



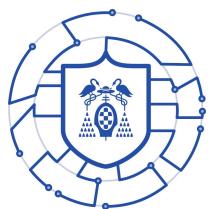
Universidad
de Alcalá

Práctica 2 - SIEMs

Máster en ciberseguridad

Asignatura: Sistemas de información para la
ciberseguridad

Kevin van Liebergen Jorge Lafuente



2021

Índice general

Índice general	1
Índice de figuras	2
1 Enunciado	4
2 Parte 1 - OSSIM & OSSEC	5
2.1. Tripwire	5
2.2. Diagrama de arquitectura	7
2.3. Instalación	8
2.4. Ejemplo práctico	10
2.5. SIEM	14
3 Parte 2 - SIEM en ElasticSearch	16
3.1. Diagrama de arquitectura	16
3.2. Instalación y configuración	17
3.3. HAProxy	21
3.4. Kibana	24
3.5. Docker-compose	27
3.6. Auditbeat	32
3.7. Packetbeat	38
3.8. Filebeat	45

Índice de figuras

1.	Tripwire --check	5
2.	Resultados obtenidos del tripwire check	6
3.	Resultados obtenidos del tripwire check	7
4.	Diagrama de arquitectura	7
5.	Página bienvenida OSSIM	8
6.	OSSEC mensaje finalización	9
7.	OSSECs connectados al OSSIM	10
8.	Comando para realizar fuerza bruta con Hydra	10
9.	Alertas desde el dashboard por la fuerza bruta	11
10.	Resumen del escaneo de vulnerabilidades	11
11.	Ataques detectados	12
12.	Ataques realizados detectados por los OSSEC	12
13.	Ataques realizados en forma de gráfico de queso	13
14.	Trozo del reporte PDF	13
15.	Fallo detallado del reporte PDF	14
16.	Información acerca de ataques recibidos	14
17.	Información de un ataque	15
18.	Diagrama de la estructura	16
19.	Estado del servicio Docker	17
20.	Descarga de imágenes	18
21.	Inicio del nodo 1	19
22.	Inicio del nodo 2	20
23.	Inicio del nodo 3	20
24.	Configuración usuarios de elasticsearch	21
25.	Descarga de imagen HAProxy	21
26.	Inicio del contenedor HAProxy	23
27.	Panel de estado de HAProxy	23
28.	Error de autenticación	24
29.	Estado de los nodos	24
30.	Inicio del contenedor de kibana	25
31.	Panel inicio kibana	26
32.	Estado de los nodos en kibana	26
33.	Red ‘jorge_default’ en docker	28
34.	Inicio de docker-compose	30
35.	Configuración usuarios de elasticsearch	31
36.	Descarga de auditbeat	32
37.	Instalación de auditbeat	33
38.	Creación de visualizaciones auditbeat	34
39.	Inicio de la ejecución de auditbeat	34
40.	Comprobación de recepción de datos auditbeat	35
41.	Dashboards de auditbeat	35
42.	Dashboards overview de auditbeat	36

43.	Dashboards hosts de auditbeat	36
44.	Ataque de fuerza bruta contra ssh	37
45.	Dashboard logins	37
46.	Eventos de login en el dashboard de logins	38
47.	Instalación de Packetbeat	39
48.	Creación de dashboards de Packetbeat	40
49.	Inicio del servicio de Packetbeat	41
50.	Comprobación Packetbeat	41
51.	Dashboards Packetbeat	42
52.	Dashboard DNS Overview Packetbeat	43
53.	Top 10 servidores DNS consultados	43
54.	FQDNs únicos	44
55.	Top 10 FQDNs únicos utilizados	45
56.	Descargar filebeat	46
57.	Instalar filebeat	46
58.	Habilitar entrada para suricata	47
59.	Habilitar entrada para suricata	47
60.	Iniciamos suricata	48
61.	Alertas suricata	48
62.	Iniciamos filebeat	49
63.	Comprobación filebeat	49
64.	Ingesta de datos de filebeat	50
65.	Dashboard de eventos de Suricata	50
66.	Log de eventos de Suricata	51
67.	Top alertas de Suricata	51

1 Enunciado

Parte 1

Implementar un SIEM basado en OSSIM para monitorizar los diferentes ordenadores.

- La implementación requiere:
 - Un servidor de OSSIM
 - Al menos dos agentes de OSSEC
- Explicar diferentes opciones de las posibilidades que nos ofrece OSSIM.
 - Breve resumen de partes interesantes
 - Ejemplo de recolección de logs de algún tipo de ataque
- La implementación debe cumplir con las siguientes características:
 - Se deberá realizar algún tipo de ataque sobre algún equipo para comprobar que OSSIM recibe toda la información del ataque
- El alumno deberá entregar un documento que contenga lo siguiente:
 - Explicación detallada de cada módulo y su funcionamiento
 - Diagrama de arquitectura de la implementación
 - Imágenes de captura de la implementación en funcionamiento
 - Ejemplos demostrativos de casos de uso para monitoreo SIEM
 - Las capturas de imágenes deben contar con el nombre del alumno en el prompt del sistema para poder comprobarse la autenticidad

Parte 2

- Desarrollar una implementación SIEM completa en Elasticsearch para monitoreo de tráfico DNS, registros de autenticación y eventos
- Desarrollar las visualizaciones necesarias en Kibana para monitorear cada uno de los requerimientos
- La implementación requiere:
 - Elasticsearch
 - Kibana
 - Filebeat
 - Packetbeat

- Auditbeat
- La implementación debe cumplir con las siguientes características:
 - Modulo de seguridad Elasticsearch activado y con cuentas configuradas
 - Al menos 2 nodos de Elasticsearch como master-eligible y 1 nodo dedicado a data ingest
 - Alta disponibilidad comprobable
- El alumno deberá entregar un documento que contenga lo siguiente:
 - Explicación detallada de cada modulo y su funcionamiento
 - Diagrama de arquitectura de la implementación
 - Imágenes de captura de la implementación en funcionamiento
 - Ejemplos demostrativos de casos de uso para monitoreo SIEM
 - Las capturas de imágenes deben contar con el nombre del alumno en el prompt del sistema para poder comprobarse la autenticidad

2 Parte 1 - OSSIM & OSSEC

2.1. Tripwire

En esta práctica además se ha instalado el HIDS Tripwire para conocer su funcionamiento e instalación, como aparecen en las diapositivas únicamente es necesario realizar el siguiente comando para instalarlo:

```
$ sudo apt install tripwire
```

Una vez hemos modificado los archivos de configuración e iniciar el servicio podemos realizar una comprobación acerca de los archivos del sistema en busca en ficheros maliciosos o ficheros legítimos modificados con el comando que aparece en la imagen 1.

```
kevin@VirtualBox:~$ sudo tripwire --check
Parsing policy file: /etc/tripwire/tw.pol
*** Processing Unix File System ***
Performing integrity check...
```

Figura 1: Tripwire --check

El comando anterior nos arroja información general acerca de las esadísticas de los ficheros del sistema, así como archivos modificados, el nivel de gravedad, binarios y demás información que podemos observar en la imagen 2.

Section: Unix File System					
Rule Name	Severity Level	Added	Removed	Modified	
Other binaries	66	0	0	0	
Tripwire Binaries	100	0	0	0	
Other libraries	66	0	0	0	
Root file-system executables	100	0	0	0	
Tripwire Data Files	100	0	0	0	
* System boot changes (/var/log)	100	1	0	0	
Root file-system libraries (/lib)	100	0	0	0	
Critical system boot files	100	0	0	0	
Other configuration files (/etc)	66	0	0	0	
Boot Scripts	100	0	0	0	
Security Control	66	0	0	0	
Root config files	100	0	0	0	
* Devices & Kernel information	100	4410	4329	0	
Invariant Directories	66	0	0	0	
Total objects scanned: 188206					
Total violations found: 8740					

Figura 2: Resultados obtenidos del tripwire check

Además, te muestra información más específica acerca de cada fichero, por ejemplo los archivos que han sido añadidos que no vienen por defecto en el sistema operativo como son `/var/log/mail.err`, `/etc/postfix/sasl_passwd`, etc, y archivos que han sido modificados, tal y como se muestra en la imagen 3.

```

Added:
"/var/log/mail.err"

-----
Rule Name: Other configuration files (/etc)
Severity Level: 66
-----

Added:
"/etc/postfix/cacert.pem"
"/etc/postfix/sasl_passwd"
"/etc/postfix/sasl_passwd.db"

Modified:
"/etc/alternatives"
"/etc/alternatives/.dotlock"
"/etc/alternatives/.dotlock.1.gz"

```

Figura 3: Resultados obtenidos del tripwire check

2.2. Diagrama de arquitectura

Continuando con la práctica vamos a comenzar a instalar la arquitectura solicitada, dos agentes OSSEC y un servidor OSSIM. La arquitectura creada es la que aparece en la figura 4, en una red NAT de VirtualBox, siendo la máquina Ubuntu un agente OSSEC y la máquina AlienVault un Debian con un agente OSSEC y el servidor OSSIM.

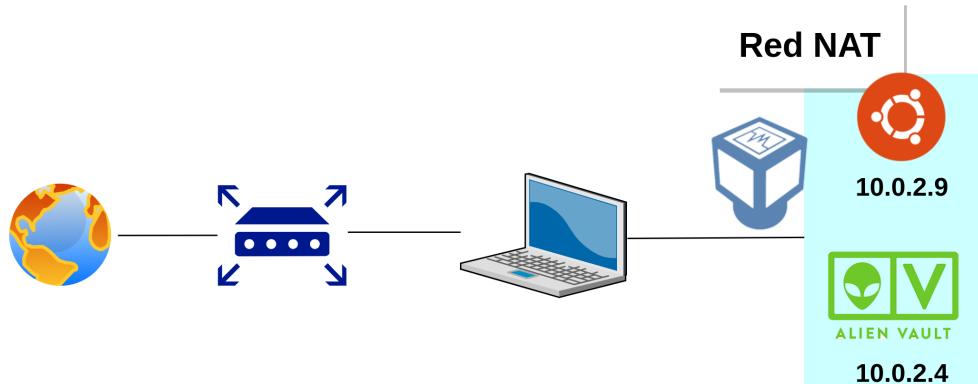


Figura 4: Diagrama de arquitectura

Como se ha comentado en las diapositivas los detalles técnicos, no nos vamos a centrar demasiado en ello, cabe mencionar que OSSEC es un IDS

basado en Host, que recoge información y la envía a un servidor central, que se encarga de monitorizar y controlar sistemas.

OSSIM es un SIEM Open Source que centraliza los datos recogidos por los agentes OSSEC, según las diapositivas para instalar esta arquitectura vamos a instalar antes que nada un servidor OSSIM y después añadir los agentes OSSEC añadiendo los agentes. Para ello, vamos a utilizar VirtualBox para crear las máquinas virtuales, pero en lugar de conectar la tarjeta de red en modo *Bridge* o puente se ha decidido realizarlo en modo red Nat, como se ha comentado en Diagrama de arquitectura.

2.3. Instalación

La instalación fallaba cuando se elegía 16 gb de almacenamiento, por lo que se ha elegido 30 gb (Fixed Sized Storage) y 8 gb de memoria ram, además se ha instalado OSIMM en su última versión 5.6.5 (en lugar de la versión 5.2.4 de las diapositivas).

Una vez se ha descargando la imagen .iso e instalado en la máquina y configurado la red accedemos al servidor web, nos aparece un mensaje de bienvenida y unos campos para la creación de un usuario en el sistema como aparece en la figura 5.

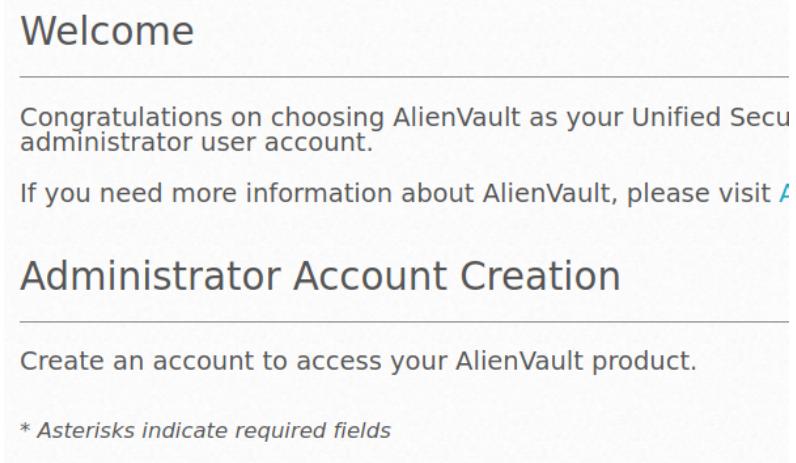


Figura 5: Página bienvenida OSSIM

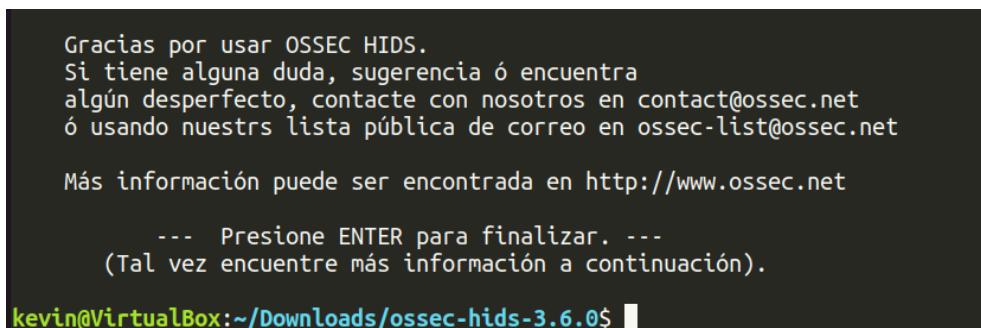
Una vez hemos configurado el usuario nuevo vamos a proceder a la creación del agente en la máquina OSSIM, como aparecen en las diapositivas tenemos que lanzar los siguientes comandos.

```
$ /etc/init.d/ossec start
```

Ahora procedemos a la creación del agente OSSEC en la máquina Ubuntu, para la instalación de OSSEC instalamos la versión 2.8.3 que se encuentra en la siguiente dirección <https://github.com/ossec/ossec-hids/releases/tag/v2.8.3>.

```
$ wget https://github.com/ossec/ossec-hids/archive/v2.8.3.tar.gz
$ tar -xvf ossec-hids-2.8.3.tar.gz
$ apt install libbz-dev libssl-dev libpcre2-dev libevent-dev
build-essential
$ cd ossec-hids-2.8.3
$ sudo ./install.sh
```

Hay que darle a **agente** en la instalación, en lugar de a local, y añadir la dirección IP del servidor central OSSIM, en este caso la dirección 10.0.2.4. El mensaje de finalización desde el ordenador Ubuntu es el que aparece en la imagen 6.



