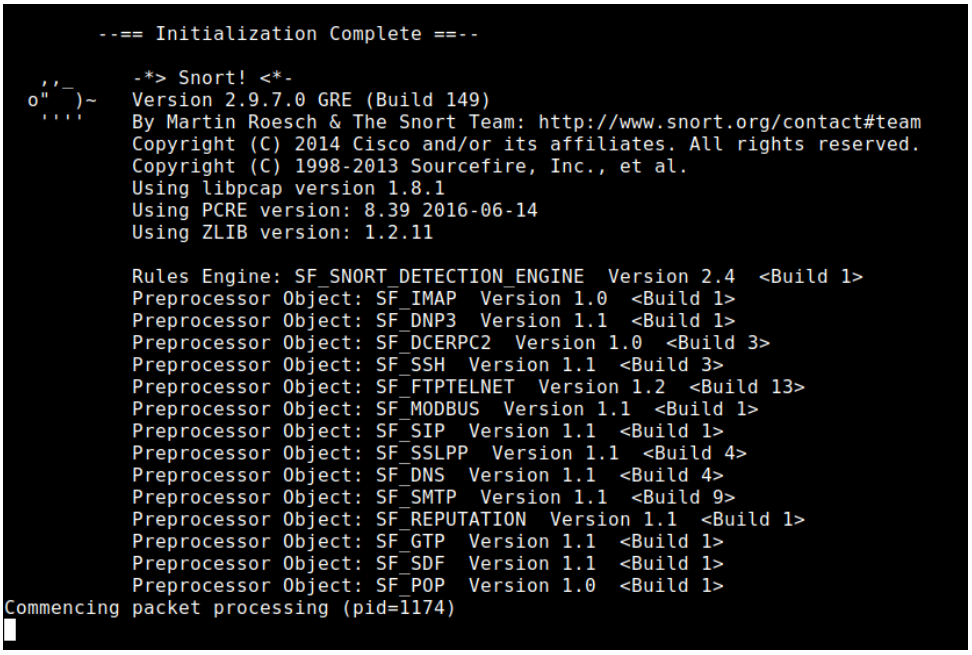
Los modos de ejecución de snort se activan dependiendo de los parámetros que le pasemos al programa.

Para iniciar a Snort en modo NIDS usamos el parámetro -c.
El modo NIDS es utilizado para especificar el directorio del archivo de configuración snort.conf.

Desde una terminal con permisos de administrador.

Accedemos a la ruta **cd /etc/snort/** desde terminal y ejecutamos.

snort -A console -c /etc/snort/snort.conf -i enp0s3



snort -A fast -c /etc/snort/snort.conf -q -i enp0s3

Este comando inicia el rastreo de Snort en enp0s3 (-i), con el modo de alerta rápida (-A), este modo crea el archivo "/var/ log/snort /alert" que contiene todos los registros que enviaremos en Elastic Stack, (-q) significa modo silencioso usando el archivo "snort.conf" (-c) es para especificar el modo NIDS de snort.

Luego de esto esperar hasta que se generen varios registros y luego analizarlos en Elastic Stack.

En este archivo se guardan los logs de snort cada vez que reciba una alerta según la regla que se haya creado en snort/rules/local.rules

Verificamos que esté generando logs

cat /var/log/snort/alert

Vincular logs de snort a Logstash

El archivo de “ingesta” de Logstash (**snort-03.conf** en nuestro caso) se usa para indicarle a Logstash de dónde está ingiriendo datos, cómo está filtrando / analizando datos y a dónde enviarlos. **Ingerimos datos del archivo /var/log/snort/alert** ya que es la ubicación predeterminada para que Snort almacene alertas. Usamos patrones de Grok para filtrar alertas de Snort y mapear / encasillar campos en la alerta a variables que las almacenaremos como en Elasticsearch. El filtrado de Grok es el mejor método de Logstash para analizar texto arbitrario y estructurarlo para que sea fácilmente consultable.

Verificamos en **nano /etc/logstash/conf.d/snort-03.conf, que la configuración del path de los logs de snort sea la correcta.**