Gracias por usar OSSEC HIDS.
Si tiene alguna duda, sugerencia ó encuentra
algún desperfecto, contacte con nosotros en contact@ossec.net
ó usando nuestras lista pública de correo en ossec-list@ossec.net

Más información puede ser encontrada en http://www.ossec.net

--- Presione ENTER para finalizar. ---
(Tal vez encuentre más información a continuación).

kevin@VirtualBox:~/Downloads/ossec-hids-3.6.0\$ █

Figura 6: OSSEC mensaje finalización

Una vez hemos instalado el agente, lo añadimos al servidor OSSIM mediante los siguientes comandos (en el servidor OSSIM):

```
$ cd /var/ossec/bin
$ ./manage_agents
```

Añadimos un agente (opción A) y le asignamos un nombre y la dirección IP de la máquina Ubuntu. Obtenemos y copiamos la clave desde el servior OSSIM y desde la máquina Ubuntu la importamos al agente, reiniciamos el servicio OSSEC mediante:

```
$ sudo /var/ossec/bin/ossec-control restart
```

Y abrimos la aplicación Web para ver que se conectan los dos agentes con el OSSIM, debido al firewall no conectaba por lo que hemos tenido que ejecutar el comando:

```
$ iptables -I OUTPUT 1 -p udp --dport 1514 -j ACCEPT
```

Y finalmente tenemos instalado y configurado el servidor OSSIM con los dos agentes OSSEC como se muestra en la imagen 7.

ID	AGENT NAME	ASSET	IP/CIDR	CURRENT IP	CURRENT USER	STATUS	ACTIONS
000	alienvault (server)	alienvault	127.0.0.1	127.0.0.1	-	Active/local	
001	ubuntu_kevin	Host-10-0-2-9	10.0.2.9	10.0.2.9	-	Active	

Figura 7: OSSECs conectados al OSSIM

2.4. Ejemplo práctico

A continuación se va a proceder a realizar un ataque de fuerza bruta hacia el servidor SSH mediante la herramienta Hydra, se ha descargado un fichero con las 256 contraseñas más utilizadas en rockyou en el repositorio SecLists¹.

Y con ello ejecutamos el comando que aparece en la figura 8, desde la máquina Ubuntu hacia el servidor OSSIM AlienVault

```
kevin@VirtualBox:~$ hydra -l root -P rockyou-20.txt ssh://10.0.2.4 -t 4
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-01-0
1 15:01:39
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to
skip waiting)) from a previous session found, to prevent overwriting, .
/hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 512 login tries (l:1/p
:512), ~128 tries per task
[DATA] attacking ssh://10.0.2.4:22/
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 468 to do in 00:11h, 4 act
ive
```

Figura 8: Comando para realizar fuerza bruta con Hydra

Y podemos observar que en tiempo real se actualiza las alertas en la sección ‘Delivery Attack’, y qué dirección IP es la causante de la autenticación por fuerza bruta, el riesgo de este ataque es ‘Low’ según AlienVault.

¹<https://github.com/danielmiessler/SecLists>

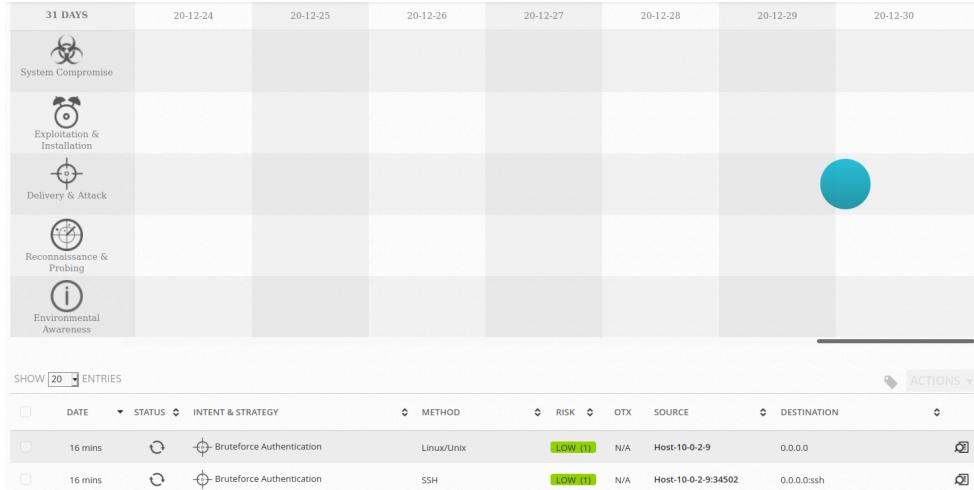


Figura 9: Alertas desde el dashboard por la fuerza bruta

Además desde la sección **Environment >Vulnerabilities** podemos hacer un escaneo de las vulnerabilidades y problemas la configuración de los archivos de los agentes OSSEC, realizando un escaneo con una duración aproximada de 30 minutos nos ha generado un reporte exportado a PDF para las máquinas escaneadas. Podemos observar en la figura 10 un resumen de los fallos en ambos sistemas, el servidor OSSIM (10.0.2.4) tiene dos fallos críticos mientras que la máquina Ubuntu no tiene fallos críticos, sin embargo ambos sistemas tienen múltiples fallos medianos y leves.

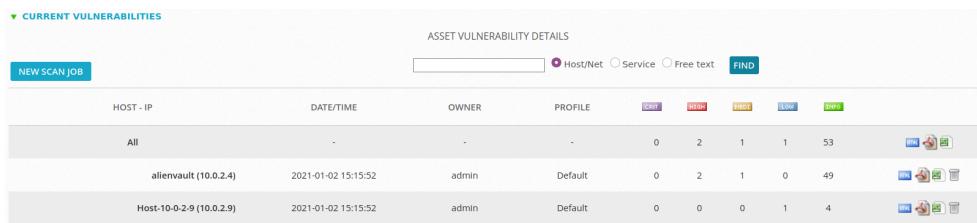


Figura 10: Resumen del escaneo de vulnerabilidades

Para realizar el escaneo se procede a realizar diversos ataques, más concretamente de tipo ‘Delivery & Attack’ y ‘Exploitation & Installation’, vemos que el servidor OSSIM detecta que están atacándole y lo muestra en el Dashboard como se muestra en la figura 11.

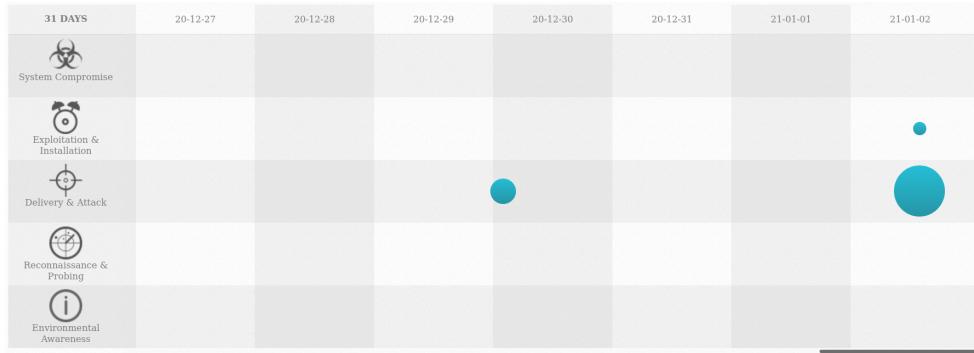


Figura 11: Ataques detectados

Además cuando se termina de escanear, observamos en la sección List View (**Analysis >Alarms >List view**) que se especifica más en detalle los ataques que se han probado, en la figura 12 se muestra Fuerza bruta por autenticación, Inyección SQL y XSS al servidor Web.

SHOW 20 ENTRIES									ACTIONS
	DATE	STATUS	INTENT & STRATEGY	METHOD	RISK	OTX	SOURCE	DESTINATION	
	26 mins	○	Bruteforce Authentication	Linux/Unix	HIGH (1)	N/A	alienVault	0.0.0.0	🔗
	25 mins	○	Bruteforce Authentication	SSH	LOW (1)	N/A	alienVault:57889	0.0.0.0:ssh	🔗
	25 mins	○	Bruteforce Authentication	SSH	LOW (1)	N/A	alienVault:53271	0.0.0.0	🔗
	26 mins	○	WebServer Attack	XSS	LOW (1)	N/A	alienVault	0.0.0.0	🔗
	26 mins	○	WebServer Attack - SQL Injection	Attack Pattern Detection	LOW (1)	N/A	alienVault	0.0.0.0	🔗
	2020-12-30 02:18:55	open	Bruteforce Authentication	Linux/Unix	LOW (1)	N/A	Host-10-0-2-9	0.0.0.0	🔗
	2020-12-30 02:11:16	open	Bruteforce Authentication	SSH	LOW (1)	N/A	Host-10-0-2-9:34502	0.0.0.0:ssh	🔗

Figura 12: Ataques realizados detectados por los OSSEC

También podemos observar información en forma de gráfico de queso como los que aparecen en la figura 13 en la sección **Environment >Detection**.

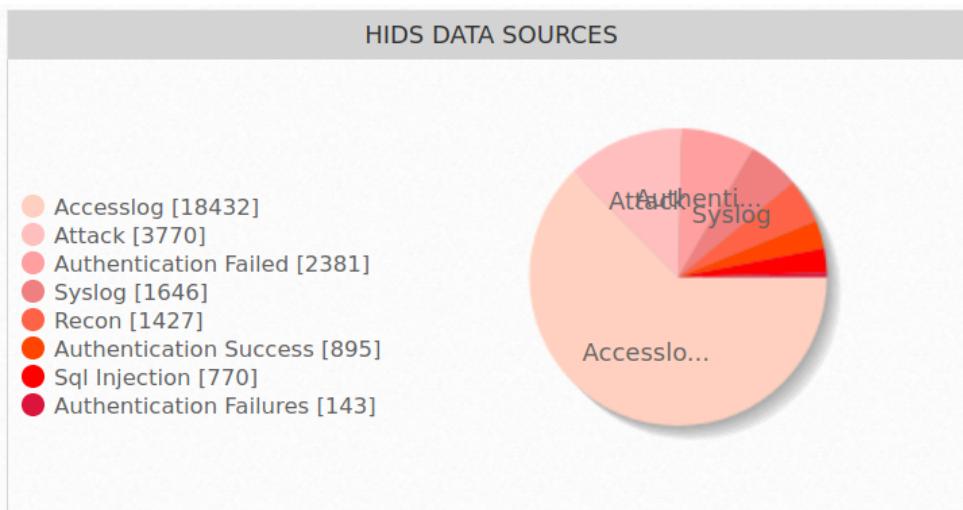


Figura 13: Ataques realizados en forma de gráfico de queso

Como se ha comentado, el escaneo de vulnerabilidades permite crear un reporte en formato HTML y PDF donde también se explican en detalle los sistemas analizados, severidad de los fallos junto con su descripción y solución. En la figura 14 se observa un resumen general del análisis mientras que en la figura 15 especifican detalles de la vulnerabilidad.

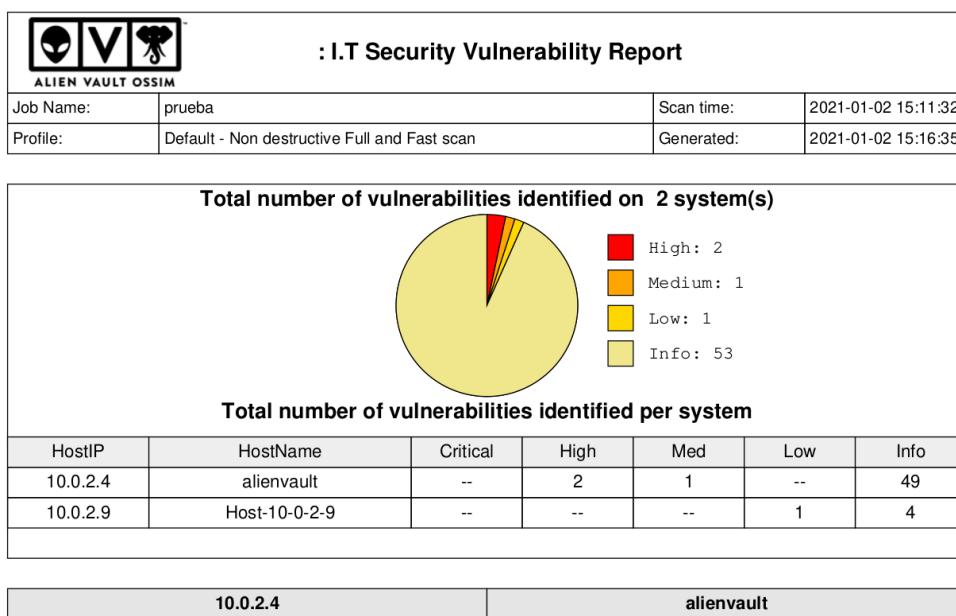


Figura 14: Trozo del reporte PDF

High:
OpenVAS Framework Components End Of Life Detection
Risk: High
Application: unknown
Port: 9390
Protocol: tcp
ScriptID: 108197
Vulnerability Detection Result:
The "OpenVAS Manager" version on the remote host has reached the end of life.
CPE: cpe:/a:openvas:openvas_manager:7.0
Installed version: 7.0
Location/URL: 9390/tcp
EOL version: 7.0
EOL date: 2020-12-31
CVSS Base Vector:
AV:N/AC:L/Au:N/C:C/I:C/A:C
Vulnerability Detection Method:
Checks if a vulnerable version is present on the target host.
Solution:
Update the OpenVAS framework / Greenbone Vulnerability Management (GVM) component version on the remote host to a still supported version.

Figura 15: Fallo detallado del reporte PDF

2.5. SIEM

Además el SIEM o Gestión de Eventos e Información de Seguridad (Security Information and Event Management) recoge y centraliza todo tipo de ataques que detecte, en la figura 16 se observa algunos ataques que ha recolectado, los que se muestran en la imagen son de severidad baja. Además es posible clickar en un ataque recolectado para que nos muestre más información (dirección origen del atacante, dirección IP víctima, tipo de ataque, protocolo utilizado, categoría y subcategoría del ataque, información adicional, prioridad, nivel de riesgo, etc.)

EVENT NAME	DATE GMT+1:00	SENSOR	OTX	SOURCE	DESTINATION	ASSET S → D	RISK	Actions
AlienVault HIDS: Login session closed.	2021-01-02 15:59:32	alienvault	N/A	0.0.0.0	0.0.0.0	2→2	LOW (0)	
AlienVault HIDS: Login session closed.	2021-01-02 15:59:32	alienvault	N/A	0.0.0.0	0.0.0.0	2→2	LOW (0)	
sudo: Session closed	2021-01-02 15:59:30	alienvault	N/A	0.0.0.0	0.0.0.0	2→2	LOW (0)	
sudo: Session closed	2021-01-02 15:59:30	alienvault	N/A	0.0.0.0	0.0.0.0	2→2	LOW (0)	
SSHd: Session disconnected	2021-01-02 15:59:30	alienvault	N/A	alienvault:54748	alienvault:22	2→2	LOW (0)	

Figura 16: Información acerca de ataques recibidos

Event details

directive_event: AV-FREE-FEED Bruteforce attack, login authentication attack against 0.0.0.0

DATE	2021-01-02 15:11:11 GMT+1:00	CATEGORY	Alarm
ALIENVault SENSOR	Unknown	SUB-CATEGORY	Bruteforce
DEVICE IP	N/A	DATA SOURCE NAME	directive_alert
EVENT TYPE ID	50004	DATA SOURCE ID	1505
UNIQUE EVENT ID#	27c90186-4d04-11eb-b3c6-08005dd6fe10	PRODUCT TYPE	Alarm
PROTOCOL	TCP	ADDITIONAL INFO	N/A
PRIORITY	RELIABILITY	RISK	OTX INDICATORS
4	10	High	0
SOURCE	alienVault [10.0.2.4]	DESTINATION	0.0.0.0

Figura 17: Información de un ataque

3 Parte 2 - SIEM en ElasticSearch

3.1. Diagrama de arquitectura

Para esta parte se ha implementado un SIEM en elasticSearch, pero en lugar de realizarlo mediante las diapositivas se ha instalado mediante docker.

Para realizar la implementación de un **SIEM** con la **pila ELK** hemos decidido utilizar docker para crear varios contenedores con los nodos de elasticsearch y kibana, con el fin de establecer la estructura de nodos necesario para ello.

Para garantizar la **alta disponibilidad**, hemos decidido crear **tres nodos** con 2 nodos dedicados a que sean elegibles para ser nodos master y 1 nodo dedicado a data e ingest.

Además de los tres nodos, dos de ellos dedicados como master, se va a utilizar un proxy inverso para realizar las operaciones con el cluster de elasticsearch a través de él y realizar el balanceo de carga y garantizar la alta disponibilidad. Para esta tarea, se va a utilizar un contenedor con **HAProxy** que apuntará a los tres nodos de elastic y expondrá el puerto que utiliza el cluster de elasticsearch.

La arquitectura que se va a montar es la siguiente:

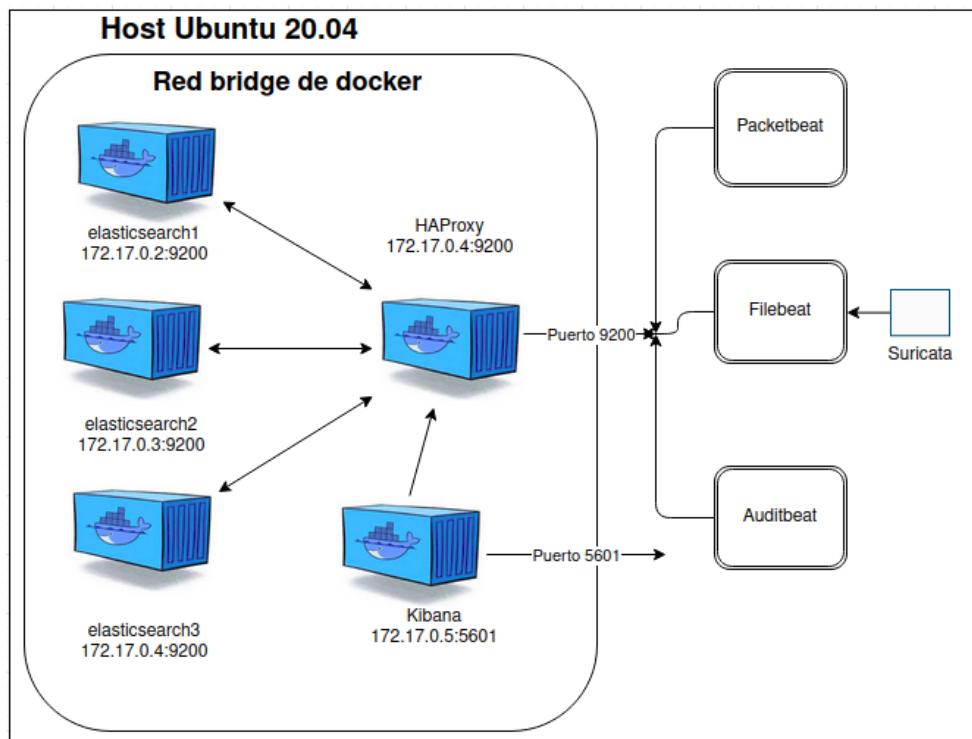


Figura 18: Diagrama de la estructura

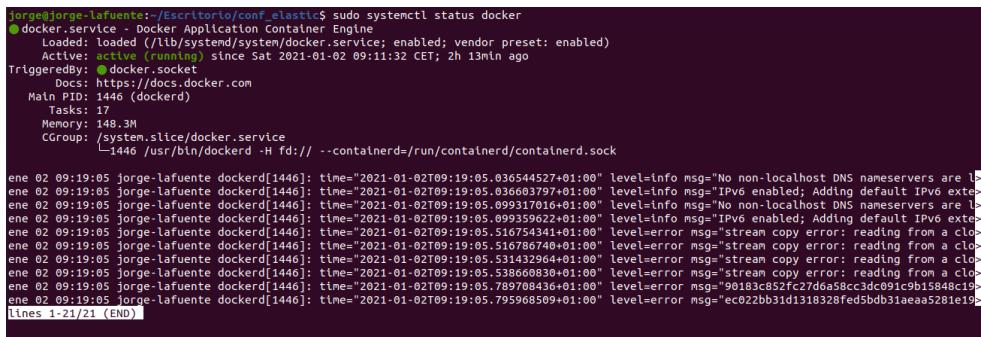
3.2. Instalación y configuración

Lo primero que vamos a hacer es instalar docker en la máquina y configurar el usuario para que tenga los permisos de docker (introducirlo en el grupo de usuarios de docker). La máquina es un Ubuntu 20.04. Para ello, ejecutamos los siguientes comandos:

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
$ sudo apt update
$ sudo apt install docker-ce
```

Con esto ya tendríamos instalado Docker, lo podemos ver con el siguiente comando:

```
$ sudo systemctl status docker
```



```
jorge@jorge-lafuente:~/Escritorio/conf_elastic$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2021-01-02 09:11:32 CET; 2h 13min ago
       Docs: https://docs.docker.com
    TriggeredBy: ● docker.socket
      Main PID: 1446 (dockerd)
        Tasks: 17
       Memory: 148.3M
      CGroup: /system.slice/docker.service
              └─1446 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.036544527+01:00" level=info msg="No non-localhost DNS nameservers are l
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.036603797+01:00" level=info msg="IPv6 enabled; Adding default IPv6 exte
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.099317016+01:00" level=info msg="No non-localhost DNS nameservers are l
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.100000000+01:00" level=info msg="IPv6 enabled; Adding default IPv6 exte
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.516754341+01:00" level=error msg="stream copy error: reading from a clo
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.516786748+01:00" level=error msg="stream copy error: reading from a clo
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.531432964+01:00" level=error msg="stream copy error: reading from a clo
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.538660830+01:00" level=error msg="stream copy error: reading from a clo
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.789798436+01:00" level=error msg="90183c852fc27d6a58cc3dc091c9b15848c19
ene 02 09:19:05 jorge-lafuente dockerd[1446]: time="2021-01-02T09:19:05.795960850+01:00" level=error msg="ecc022bb31d1318328fed5bd31aea5281e19"
lines 1-21/21 (END)
```

Figura 19: Estado del servicio Docker

Una vez tenemos docker instalado, tenemos que descargarnos las imágenes de elasticsearch y kibana con los siguientes comandos:

```
$ docker pull docker.elastic.co/elasticsearch/elasticsearch:7.10.1
$ docker pull docker.elastic.co/kibana/kibana:7.10.1
```

```
jorge@jorge-lafuente:~/Escritorio/conf_elastic$ docker pull docker.elastic.co/elasticsearch/elasticsearch:7.10.1
7.10.1: Pulling from elasticsearch/elasticsearch
Digest: sha256:5d8f1962907ef60746a8cf61c8a7f2b8755510ee36bdee0f65417f90a38a0139
Status: Image is up to date for docker.elastic.co/elasticsearch/elasticsearch:7.10.1
docker.elastic.co/elasticsearch/elasticsearch:7.10.1
jorge@jorge-lafuente:~/Escritorio/conf_elastic$ docker pull docker.elastic.co/kibana/kibana:7.10.1
7.10.1: Pulling from kibana/kibana
Digest: sha256:e6488b7cd973ece70903f5fd587e04810f9f5115ac31b35cb44569f0b408bb
Status: Image is up to date for docker.elastic.co/kibana/kibana:7.10.1
docker.elastic.co/kibana/kibana:7.10.1
jorge@jorge-lafuente:~/Escritorio/conf_elastic$ 
```

Figura 20: Descarga de imágenes

Como se ve en la imagen anterior, las imágenes ya están descargadas, ya las podemos utilizar.

Se van a desplegar 5 contenedores, 3 para los nodos de Elasticsearch, 1 para el servicio de Kibana y 1 para el proxy inverso HAProxy.

Antes de empezar a utilizarlas tenemos que definir los ficheros de configuración para cada contenedor.

Los ficheros de configuración de los nodos de Elasticsearch (elasticsearch.yml) son los siguientes:

Fichero `elasticsearch.yml` para el NODO 1

```
1 cluster.name: elastic_cluster
2 node.name: node-1
3 network.host: 0.0.0.0
4 discovery.seed_hosts: ["172.17.0.2", "172.17.0.3"
, "172.17.0.4"]
5 cluster.initial_master_nodes: ["node-1"]
6 discovery.zen.minimum_master_nodes: 2
7 node.roles: [master]
8 xpack.security.enabled: true
```

Fichero `elasticsearch.yml` para el NODO 2

```
1 cluster.name: elastic_cluster
2 node.name: node-2
3 network.host: 0.0.0.0
4 discovery.seed_hosts: ["172.17.0.2", "172.17.0.3"
, "172.17.0.4"]
5 cluster.initial_master_nodes: ["node-1"]
6 discovery.zen.minimum_master_nodes: 2
7 node.roles: [master]
8 xpack.security.enabled: true
```

Fichero `elasticsearch.yml` para el NODO 3

```
1 cluster.name: elastic_cluster
2 node.name: node-3
3 network.host: 0.0.0.0
```

```

4 discovery.seed_hosts: [ "172.17.0.2", "172.17.0.3"
, "172.17.0.4" ]
5 cluster.initial_master_nodes: [ "node-1" ]
6 discovery.zen.minimum_master_nodes: 2
7 node.roles: [ data , ingest ]
8 xpack.security.enabled: true

```

A continuación creamos o descargamos el fichero `jvm.options` con el siguiente contenido:

```

1 # Xms represents the initial size of total heap space
2 # Xmx represents the maximum size of total heap space
3
4 -Xms1g
5 -Xmx1g

```

Como se ve en los ficheros de configuración de los nodos, el nodo 1 y nodo 2 son dos nodos dedicados master-eligible (mediante `node.roles: [master]`) y el nodo 3 es un nodo dedicado a data e ingest. También se ve que se ha activado el modulo de seguridad (mediante `xpack.security.enabled: true`) de elastic para que haya autenticación obligatoria. Se ha establecido un mínimo de dos nodos master para garantizar la alta disponibilidad.

Antes de lanzar los contenedores es necesario aumentar el máximo de memoria virtual `vm.max_map_count`.

```
$ sudo sysctl -w vm.max_map_count=262144
```

Para empezar a lanzar los nodos tenemos que levantar un contenedor para cada uno de los nodos. Levantamos el nodo 1:

```

$ docker run -d \
--volume="/home/jorge/conf_elastic/elasticsearch.yml:/usr/share/
elasticsearch/config/elasticsearch.yml" \
--volume="/home/jorge/conf_elastic/jvm.options:/usr/share/
elasticsearch/config/jvm.options" \
--name elasticsearch \
docker.elastic.co/elasticsearch/elasticsearch:7.10.1

```

```

→ docker run \
--volume="/home/kevin/master/1c/sist_inf_ciberseg/elastic_files/el
asticsearch_1.yml:/usr/share/elasticsearch/config/elasticsearch.ym
l" \
--volume="/home/kevin/master/1c/sist_inf_ciberseg/elastic_files/jv
m.options:/usr/share/elasticsearch/config/jvm.options" docker.elas
tic.co/elasticsearch/elasticsearch:7.10.1

```

Figura 21: Inicio del nodo 1

Levantamos el nodo 2:

```
$ docker run -d \
--volume="/home/jorge/conf_elastic/elasticsearch2.yml:/usr/share/
elasticsearch/config/elasticsearch.yml" \
--volume="/home/jorge/conf_elastic/jvm.options:/usr/share/
elasticsearch/config/jvm.options" \
--name elasticsearch2 \
docker.elastic.co/elasticsearch/elasticsearch:7.10.1
```

```
jorge@jorge-lafuentec:~$ docker run -d --volume="/home/jorge/conf_elastic/elasticsearch2.yml:/usr/share/elasticsearch/config/elasticsearch.yml" \
--volume="/home/jorge/conf_elastic/jvm.options:/usr/share/elasticsearch/config/jvm.options" --name elasticsearch2 docker.elastic.co/elasticsearch
/6b9661ec1068bf71c04f8c429e62e96bf313f9066b50f891bf2f7e4cd06794
```

Figura 22: Inicio del nodo 2

Levantamos el nodo 3:

```
$ docker run -d \
--volume="/home/jorge/conf_elastic/elasticsearch3.yml:/usr/share/
elasticsearch/config/elasticsearch.yml" \
--volume="/home/jorge/conf_elastic/jvm.options:/usr/share/
elasticsearch/config/jvm.options" \
--name elasticsearch3 \
docker.elastic.co/elasticsearch/elasticsearch:7.10.1
```