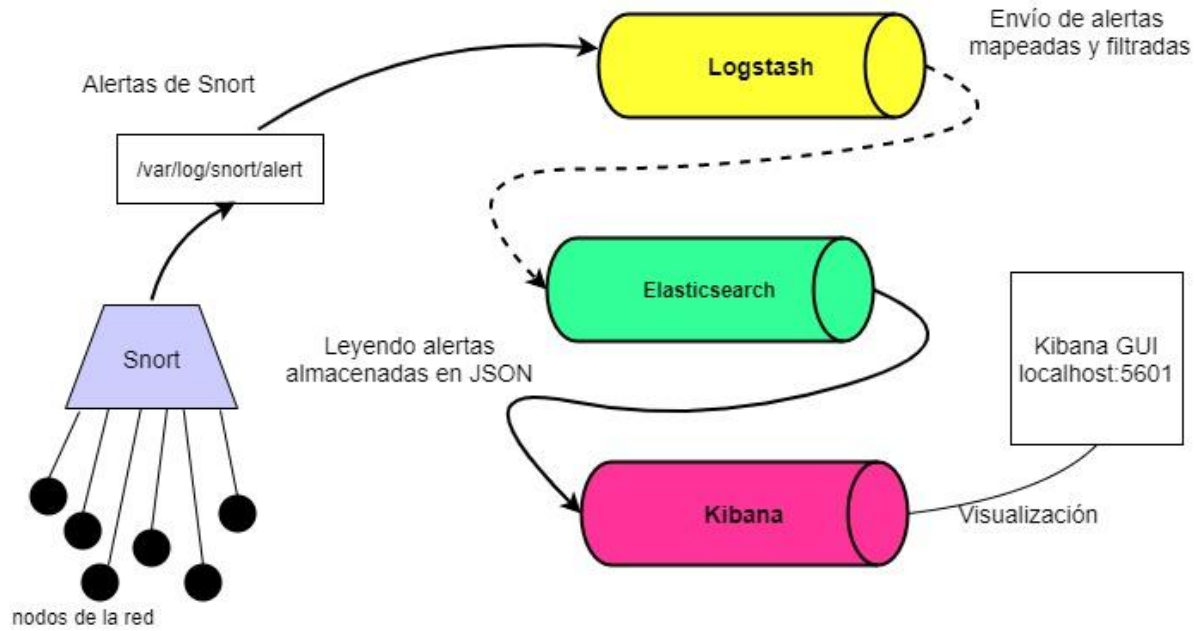
Se testea la configuración agregada

/usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/snort-03.conf --config.test_and_exit

/usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/snort-03.conf --config.reload.automatic

ELK + Snort (SELK) - Funcionamiento

Como se mencionó anteriormente, Elasticsearch, Logstash y Kibana son herramientas proporcionadas por la empresa Elastic y, por lo tanto, funcionan bien juntas desde el primer momento. Hacer que esta pila esté lista y funcionando con su configuración predeterminada no es demasiado difícil, pero la configuración se vuelve más compleja cuando se intenta personalizar configuraciones, como alterar los métodos de almacenamiento de Elasticsearch o los patrones de índice personalizados de Kibana. Logstash requirió la mayor parte del trabajo, ya que necesitábamos instruir al sistema sobre cómo analizar y filtrar nuestras alertas de Snort.



La pila SELK se creó para trabajar junto con Snort y puede ingerir datos directamente desde la salida de alerta de Snort e ingerir sus alertas generadas. Las alertas son ingeridas, filtradas / mapeadas y enviadas por Logstash a Elasticsearch. Elasticsearch almacena e indexa los datos y luego Kibana lee los datos en Elasticsearch y los muestra en una interfaz gráfica fácil de usar. En Kibana, la pestaña "Descubrir" se puede utilizar para consultar datos y mostrarlos por campos en la alerta o en JSON sin procesar.

CONCLUSIÓN

El sistema SELK proporciona a los usuarios finales un sistema para analizar, almacenar, analizar y visualizar alertas de Snort, y este método es aplicable a cualquier dato. Se puede hacer mucho más con los datos de alerta de Snort y analizarlos fácilmente en su totalidad o por días, meses, años, etc. Si Snort está configurado para ejecutarse en tiempo real, podemos monitorear nuestros datos de alerta de Snort en vivo y encontrar ataques, analizarlos y detenerlos antes de que escalen más fácilmente.

Links útiles

https://www.elastic.co/es/downloads/

https://www.snort.org/

https://www.avantica.com/es/blog/elk-stack-implementacion-facil-con-docker#:~:text=ELK%20es%20la%20combinaci%C3%B3n%20de,de%20los%20datos%20de%20registro.

https://www.youtube.com/watch?v=8ycqfPtGUB0

https://blog.snort.org/2017/11/snort-30-with-elasticsearch-logstash.html

https://www.youtube.com/watch?v=x0YqJIV5RZk

https://medium.com/@armendgashx/forwarding-snort-logs-to-elk-stack-371232699e7f

https://www.youtube.com/watch?v=5lJK0EsFepg

https://www.adictosaltrabajo.com/2016/05/16/extraccion-de-logs-con-logstash/

https://www.elastic.co/es/blog/a-practical-introduction-to-logstash

https://www.youtube.com/watch?v=BTqyRxqnyE8

https://www.youtube.com/watch?v=Cu6LjFslQWI&t=455s