```
jorge@jorge-lafuentec:~$ docker run -d --volume="/home/jorge/conf_elastic/elasticsearch3.yml:/usr/share/elasticsearch/config/elasticsearch.yml" \
--volume="/home/jorge/conf_elastic/jvm.options:/usr/share/elasticsearch/config/jvm.options" --name elasticsearch3 docker.elastic.co/elasticsearch
/7ad6ede7542cf8a1bb8d9874dd82f66f68d96cea563268613c9cab9f1fb02c7c
```

Figura 23: Inicio del nodo 3

Como vemos, los tres contenedores hacen uso de un volumen local que son los ficheros de configuracion necesarios. En cuanto al fichero **elasticsearch.yml** de cada uno de los nodos es diferente, como se ha descrito anteriormente.

La opción **-d** hace que el contendor se ejecute en segundo plano. Si queremos ver los logs del contenedor podemos utilizar el comando **\$ docker logs <Nombre del contendor>**.

Antes de acceder al cluster tenemos que configurar las credenciales de los usuarios para que nos podamos comunicar con el cluster, ya que se ha habilitado el módulo de seguridad **xpack.security.enabled**.

Para ello, tenemos que entrar en uno de los contenedores, por ejemplo el nodo 1:

```
$ docker exec -it elasticsearch /bin/bash
```

Una vez dentro, ejecutamos el siguiente comando para establecer las contraseñas de los usuarios creados por defecto:

```
# bin/elasticsearch-setup-passwords interactive
```

```
jorge@jorge-lafuente:~$ docker exec -it elasticsearch /bin/bash
[root@b443123bbc44 elasticsearch]# bin/elasticsearch-setup-passwords interactive
Initiating the setup of passwords for reserved users elastic,apm_system,kibana_system,logstash_system,beats_system,remote_monitoring_user.
You will be prompted to enter passwords as the process progresses.
Please confirm that you would like to continue [y/N]y

Enter password for [elastic];
Reenter password for [elastic];
Enter password for [apm_system];
Reenter password for [apm_system];
Enter password for [kibana_system];
Reenter password for [kibana_system];
Enter password for [logstash_system];
Reenter password for [logstash_system];
Enter password for [beats_system];
Reenter password for [beats_system];
Enter password for [remote_monitoring_user];
Reenter password for [remote_monitoring_user];
Changed password for user [apm_system]
Changed password for user [kibana_system]
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [remote_monitoring_user]
Changed password for user [elastic]
```

Figura 24: Configuración usuarios de elasticsearch

Una vez creadas la credenciales ya podemos operar con el clúster.

3.3. HAProxy

Una vez que tenemos los tres nodos de elasticsearch levantados, ya se podría empezar a operar con el cluster, pero como son contenedores, hay que exponer el puerto de comunicación con el cluster, que por defecto es el puerto 9200. Para ello, para dotar de mayor seguridad al cluster, para proporcionar alta disponibilidad y para proporcionar **balanceo de carga** vamos a levantar un contenedor con un **proxy inverso: HAProxy**. Para ello, descargamos la imagen de HAProxy:

```
$ docker pull haproxy:1.7
```

```
jorge@jorge-lafuente:~$ docker pull haproxy:1.7
1.7: Pulling from library/haproxy
6ec7b7d162b2: Pull complete
b3d2c97113d6: Pull complete
8c08610d1eaa: Pull complete
8b8d44187533: Pull complete
e5e17b0cf2aa: Pull complete
Digest: sha256:c958006a00d759996472218df80a33d71a62bfe54b2554c487763cb9b07ee0de
Status: Downloaded newer image for haproxy:1.7
docker.io/library/haproxy:1.7
```

Figura 25: Descarga de imagen HAProxy

Para ejecutar el contenedor, tenemos que crear un fichero de configuración previo (**haproxy.cfg**):

```
1 #-----  
2 # Global settings  
3 #-----  
4 global  
5     maxconn    4000  
6     log        127.0.0.1 local0  
7     log        127.0.0.1 local1 notice  
8     daemon  
9 defaults  
10    log        global  
11    mode       http  
12    retries   3  
13    timeout   client 50s  
14    timeout   connect 5s  
15    timeout   server 15s  
16    timeout   http-keep-alive 10s  
17  
18 frontend.elasticsearch-9200  
19     bind 0.0.0.0:9200  
20     default_backend.elasticsearch-9200  
21  
22 backend.elasticsearch-9200  
23     option tcpka  
24     option tcplog  
25     option redispatch  
26     option httpchk GET / HTTP/1.0\r\nAuthorization:\ Basic<PASSWORD>  
27     timeout server 15m  
28     balance      roundrobin  
29     server       node-1 172.17.0.2:9200 check  
30     server       node-2 172.17.0.3:9200 check  
31     server       node-3 172.17.0.4:9200 check  
32  
33 # monitor port  
34 listen.status  
35     bind 0.0.0.0:8088  
36     stats enable  
37     stats uri /
```

Donde pone <PASSWORD> habría que introducir las credenciales para comunicarse con el clúster de elasticsearch.

Las credenciales se tienen que incluir codificadas en **base64** de la siguiente forma: usuario:contraseña y codificado en base64.

Como se ve en el fichero, se establece una interfaz de comunicación con el proxy (frontend elasticsearch-9200) que se comunica con el cluster de elasticsearch por el puerto 9200 interno de cada contenedor. También vemos que expone su puerto interno 9200 para la comunicación con la interfaz, que después habrá que conectar con el puerto externo 9200 para que los clientes se puedan comunicar con el cluster.

También establece un puerto para ver estadísticas de uso del proxy en tiempo real.

Levantamos el proxy:

```
$ docker run -p 9200:9200 -p 8088:8088 -v
/home/jorge/Escritorio/conf_haproxy/haproxy.cfg:/usr/local/
etc/haproxy/haproxy.cfg haproxy:1.7
```

```
1c/sist_inf_ciberseg/elastic_files took 1h 1m 33s
→ docker run -p 9200:9200 -p 8088:8088 -v /home/kevin/master/1c/
sist_inf_ciberseg/elastic_files/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg haproxy:1.7
<7>haproxy-systemd-wrapper: executing /usr/local/sbin/haproxy -p
/run/haproxy.pid -db -f /usr/local/etc/haproxy/haproxy.cfg -Ds
```

Figura 26: Inicio del contenedor HAProxy

Como vemos, se expone el puerto 9200, que es el puerto por el cual los clientes se van a comunicar con el cluster de elasticsearch a través del proxy y el puerto 8088 en el que podemos ver las estadísticas del proxy.

A través del puerto 8088 se puede ver el estado de las conexiones del proxy inverso con el cluster de elastic:

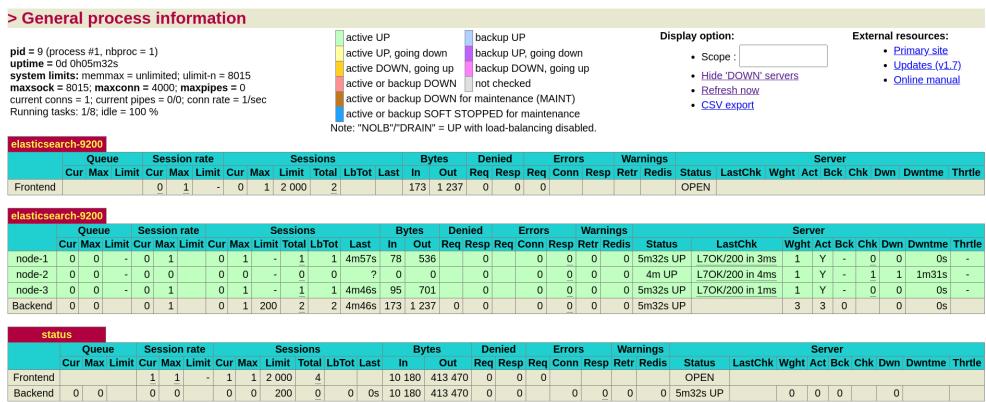


Figura 27: Panel de estado de HAProxy

Ya podemos ver el estado del cluster gracias a la API que tiene elasticsearch:

```
$ curl -XGET localhost:9200/_cat/nodes?v
```

```
jorge@jorge-lafuente:~$ curl localhost:9200/_cat/nodes?pretty
{
  "error" : {
    "root_cause" : [
      {
        "type" : "security_exception",
        "reason" : "missing authentication credentials for REST request [/_cat/nodes?pretty]",
        "header" : {
          "WWW-Authenticate" : "Basic realm=\"security\" charset=\"UTF-8\""
        }
      }
    ],
    "type" : "security_exception",
    "reason" : "missing authentication credentials for REST request [/_cat/nodes?pretty]",
    "header" : {
      "WWW-Authenticate" : "Basic realm=\"security\" charset=\"UTF-8\""
    }
  },
  "status" : 401
}
```

Figura 28: Error de autenticación

Pero no podemos ver aún el estado de los nodo porque hemos habilitado el modulo de seguridad de elasticsearch que hace obligatoria la autenticación.

Para poder ver el estado del cluster tenemos que introducir el usuario con el que autenticarnos:

```
→ curl -u elastic "localhost:9200/_cat/nodes?pretty"
Enter host password for user 'elastic':
172.17.0.3 35 75 7 0.58 0.91 0.74 m - node-2
172.17.0.4 29 75 7 0.58 0.91 0.74 di - node-3
172.17.0.2 48 75 7 0.58 0.91 0.74 m * node-1
```

Figura 29: Estado de los nodos

Como se ve en la imagen, el nodo 1 y el nodo 2 son nodos dedicados master-eligible y el nodo 3 es un nodo dedicado a data e ingest.

3.4. Kibana

Ya tenemos creada toda las estructura de elasticsearch preparada, ya podemos levantar el contenedor de kibana para interactuar con el cluster de elastic.

Para ello, vamos a ver el fichero de configuración de kibana para comunicarse con el cluster (**kibana.yml**):

```

1 server.name: kibana
2 server.host: "0"
3 elasticsearch.hosts: [ "http://elasticsearch:9200"
4   ]
4 monitoring.ui.container.elasticsearch.enabled:
  true
5 elasticsearch.username: "kibana"
6 elasticsearch.password: "password"

```

Vemos que se conecta con el servicio "http://elasticsearch:9200", pero para eso, ambos contenedores se tienen que conocer, por lo que al levantar el contenedor se lo indicaremos (es decir, en el fichero /etc/hosts del contenedor de kibana tiene que haber una entrada para comunicarse con el cluster de elasticsearch).

También se incluyen las credenciales para que kibana pueda interactuar con el cluster.

Levantamos el contenedor:

```

$ docker run -d --link elasticsearch:elasticsearch -p 5601:5601
--volume="/home/jorge/Escritorio/conf_kibana/config/kibana.yml:/usr/
share/kibana/config/kibana.yml" --name kibana
docker.elastic.co/kibana/kibana:7.10.1

```

```
jorge@jorge-lafuente:~$ docker run -d --link elasticsearch:elasticsearch -p 5601:5601 --volume="/home/jorge/Escritorio/conf_kibana/config/kibana
.yml:/usr/share/kibana/config/kibana.yml" --name kibana docker.elastic.co/kibana/kibana:7.10.1
77ac17aca36ba845099a6f2a93063e101dcb98854cfb34130e54c590b24ba5f5

```

Figura 30: Inicio del contenedor de kibana

Como vemos, hay que indicarle el fichero de configuración y con la opción **-link** conectamos con el cluster de elasticsearch.

Ya podemos entrar a la aplicación web de kibana esperando un poco y accediendo al siguiente enlace:

<http://localhost:5601>

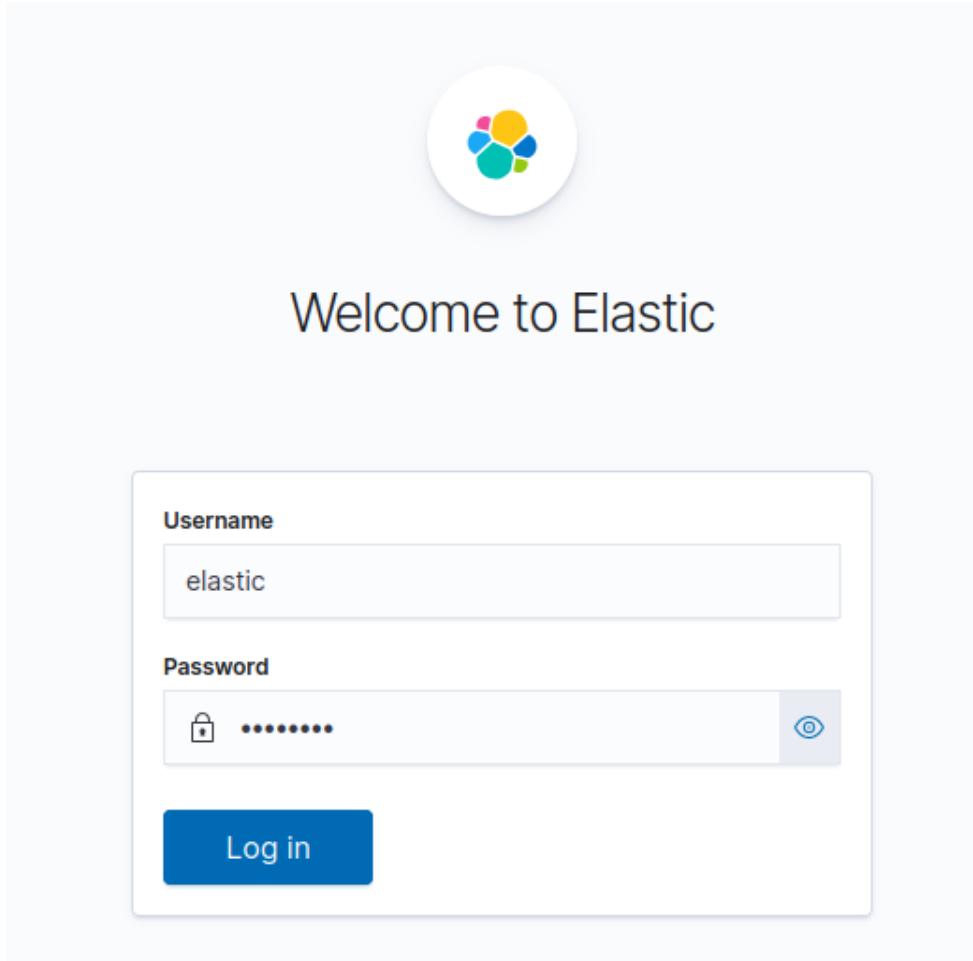


Figura 31: Panel inicio kibana

Nos logueamos con el usuario elastic en kibana. Podemos mirar el estado de los nodos a través de la herramienta **Dev Tools**:

	ip	heap.percent	ram.percent	cpu	load_1m	load_5m	load_15m	node.role	master	name
1	172.17.0.2	52	96	34	2.73	1.74	0.91	n	*	node-1
2	172.17.0.3	21	96	34	2.73	1.74	0.91	n	-	node-2
3	172.17.0.4	48	96	34	2.73	1.74	0.91	dt	-	node-3
4										
5										

Figura 32: Estado de los nodos en kibana

Ya tenemos la estructura de elasticsearch preparada para empezar a recoger y analizar datos de cualquier fuente.

3.5. Docker-compose

Ya hemos visto como crear la infraestructura necesaria para crear el cluster de elasticseach, utilizar el proxy inverso HAProxy para ayudar con el balanceo de carga y la alta disponibilidad del cluster de elasticsearh y por último, la instalación y configuración de kibana para el tratamiento y gestión del cluster, todo en base a contenedores docker.

Ya que toda la infraestructura está basada en contenedores docker, hemos decidido crear un fichero **docker-compose** (docker-compose.yml) con el que desplegar toda la infraestructura con un solo comando, a falta de unas configuraciones de seguridad, que habrá que hacer una vez creados todos los contenedores.

Para empezar, tenemos que instalar docker-compose (estos son los comandos necesarios para instalar docker-compose en una máquina **Ubuntu 20.04**):

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)"  
-o /usr/local/bin/docker-compose  
  
$ sudo chmod +x /usr/local/bin/docker-compose
```

Una vez que tenemos instalado docker-compose en la máquina, ya podemos crear el fichero yml para desplegar la infraestructura, pero antes hay que crear una red de docker para poder configurar adecuadamente los ficheros de configuración de los nodos de elasticsearch. Para ello, creamos la red **jorge_default** con el siguiente comando:

```
$ docker network create -d bridge jorge_default
```

Una vez creada, revisamos la dirección IP de la red para despues trasladarla a los ficheros de configuración:

```
jorge@jorge-lafuente:~$ docker inspect jorge_default
[{"Name": "jorge_default",
 "Id": "6fee144ea413c60899078e29483091581a9cc3d0fa446acf885b42d53909e4a7",
 "Created": "2021-01-06T14:10:21.674234424+01:00",
 "Scope": "local",
 "Driver": "bridge",
 "EnableIPv6": false,
 "IPAM": {
     "Driver": "default",
     "Options": null,
     "Config": [
         {
             "Subnet": "172.20.0.0/16",
             "Gateway": "172.20.0.1"
         }
     ]
 },
 "Internal": false,
 "Attachable": true,
 "Ingress": false,
 "ConfigFrom": {
     "Network": ""
 },
 "ConfigOnly": false,
 "Containers": {},
 "Options": {},
 "Labels": {
     "com.docker.compose.network": "default",
     "com.docker.compose.project": "jorge",
     "com.docker.compose.version": "1.26.0"
 }
}]
```

Figura 33: Red ‘jorge_default’ en docker

Vemos que la red tiene la IP **172.20.0.0** por lo que tendremos que configurar los ficheros elasticsearch.yml con las direcciones IP correspondientes a esta red.

Con la red ya creada, podemos crear el fichero **docker-compose.yml**:

```
1 version: '2.2'
2 services:
3   elasticsearch:
4     image: elasticsearch:7.10.1
5     container_name: elasticsearch
6     volumes:
7       - <ruta_absoluta_fichero>/elasticsearch .
8         yml:/usr/share/elasticsearch/config/
9           elasticsearch.yml
10        - <ruta_absoluta_fichero>/jvm.options:/usr/
11          share/elasticsearch/config/jvm.options
12
13   elasticsearch2:
```

```

11   image: elasticsearch:7.10.1
12   container_name: elasticsearch2
13   volumes:
14     - <ruta_absoluta_fichero>/elasticsearch2.
15       yml:/usr/share/elasticsearch/config/
16         elasticsearch.yml
17   - <ruta_absoluta_fichero>/jvm.options:/usr/
18     share/elasticsearch/config/jvm.options
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

elasticsearch2:

- image:** elasticsearch:7.10.1
- container_name:** elasticsearch2
- volumes:**
 - <ruta_absoluta_fichero>/elasticsearch2.
 - yml:/usr/share/elasticsearch/config/
 - elasticsearch.yml
- <ruta_absoluta_fichero>/jvm.options:/usr/
- share/elasticsearch/config/jvm.options

elasticsearch3:

- image:** elasticsearch:7.10.1
- container_name:** elasticsearch3
- volumes:**
 - <ruta_absoluta_fichero>/elasticsearch3.
 - yml:/usr/share/elasticsearch/config/
 - elasticsearch.yml
- <ruta_absoluta_fichero>/jvm.options:/usr/
- share/elasticsearch/config/jvm.options

haproxy:

- image:** haproxy:1.7
- container_name:** haproxy
- links:**
 - elasticsearch
 - elasticsearch2
 - elasticsearch3
- volumes:**
 - <ruta_absoluta_fichero>/haproxy.cfg:/usr/
 - local/etc/haproxy/haproxy.cfg
- ports:**
 - "9200:9200"
 - "8088:8088"

kibana:

- image:** kibana:7.10.1
- container_name:** kibana
- links:**
 - elasticsearch
 - elasticsearch2
 - elasticsearch3
- volumes:**
 - <ruta_absoluta_fichero>/kibana.yml:/usr/
 - share/kibana/config/kibana.yml
- ports:**
 - "5601:5601"

```

47 networks:
48   default:
49     external:
50       name: <nombre_red>

```

En este documento habría que introducir los directorios donde se guarden los ficheros de configuración de los nodos de elasticsearch, haproxy y kibana. Estos ficheros de configuración se han descrito anteriormente. También habría que introducir el nombre de la red de docker que hemos creado anteriormente.

Con este fichero de configuración de docker-compose, que se llama **docker-compose** ya solo necesitamos ir con una terminal al directorio donde guardamos dicho fichero y ejecutar el siguiente comando:

```
$ docker-compose up
```

```
jorge@jorge-lafuente:~$ docker-compose up
Creating elasticsearch2 ... done
Creating elasticsearch ... done
Creating elasticsearch3 ... done
Creating kibana ... done
Creating haproxy ... done
Attaching to elasticsearch, elasticsearch2, elasticsearch3, haproxy, kibana
haproxy | <7>haproxy-systemd-wrapper: executing /usr/local/sbin/haproxy -p /run/haproxy.pid -db -f /usr/local/etc/haproxy/haproxy.conf -Ds
elasticsearch | {"type": "server", "timestamp": "2021-01-06T13:13:03,306Z", "level": "INFO", "component": "o.e.n.Node", "cluster.name": "elasticsearch_cluster", "node.name": "node-1", "message": "version[7.10.1], pid[7], build[default/docker/1c34507e66d7db1211f66f3513706fdf548736aa/2020-12-05T01:00:33.671820Z], OS[Linux/5.4.0-59-generic/amd64], JVM[AdoptOpenJDK/OpenJDK 64-Bit Server VM/15.0.1/15.0.1+9]" }
elasticsearch | {"type": "server", "timestamp": "2021-01-06T13:13:03,324Z", "level": "INFO", "component": "o.e.n.Node", "cluster.name": "elasticsearch_cluster", "node.name": "node-1", "message": "JVM home [/usr/share/elasticsearch/jdk], using bundled JDK [true]" }
elasticsearch | {"type": "server", "timestamp": "2021-01-06T13:13:03,325Z", "level": "INFO", "component": "o.e.n.Node", "cluster.name": "elasticsearch_cluster", "node.name": "node-1", "message": "JVM arguments [-Xshare:auto, -Des.networkaddress.cache.ttl=60, -Des.networkaddress.cache.negative.ttl=10, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna.nosys=true, -XX:-OmitStackTraceInFastThrow" }
```

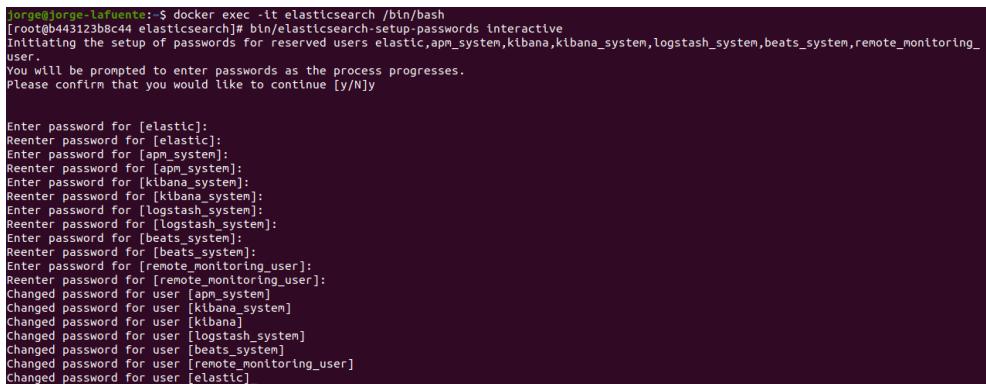
Figura 34: Inicio de docker-compose

En este momento ya tendríamos la infraestructura completa para empezar a configurarla. Solo faltaría configurar las credenciales de los usuarios del cluster de elasticsearch. Para ello, como se ha explicado anteriormente, lo primero es introducirse en uno de los contenedores de elasticsearch con el comando:

```
$ docker exec -it elasticsearch /bin/bash
```

Una vez dentro, ejecutamos el siguiente comando para establecer las contraseñas de los usuarios creados por defecto:

```
$ bin/elasticsearch-setup-passwords interactive
```



```
jorge@jorge-lafuente:~$ docker exec -it elasticsearch /bin/bash
[root@b443123bb8c44 elasticsearch]# bin/elasticsearch-setup-passwords interactive
Initiating the setup of passwords for reserved users elastic,apm_system,kibana,kibana_system,logstash_system,beats_system,remote_monitoring_user.
You will be prompted to enter passwords as the process progresses.
Please confirm that you would like to continue [y/N]

Enter password for [elastic]:
Reenter password for [elastic]:
Enter password for [apm_system]:
Reenter password for [apm_system]:
Enter password for [kibana_system]:
Reenter password for [kibana_system]:
Enter password for [logstash_system]:
Reenter password for [logstash_system]:
Enter password for [beats_system]:
Reenter password for [beats_system]:
Enter password for [remote_monitoring_user]:
Reenter password for [remote_monitoring_user]:
Changed password for user [apm_system]
Changed password for user [kibana_system]
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [remote_monitoring_user]
Changed password for user [elastic]
```

Figura 35: Configuración usuarios de elasticsearch

Recordemos que la credenciales que le pongamos a estos usuarios deben ser las mismas que se pusieron en el fichero **haproxy.cfg** y fichero **kibana.yml** para configurar ambos servicios.

Ya tendríamos configuradas las credenciales de los usuarios del cluster de elastic y podríamos empezar a operar con el cluster.

3.6. Auditbeat

El SIEM va a monitorizar un solo host, que es el host anfitrión donde están corriendo los contenedores. Además del host, el SIEM va a monitorizar la red a la que está conectada el host.

Para monitorizar todos los eventos del host, incluyendo los eventos de autenticación, se va a utilizar **Auditbeat**.

Auditbeat es una herramienta que monitoriza la actividad de los usuarios y los procesos, y analiza los datos de los eventos en la pila de elastic. Auditbeat se comunica directamente con el marco de auditoría de Linux, recoge los mismos datos que auditd, y envía los eventos a la Elastic Stack en tiempo real.

Auditbeat permite mirar cuidadosamente los directorios para cualquier sistema. Los cambios en los ficheros y directorios se envían en tiempo real a Elasticsearch, cada mensaje contiene metadatos y hash criptográficos del contenido del archivo para su posterior análisis.

Primero nos descargamos auditbeat:

```
$ curl -L -O https://artifacts.elastic.co/downloads/beats/
auditbeat/auditbeat-oss-7.8.1-amd64.deb
```

```
jorge@jorge-lafuente:~/auditbeat$ curl -L -O https://artifacts.elastic.
co/downloads/beats/auditbeat/auditbeat-7.10.1-amd64.deb
% Total    % Received % Xferd  Average Speed   Time     Time     Time
Current                                         Dload  Upload   Total   Spent   Left
Speed
0     0      0      0      0      0      0      0 ---:---:---:---:---:---:---:---:
0     0      0      0      0      0      0      0 ---:---:---:---:---:---:---:---:
3 26.8M   3 926k   0      0    748k      0 0:00:36  0:00:01  0:00:3
11 26.8M  11 3205k   0      0   1429k      0 0:00:19  0:00:02  0:00:1
20 26.8M  20 5589k   0      0   1727k      0 0:00:15  0:00:03  0:00:1
31 26.8M  31 8677k   0      0   2048k      0 0:00:13  0:00:04  0:00:0
44 26.8M  44 11.9M   0      0   2338k      0 0:00:11  0:00:05  0:00:0
57 26.8M  57 15.5M   0      0   2548k      0 0:00:10  0:00:06  0:00:0
73 26.8M  73 19.6M   0      0   2786k      0 0:00:09  0:00:07  0:00:0
84 26.8M  84 22.7M   0      0   2825k      0 0:00:09  0:00:08  0:00:0
90 26.8M  90 24.4M   0      0   2709k      0 0:00:10  0:00:09  0:00:0
97 26.8M  97 26.0M   0      0   2610k      0 0:00:10  0:00:10  ---:---
100 26.8M 100 26.8M   0      0   2600k      0 0:00:10  0:00:10  ---:---
- 2675k
```

Figura 36: Descarga de auditbeat

Instalamos la herramienta en el host:

```
$ sudo dpkg -i auditbeat-oss-7.8.1-amd64.deb
```

```
jorge@jorge-lafuente:~/auditbeat$ sudo dpkg -i auditbeat-7.10.1-amd64.deb
[sudo] contraseña para jorge:
Seleccionando el paquete auditbeat previamente no seleccionado.
(Leyendo la base de datos ... 194174 ficheros o directorios instalados
actualmente.)
Preparando para desempaquetar auditbeat-7.10.1-amd64.deb ...
Desempaquetando auditbeat (7.10.1) ...
Configurando auditbeat (7.10.1) ...
Procesando disparadores para systemd (245.4-4ubuntu3.3) ...
```

Figura 37: Instalación de auditbeat

Una vez instalada la herramienta, tenemos que configurarla en su ficheros `/etc/auditbeat/auditbeat.yml`. En ellos hay que incluir las credenciales para kibana y elasticsearch y vamos a incluir más directorios para el análisis:

```
1 — module: file_integrity
2   paths:
3     - /bin
4     - /usr/bin
5     - /sbin
6     - /usr/sbin
7     - /etc
8     - /root
9     - /boot
10    - /lib
11
12 setup.kibana:
13   host: "localhost:5601"
14   username: "elastic"
15   password: "password"
16 output.elasticsearch:
17   hosts: ["localhost:9200"]
18   username: "elastic"
19   password: "password"
```

Además del módulo que se especifica en el documento anterior, el modulo de integridad de la estructura de directorios y ficheros, en el documento `auditbeat.yml` hay otros módulos para el análisis del sistema.

Una vez que tenemos configurada la herramienta para integrarse con elasticsearch y comunicarse con kibana, vamos a crear el índice y las visualizaciones en kibana con el siguiente comando:

```
$ sudo auditbeat setup
```

```
jorge@jorge-lafuente:~$ sudo auditbeat setup
Overwriting ILM policy is disabled. Set `setup.ilm.overwrite: true` for
enabling.

Index setup finished.
Loading dashboards (Kibana must be running and reachable)
Loaded dashboards
```

Figura 38: Creación de visualizaciones auditbeat

Ya podemos iniciar la ejecución de la herramienta:

```
$ sudo service auditbeat start
```

```
jorge@jorge-lafuente:~$ sudo service auditbeat start
jorge@jorge-lafuente:~$ sudo service auditbeat status
● auditbeat.service - Audit the activities of users and processes on y>
  Loaded: loaded (/lib/systemd/system/auditbeat.service; disabled; >
  Active: active (running) since Sun 2021-01-03 09:39:27 CET; 7s ago
    Docs: https://www.elastic.co/products/beats/auditbeat
   Main PID: 54020 (auditbeat)
     Tasks: 11 (limit: 9378)
    Memory: 116.4M
      CGroup: /system.slice/auditbeat.service
              └─54020 /usr/share/auditbeat/bin/auditbeat --environment >

ene 03 09:39:33 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:33.0>
ene 03 09:39:33 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:33.0>
ene 03 09:39:34 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:34.4>
ene 03 09:39:34 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:34.4>
ene 03 09:39:34 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:34.5>
ene 03 09:39:34 jorge-lafuente auditbeat[54020]: 2021-01-03T09:39:34.6>
lines 1-20/20 (END)
```

Figura 39: Inicio de la ejecución de auditbeat

Podemos comprobar que el sistema esta recibiendo los datos de manera correcta a través de kibana:

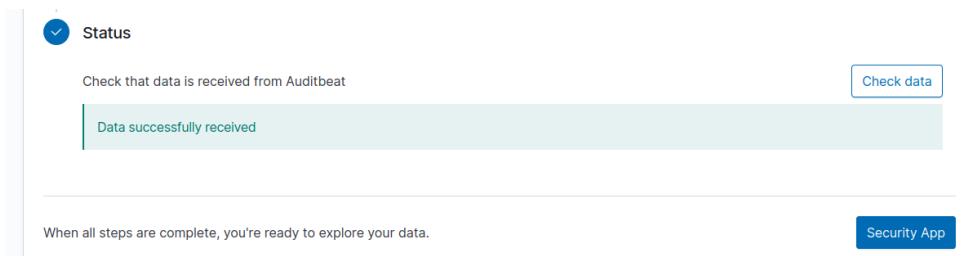


Figura 40: Comprobación de recepción de datos auditbeat

Ya se están capturando datos en Elasticsearch a través de Auditbeat. Para ello, la propia herramienta Auditbeat crea unas visualizaciones y unos dashboards para monitorizar los cambios y eventos en el sistema:

A screenshot of the Auditbeat Dashboards page. The title is "Dashboards" and there is a "Create dashboard" button. Below is a search bar and a table with 12 rows. The columns are "Title", "Description", and "Actions".

Title	Description	Actions
[Auditbeat Auditd] Executions ECS	Overview of kernel executions	
[Auditbeat Auditd] Overview ECS	Summary of Linux kernel audit events.	
[Auditbeat Auditd] Sockets ECS	Summary of socket related syscall events.	
[Auditbeat File Integrity] Overview ECS	Monitor file integrity events.	
[Auditbeat System] Host Dashboard ECS	System Hosts	
[Auditbeat System] Login Dashboard ECS	System Logins	
[Auditbeat System] Package Dashboard ECS	System Packages	
[Auditbeat System] Process Dashboard ECS	System Processes	
[Auditbeat System] Socket Dashboard ECS	System Sockets	
[Auditbeat System] System Overview ECS	Overview of System Information.	
[Auditbeat System] User Dashboard ECS	System Users	

Rows per page: 20 < 1 >

Figura 41: Dashboards de auditbeat

Con el dashboard **Overview** se puede ver un resumen de todo lo que monitoriza Auditbeat:

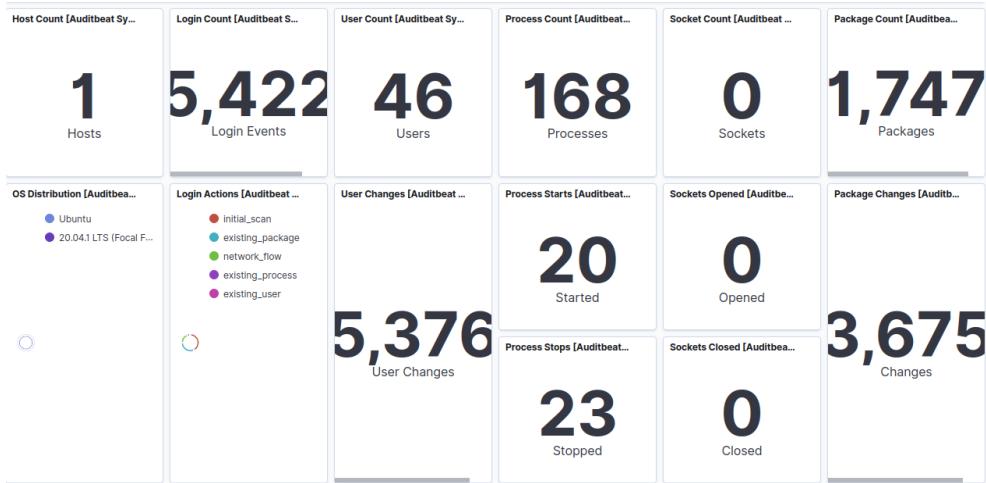


Figura 42: Dashboards overview de auditbeat

En el dashboard de **hosts** se puede ver datos relativos a los hosts que se estan monitorizando, En este caso solo se esta monitorizando uno:

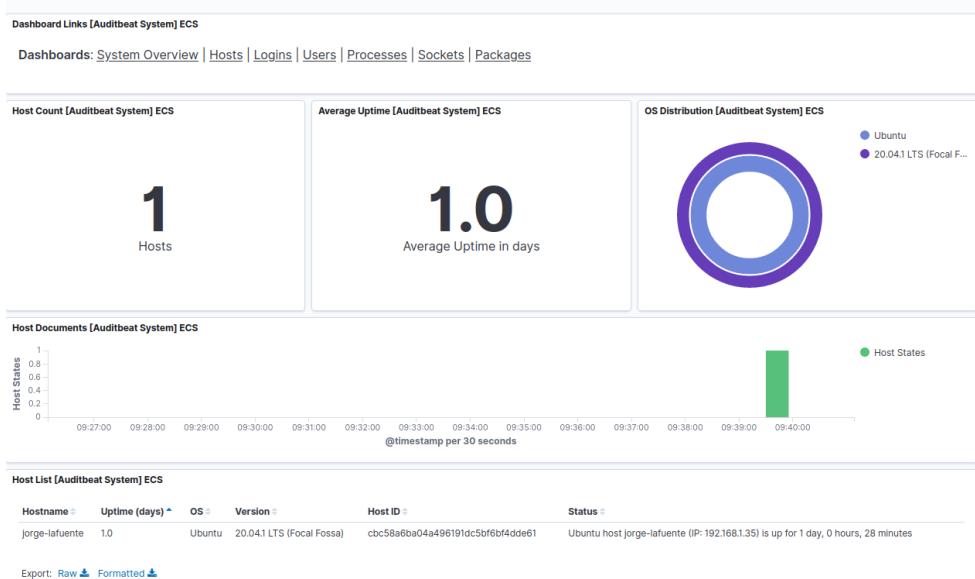


Figura 43: Dashboards hosts de auditbeat

Como hemos visto, hay muchos dashboards con los que se pueden monitorizar muchas aspectos del host pero nos vamos a centrar en los eventos de autenticación porque vamos a realizar un ataque de fuerza bruta contra el servicio **SSH** del host para entrar con el usuario "jorgez ver si con el dashboard de **Login** se puede detectar el ataque.

Realizamos el ataque de fuerza bruta contra el servicio ssh con la herramienta **hydra**:

```
$ hydra 192.168.1.35 ssh -l jorge -P rockyou.txt -f -vV
```

```
jorge@jorge-lafuente:~$ hydra 192.168.1.35 ssh -l jorge -P rockyou.txt
-f -vV
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-01-
03 19:26:33
[WARNING] Many SSH configurations limit the number of parallel tasks, it
is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 20 login tries (l:1
/p:20), ~2 tries per task
[DATA] attacking ssh://192.168.1.35:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://jorge@1
92.168.1.35:22
[INFO] Successful, password authentication is supported by ssh://192.16
8.1.35:22
[ATTEMPT] target 192.168.1.35 - Login "jorge" - pass "david" - 1 of 20
```

Figura 44: Ataque de fuerza bruta contra ssh

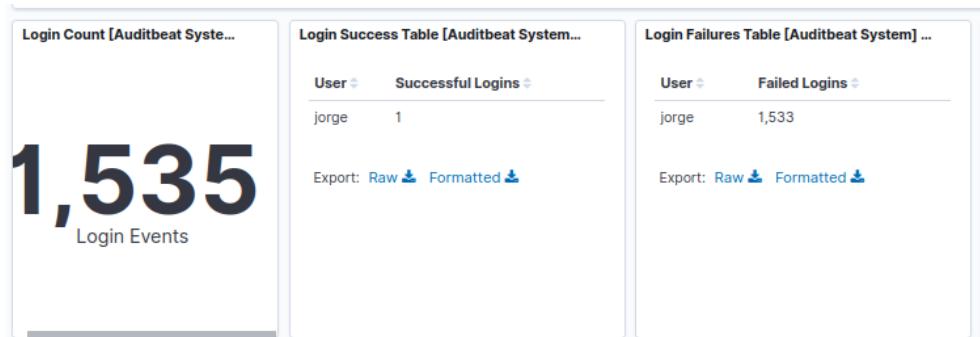


Figura 45: Dashboard logins

En el dashboard de **Logins** podemos ver que se han intentado logear más de 1500 veces con el usuario "jorge".entre ellas solo una ha sido fructifera. Por lo que podríamos entender que es un ataque de fuerza bruta. En el mismo dashboard, si miramos los eventos de intentos de acceso, vemos muchos intentos fallidos con el usuario "jorge":

Login Events [Auditbeat System] ECS					
Time	host.hostname	user.name	event.outcome	message	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	
> Jan 3, 2021 @ 19:39:24.000	jorge-lafuente	jorge	failure	Failed login by user jorge (UID: 1000) on ssh:notty	

Figura 46: Eventos de login en el dashboard de logins

3.7. Packetbeat

Ya estamos analizando datos relativos al host, ahora vamos a configurar una herramienta para analizar el tráfico de red y distintos servicios como servicios de bases de datos, rpc, etc. La herramienta que se va a utilizar es la herramienta **Packetbeat**. A través de esta herramienta vamos a poder monitorizar el tráfico DNS.

Packetbeat es un analizador de paquetes de red en tiempo real que se puede utilizar con Elasticsearch para proporcionar un sistema de monitorización de aplicaciones y análisis de rendimiento. Packetbeat completa la plataforma Beats proporcionando visibilidad entre los servidores de su red.

Packetbeat funciona capturando el tráfico de red entre sus servidores de aplicaciones, decodificando los protocolos de la capa de aplicación (HTTP, MySQL, Redis, etc.), correlacionando las solicitudes con las respuestas, y registrando los campos de interés para cada transacción. Actualmente, Packetbeat soporta los siguientes protocolos:

- ICMP (v4 and v6)
- DHCP (v4)
- DNS
- HTTP
- AMQP 0.9.1
- Cassandra
- Mysql
- PostgreSQL

- Redis
- Thrift-RPC
- MongoDB
- Memcache
- NFS
- TLS

Lo primero que debemos hacer es descargar e instalar la herramienta:

```
jorge@jorge-lafuente:~$ curl -L -O https://artifacts.elastic.co/downloads/beats/packetbeat/packetbeat-7.10.1-amd64.deb
          % Total    % Received % Xferd  Average Speed   Time     Time     Time
Current                                            Dload  Upload   Total  Spent  Left
Speed
0       0      0      0      0      0      0  --::--  --::--  --::--
0 26.5M  0      0      0      0      0      0  --::--  --::--  --::--
6 26.5M  6 1654k  0      0  1231k  0  0:00:22  0:00:01  0:00:2
19 26.5M 19 5337k  0      0  2271k  0  0:00:11  0:00:02  0:00:0
32 26.5M 32 8777k  0      0  2623k  0  0:00:10  0:00:03  0:00:0
37 26.5M 37 9.8M   0      0  2334k  0  0:00:11  0:00:04  0:00:0
46 26.5M 46 12.2M  0      0  2350k  0  0:00:11  0:00:05  0:00:0
60 26.5M 60 16.1M  0      0  2609k  0  0:00:10  0:00:06  0:00:0
69 26.5M 69 18.5M  0      0  2586k  0  0:00:10  0:00:07  0:00:0
74 26.5M 74 19.6M  0      0  2418k  0  0:00:11  0:00:08  0:00:0
75 26.5M 75 20.1M  0      0  2203k  0  0:00:12  0:00:09  0:00:0
77 26.5M 77 20.5M  0      0  2033k  0  0:00:13  0:00:10  0:00:0
78 26.5M 78 20.9M  0      0  1890k  0  0:00:14  0:00:11  0:00:0
80 26.5M 80 21.3M  0      0  1771k  0  0:00:15  0:00:12  0:00:0
81 26.5M 81 21.7M  0      0  1671k  0  0:00:16  0:00:13  0:00:0
83 26.5M 83 22.2M  0      0  1585k  0  0:00:17  0:00:14  0:00:0
85 26.5M 85 22.8M  0      0  1525k  0  0:00:17  0:00:15  0:00:0
90 26.5M 90 24.1M  0      0  1510k  0  0:00:18  0:00:16  0:00:0
95 26.5M 95 25.3M  0      0  1495k  0  0:00:18  0:00:17  0:00:0
98 26.5M 98 26.3M  0      0  1468k  0  0:00:18  0:00:18  --::-
100 26.5M 100 26.5M 0      0  1463k  0  0:00:18  0:00:18  --::-
- 1050k
jorge@jorge-lafuente:~$ sudo dpkg -i packetbeat-7.10.1-amd64.deb
Seleccionando el paquete packetbeat previamente no seleccionado.
(Leyendo la base de datos ... 194207 ficheros o directorios instalados
actualmente.)
Preparando para desempaquetar packetbeat-7.10.1-amd64.deb ...
Desempaquetando packetbeat (7.10.1) ...
Configurando packetbeat (7.10.1) ...
Procesando disparadores para systemd (245.4-4ubuntu3.3) ...
```

Figura 47: Instalación de Packetbeat

Una vez que esta descargada e instalada, vamos a crear los dashboards en kibana:

```
$ sudo packetbeat setup -e
```

```
jorge@jorge-lafuente:~$ sudo packetbeat setup -e
2021-01-03T10:48:49.012+0100    INFO    instance/beat.go:645    Home pa
th: [/usr/share/packetbeat] Config path: [/etc/packetbeat] Data path: [
/var/lib/packetbeat] Logs path: [/var/log/packetbeat]
2021-01-03T10:48:49.020+0100    INFO    instance/beat.go:653    Beat ID
: 31ec0e6c-4dcb-48da-92ac-232feb0b25ca
2021-01-03T10:48:49.071+0100    INFO    [beat] instance/beat.go:981 B
eat info {"system_info": {"beat": {"path": {"config": "/etc/pack
etbeat", "data": "/var/lib/packetbeat", "home": "/usr/share/packetbeat"
, "logs": "/var/log/packetbeat"}, "type": "packetbeat", "uuid": "31ec0e
6c-4dcb-48da-92ac-232feb0b25ca"}}}
2021-01-03T10:48:49.071+0100    INFO    [beat] instance/beat.go:990 B
uild info {"system_info": {"build": {"commit": "1da173a9e716715a7
a54bb3ff4db05b5c24fc8ce", "libbeat": "7.10.1", "time": "2020-12-04T22:4
9:30.000Z", "version": "7.10.1"}}}
2021-01-03T10:48:49.071+0100    INFO    [beat] instance/beat.go:993 G
o runtime info {"system_info": {"go": {"os": "linux", "arch": "amd64", "ma
x_procs": 4, "version": "go1.14.12"}}}
2021-01-03T10:48:49.074+0100    INFO    [beat] instance/beat.go:997 H
ost info {"system_info": {"host": {"architecture": "x86_64", "boot
_time": "2021-01-02T09:11:18+01:00", "containerized": false, "name": "jorge-"}}}
```

Figura 48: Creación de dashboards de Packetbeat

Ya podemos iniciar el servicio de packetbeat para que empiece a enviar datos a los nodos de elasticsearch:

```
jorge@jorge-lafuente:~$ sudo service packetbeat start
jorge@jorge-lafuente:~$ sudo service packetbeat status
● packetbeat.service - Packetbeat analyzes network traffic and sends t>
   Loaded: loaded (/lib/systemd/system/packetbeat.service; disabled;)
   Active: active (running) since Sun 2021-01-03 10:49:46 CET; 2s ago
     Docs: https://www.elastic.co/products/beats/packetbeat
   Main PID: 57927 (packetbeat)
      Tasks: 9 (limit: 9378)
     Memory: 35.9M
    CGroup: /system.slice/packetbeat.service
            └─57927 /usr/share/packetbeat/bin/packetbeat --environmen>

ene 03 10:49:46 jorge-lafuente packetbeat[57927]: 2021-01-03T10:49:46.>
lines 1-20/20 (END)
```

Figura 49: Inicio del servicio de Packetbeat

Podemos comprobar que se estan indexando datos de packetbeat en los nodos de eslasticsearch:

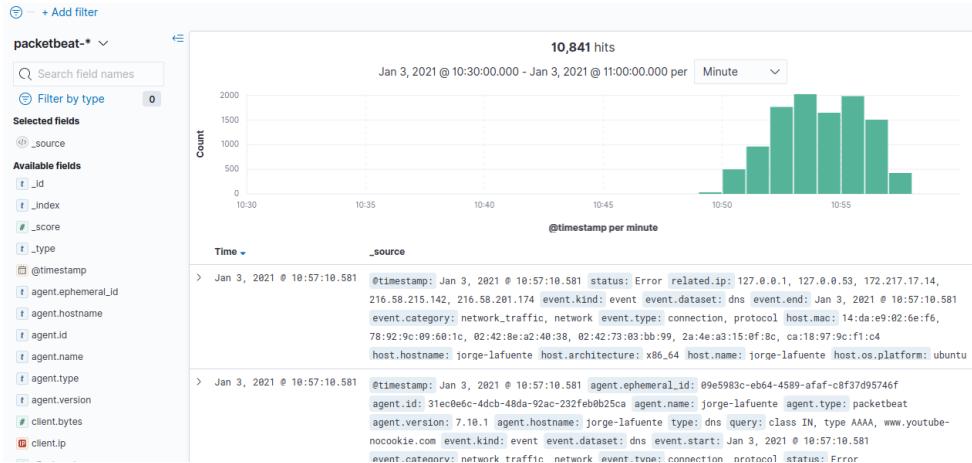


Figura 50: Comprobación Packetbeat

Podemos ver los dashboards que se han creado:

Dashboards			
<input type="text"/> packe		Create dashboard	
<input type="checkbox"/>	Title	Description	Actions
<input type="checkbox"/>	[Packetbeat] Overview ECS	Packetbeat overview dashboard.	
<input type="checkbox"/>	[Packetbeat] Cassandra ECS		
<input type="checkbox"/>	[Packetbeat] MongoDB ECS		
<input type="checkbox"/>	[Packetbeat] NFS ECS	NFSv3 and NFSv4 transactions over TCP.	
<input type="checkbox"/>	[Packetbeat] HTTP ECS		
<input type="checkbox"/>	[Packetbeat] Flows ECS		
<input type="checkbox"/>	[Packetbeat] DHCPv4 ECS	DHCPv4 Overview	
<input type="checkbox"/>	[Packetbeat] MySQL performance ECS		
<input type="checkbox"/>	[Packetbeat] DNS Overview ECS	Overview of DNS request and response metrics.	
<input type="checkbox"/>	[Packetbeat] DNS Tunneling ECS	Detecting tunneling over DNS.	
<input type="checkbox"/>	[Packetbeat] PgSQL performance ECS	Postgres database query performance.	
<input type="checkbox"/>	[Packetbeat] Thrift performance ECS		
<input type="checkbox"/>	[Packetbeat] TLS Sessions ECS	TLS Sessions ECS	

Rows per page: 20 < 1 >

Figura 51: Dashboards Packetbeat

Uno de los objetivos de la práctica era monitorizar el tráfico DNS y gracias a **Packetbeat** lo podemos hacer. Para monitorizar este tráfico, podemos utilizar el dashboard **DNS Overview** con el que se pueden consultar muchos datos acerca del tráfico DNS, como los servidores a los que se ha consultado, bytes transferidos, tipos de peticiones, etc:

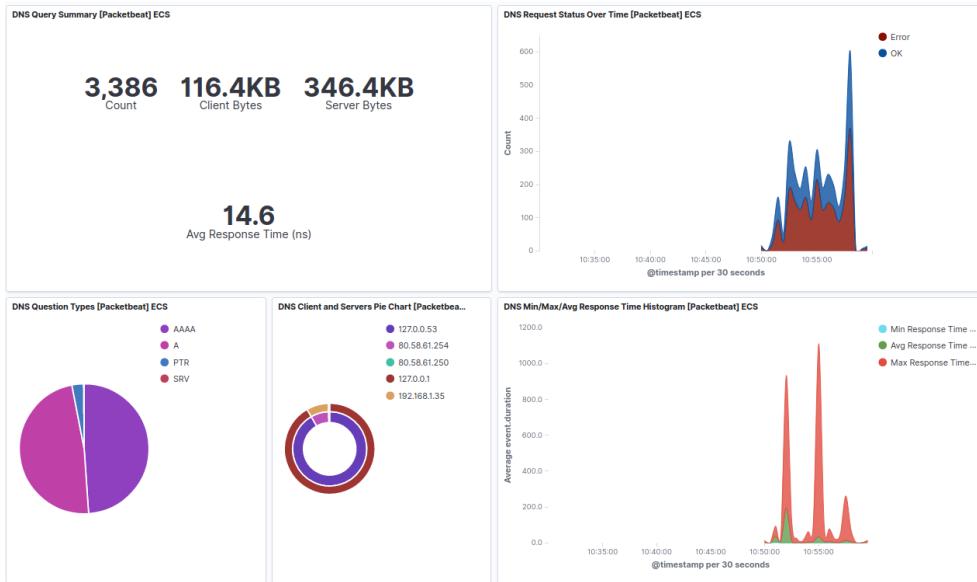


Figura 52: Dashboard DNS Overview Packetbeat

También el anterior dashboard se puede consultar el top 10 de servidores DNS consultados:

DNS Top 10 Questions [Packetbeat] ECS	
Question	Count
www.youtube-nocookie.com	748
www.youtube.com	28
www.google.com	22
r1---sn-h5qzen7d.googlevideo.com	20
s.ytimg.com	18
www.acb.com	18
acb.com	16
incoming.telemetry.mozilla.org	14
www.elastic.co	14
openmaptiles.org	12

Export: [Raw](#) [Formatted](#)

Figura 53: Top 10 servidores DNS consultados

También hay otro dashboard que se llama **DNS Tunneling** en el cual se pueden comprobar los FQDNs de los servidores DNS que se han consultado. Y

además hacer un ranking sobre ellos para encontrar posibles amenazas:

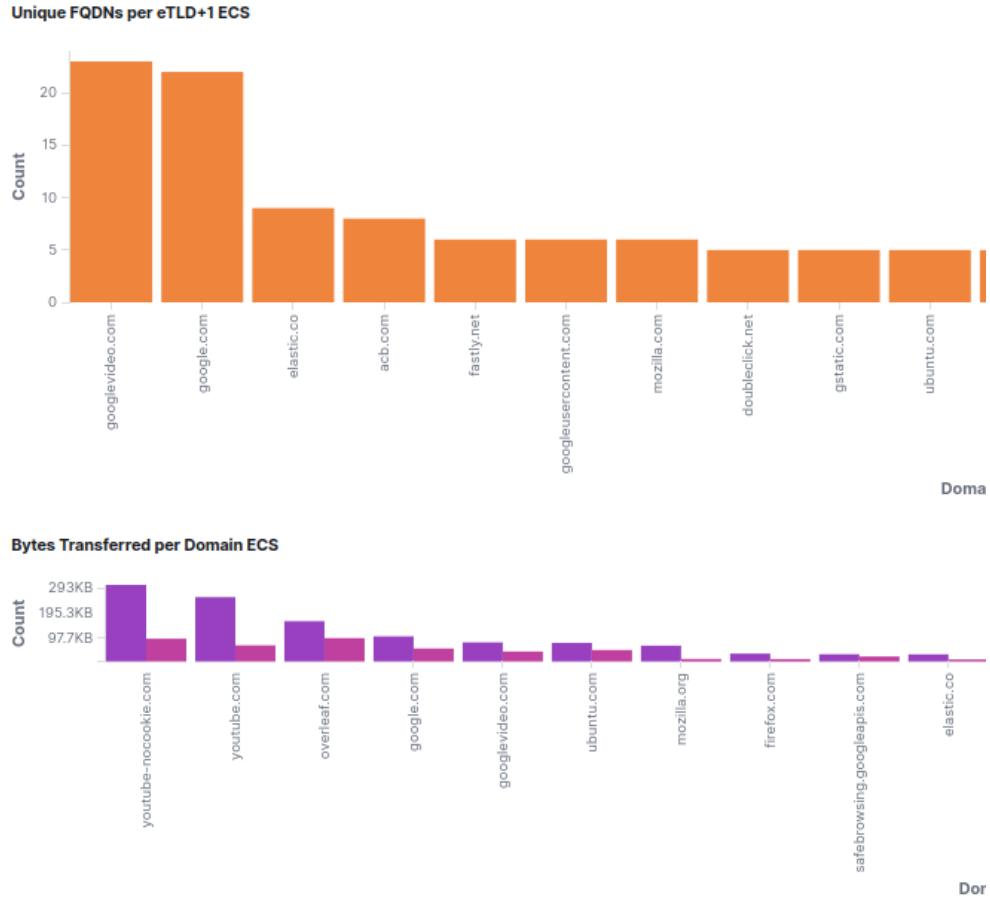


Figura 54: FQDNs únicos

ETLD+1	Count	Unique Domains
overleaf.com	3,108	4
youtube-nocookie.com	2,732	1
youtube.com	2,366	1
google.com	1,667	22
ubuntu.com	1,145	5
googlevideo.com	1,059	23
safebrowsing.googleapis.com	618	1
gstatic.com	444	5
firefox.com	401	1
elastic.co	379	9

Figura 55: Top 10 FQDNs únicos utilizados

3.8. Filebeat

Para finalizar la configuración del SIEM, se van a incluir los logs y alertas generadas por un IDS. El IDS que se va a utilizar es **Suricata**.

Suricata es un sistema de detección de intrusos (IDS) y un sistema de prevención de intrusos (IPS) de código abierto. Fue desarrollado por la Fundación de Seguridad Abierta (OISF).

Se va a instalar suricata en el host para monitorizar la actividad del mismo y de la red a la que esta conectado. Las alertas que vaya generando se van a guardar en elasticsearch a través de la herramienta **Filebeat** que se va a encargar de comunicar ambos sistemas.

Filebeat es un cargador ligero para el envío y la centralización de datos de registro. se uinstala un agente en el host y Filebeat monitoriza los archivos de registro o las ubicaciones que se especifiquen, recoge los eventos de registro y los envía a Elasticsearch o Logstash para su indexación.

Así es como funciona Filebeat: Cuando inicia Filebeat, inicia una o más entradas que buscan en las ubicaciones que ha especificado para los datos de registro. Por cada registro que Filebeat localiza, Filebeat inicia un harvester. Cada harvester lee un único registro para el nuevo contenido y envía los nuevos datos de registro a libbeat, que agrega los eventos y envía los datos agregados a la salida que ha configurado para Filebeat.

Lo que vamos a hacer es iniciar una entrada, la de suricata y Filebeat se va a encargar de llevar las alertas al cluster de elasticsearch.

Descargamos la herramienta Filebeat:

```
$ curl -L -0 https://artifacts.elastic.co/downloads/
beats/filebeat/filebeat-7.10.1-amd64.deb
```

```
jorge@jorge-lafuente:~/conf_filebeat$ curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.10.1-amd64.deb
% Total    % Received % Xferd  Average Speed   Time     Time     Time
Current                                         Dload  Upload   Total   Spent   Left
Speed
0      0    0      0    0      0      0  --::--  --::--  --::--
0 32.6M  0  6846    0      0  11885    0  0:48:03  --::--  0:48:0
5 32.6M  5 1947k   0      0  1308k    0  0:00:25  0:00:01  0:00:2
14 32.6M 14 4731k   0      0  1900k    0  0:00:17  0:00:02  0:00:1
25 32.6M 25 8523k   0      0  2422k    0  0:00:13  0:00:03  0:00:1
36 32.6M 36 12.0M   0      0  2743k    0  0:00:12  0:00:04  0:00:0
48 32.6M 48 15.9M   0      0  2963k    0  0:00:11  0:00:05  0:00:0
58 32.6M 58 19.1M   0      0  3025k    0  0:00:11  0:00:06  0:00:0
66 32.6M 66 21.6M   0      0  2960k    0  0:00:11  0:00:07  0:00:0
73 32.6M 73 23.9M   0      0  2886k    0  0:00:11  0:00:08  0:00:0
78 32.6M 78 25.6M   0      0  2773k    0  0:00:12  0:00:09  0:00:0
87 32.6M 87 28.4M   0      0  2779k    0  0:00:12  0:00:10  0:00:0
94 32.6M 94 30.9M   0      0  2757k    0  0:00:12  0:00:11  0:00:0
99 32.6M 99 32.6M   0      0  2675k    0  0:00:12  0:00:12  --::--
100 32.6M 100 32.6M  0      0  2674k    0  0:00:12  0:00:12  --::--
- 2226k
```

Figura 56: Descargar filebeat

Lo primero que hacemos es descargar **Filebeat**:

```
$ sudo dpkg -i filebeat-7.10.1-amd64.deb
```

```
jorge@jorge-lafuente:~/conf_filebeat$ sudo dpkg -i filebeat-7.10.1-amd64.deb
Seleccionando el paquete filebeat previamente no seleccionado.
(Leyendo la base de datos ... 196891 ficheros o directorios instalados
actualmente.)
Preparando para desempaquetar filebeat-7.10.1-amd64.deb ...
Desempaquetando filebeat (7.10.1) ...
Configurando filebeat (7.10.1) ...
Procesando disparadores para systemd (245.4-4ubuntu3.3) ...
```

Figura 57: Instalar filebeat

Una vez tenemos instalada la herramienta hay que configurarla para que pueda comunicarse con el cluster de elasticsearh y con kibana. Para ello, modificamos el fichero `/etc/filebeat/filebeat.yml` e incluimos las credenciales y las uri para conectar con el cluster:

```
1 setup.kibana:
2   host: "localhost:5601"
3   username: "elastic"
4   password: "password"
5
```

```
6 output.elasticsearch:  
7   hosts: ["localhost:9200"]  
8   username: "elastic"  
9   password: "password"
```

Una vez configurado para comunicarse con el cluster y con kibana, hay que configurarla para que extraiga las alertas de suricata para enviarselos. Para ello, tenemos que habilitar la entrada en filebeat para suricata:

```
$ sudo filebeat modules enable suricata
```

```
jorge@jorge-lafuente:~$ sudo filebeat modules enable suricata  
Enabled suricata
```

Figura 58: Habilitar entrada para suricata

Ya lo tenemos todo configurado para empezar a utilizar filebeat. Primero vamos a crear los dashboards para suricata:

```
$ sudo filebeat setup
```

```
jorge@jorge-lafuente:~$ sudo filebeat setup  
Overwriting ILM policy is disabled. Set `setup.ilm.overwrite: true` for  
enabling.  
  
Index setup finished.  
Loading dashboards (Kibana must be running and reachable)  
Loaded dashboards  
Setting up ML using setup --machine-learning is going to be removed in  
8.0.0. Please use the ML app instead.  
See more: https://www.elastic.co/guide/en/machine-learning/current/index.html  
Loaded machine learning job configurations  
Loaded Ingest pipelines
```

Figura 59: Habilitar entrada para suricata

Una vez instalada y actualizada la herramienta suricata, la iniciamos para empezar a monitorizar el host:

```
$ sudo suricata -c /etc/suricata/suricata.yml -i wlp3s0
```

```
jorge@jorge-lafuente:~$ sudo suricata -c /etc/suricata/suricata.yaml -i wlp3s0
3/1/2021 -- 16:50:00 - <Notice> - This is Suricata version 6.0.1 RELEASE running in SYSTEM mode
3/1/2021 -- 16:50:30 - <Warning> - [ERRCODE: SC_ERR_SYSCALL(50)] - Failure when trying to set feature via ioctl for 'wlp3s0': Operation not supported (95)
3/1/2021 -- 16:50:30 - <Notice> - all 4 packet processing threads, 4 management threads initialized, engine started.
```

Figura 60: Iniciamos suricata

Vemos que se empiezan a generar alertas:

```
$ tail -f /var/log/suricata/fast.log
```

```
jorge@jorge-lafuente:~$ tail -f /var/log/suricata/fast.log
01/03/2021-16:34:59.717510  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:34:59.817757  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:35:00.017865  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:35:00.418486  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:35:01.218895  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:35:02.819490  [**] [1:2016149:2] ET INFO Session Traversal Utilities for NAT (STUN Binding Request) [**] [Classification: Attempted User Privilege Gain] [Priority: 1] {UDP} 192.168.1.35:51373 -> 54.172.47.69:3478
01/03/2021-16:35:06.019681  [**] [1:2016149:2] ET INFO Session Traversa
```

Figura 61: Alertas suricata

Una vez que tenemos suricata activo, podemos activar filebeat para empezar a enviar los logs de las alertas:

```
$ sudo systemctl start filebeat.service
```

```
jorge@jorge-lafuente:~$ sudo service filebeat start
jorge@jorge-lafuente:~$ sudo service filebeat status
● filebeat.service - Filebeat sends log files to Logstash or directly >
  Loaded: loaded (/lib/systemd/system/filebeat.service; disabled; v>
  Active: active (running) since Sun 2021-01-03 17:09:33 CET; 1s ago
    Docs: https://www.elastic.co/products/beats/filebeat
 Main PID: 72384 (filebeat)
   Tasks: 10 (limit: 9378)
  Memory: 27.2M
    CGroup: /system.slice/filebeat.service
            └─72384 /usr/share/filebeat/bin/filebeat --environment sy>

ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.42>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.43>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.43>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.43>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.44>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.46>
ene 03 17:09:33 jorge-lafuente filebeat[72384]: 2021-01-03T17:09:33.46>
lines 1-20/20 (END)
```

Figura 62: Iniciamos filebeat

Podemos comprobar que se estan introduciendo datos en elasticsearch:

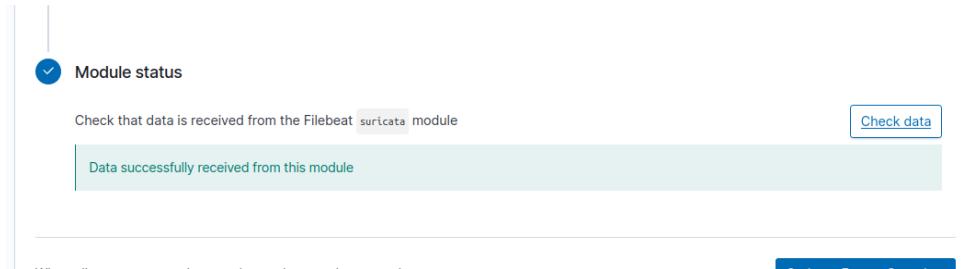


Figura 63: Comprobación filebeat

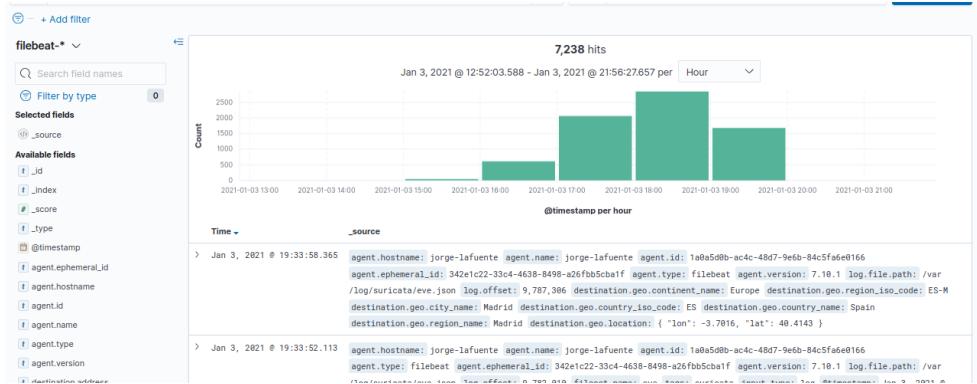


Figura 64: Ingesta de datos de filebeat

Para monitorizar las alertas se crean 2 dashboards, uno de eventos y otro de alertas. En el de eventos se pueden ver todos los eventos que se van generando en el log de suricata:



Figura 65: Dashboard de eventos de Suricata

Como se puede ver en la imagen anterior, en el dashboard se puede consultar el tipo de actividad que ha habido en los hosts, los protocolos que estan generando estos eventos, los paises origen y destino que estan involucrados en los eventos, etc. Además, se pueden ver por orden de aparicion, los eventos que va surgiendo en un log de eventos del dashboard:

Time	host.name	suricata.eve.flow_id	network.transport	source.ip	source.port	destination.ip	destination.port	destination.geo.region_name	destination.geo.country_name
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	514755342200050	tcp	192.168.43.133	80	120.116.249.42	49110	Taipei City	TW
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	1359182420078126	tcp	192.168.43.133	80	150.119.84.143	2144	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	1781948485984710	tcp	192.168.43.133	80	255.131.239.48	34124	-	-
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	1359184567562393	tcp	192.168.43.133	80	63.30.215.101	25775	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	1640059544145127	tcp	192.168.43.133	80	63.198.6.21	4215	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	1218447079035059	tcp	192.168.43.133	80	99.133.224.218	47952	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	514781784642190	tcp	192.168.43.133	80	169.29.36.230	7866	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	514763932602420	tcp	192.168.43.133	80	194.237.226.216	21126	-	SE
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	036976397526372	tcp	192.168.43.133	80	130.39.253.226	64984	Louisiana	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	2062878452353558	tcp	192.168.43.133	80	187.227.231.158	6081	Guerrero	MX
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	233291103294890	tcp	192.168.43.133	80	22.229.8.0	6711	-	US
> Jan 4, 2021 @ 00:22:03.206	jorge-lafuente	57065978451690630	tcp	192.168.43.133	80	528.183.229.68	45127	-	-

Figura 66: Log de eventos de Suricata

Por último, en el dashboard de alertas de suricata se pueden monitorizar todas las alertas que van apareciendo en suricata en base a las reglas definidas en suricata. Por ejemplo, en el dashboard se puede ver el top de alertas que han aparecido en suricata:

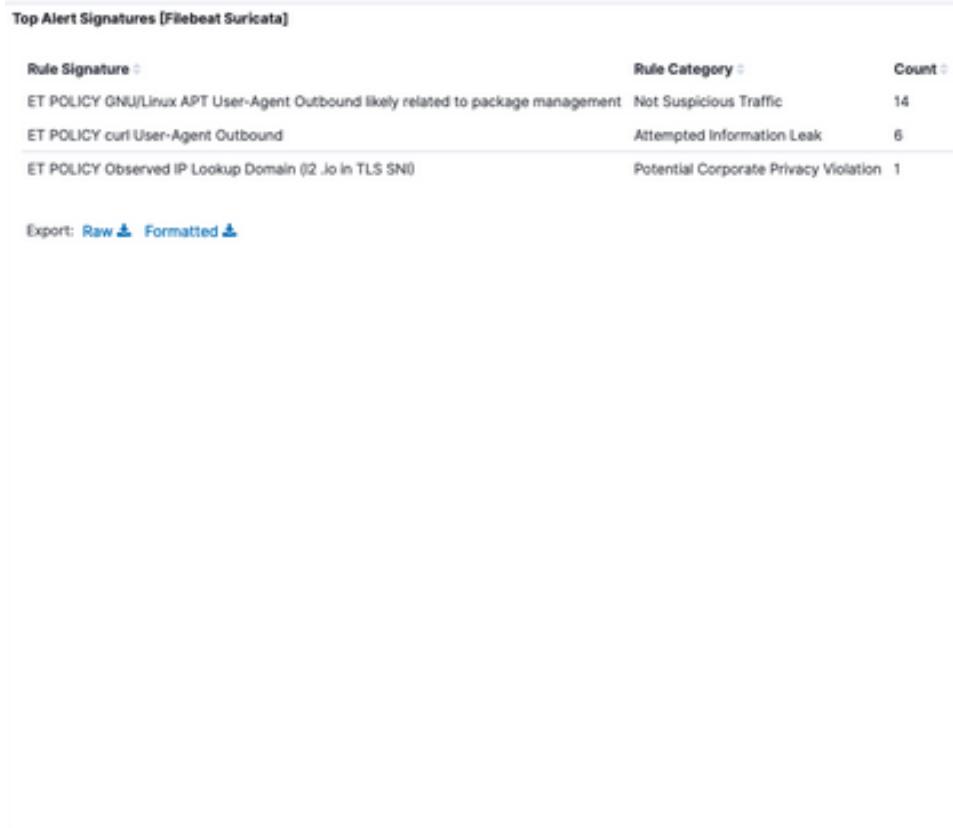


Figura 67: Top alertas de Suricata