Trabajo de Final de Grado

UNIVERSITAT DE BARCELONA

GRADO DE INFORMÁTICA
Facultad de Matemáticas e Informática
Universitat de Barcelona

DESARROLLO DE UN SIEM MEDIANTE ELK

Autor: Sergio Plans Acerete

Director:	**Prof. Raúl Roca Cánovas**
Realizado en:	**Facultad de Matemáticas e Informática**
Barcelona,	**27 de Junio de 2019**

Agradecimientos

Dado que este proyecto marca el final de una época, quiero agradecer no solo el apoyo que he recibido en el tiempo que ha durado este proyecto sino durante toda la carrera.

En primer lugar a mis padres, ya que sin ellos no habría podido estudiar una carrera y además de algo que me apasiona desde pequeño como es la informática.

Quiero agradecer en especial el apoyo de mi compañera Cristina Reina León ya que sin ella tanto este proyecto como estos cuatro años de carrera habrían resultado muchísimo más duros. Sin todas esas horas estudiando temarios y sin todos esos momentos de descanso hablando de muchas otras cosas, nada de esto habría sido posible. En la UB se ha iniciado una amistad que se mantendrá por muchos años.

También por supuesto agradezco el apoyo de todos mis amigos de toda la vida Walter Martín, Sergi Pica, Laura Martínez e Iris Bustos. Que con todo su apoyo y buenos momentos durante estos años me han ayudado a desconectar cuando era necesario y motivado a continuar cuando la motivación brillaba por su ausencia.

Índice

Capítulo 1
Introducción

1.1. Introducción

En la época en la que vivimos actualmente todo lo que nos rodea está de una manera u otra conectado a las redes, esto nos aporta grandes beneficios, pero, también es algo con lo que se debe tener cuidado, ya que un simple fallo en la seguridad de una red puede conllevar grandes problemas.

Por ello se han ideado muchos sistemas que evitan o intentan evitar todos estos posibles fallos de seguridad, desde los conocidos antivirus que los usuarios medios utilizan en sus equipos, a sistemas que detectan posibles fallos de seguridad en tiempo real según el tránsito de la red, uno de estos sistemas es sobre el que trataré en este proyecto.

1.2. Objetivo

El objetivo de este proyecto es el desarrollo de un sistema de gestión de eventos e información de seguridad, abreviado con las siglas SIEM (Security Information and Event Management). Este sistema será desarrollado con ELK Stack.

El objetivo no es tan solo desarrollar este sistema, sino también aprender sobre conceptos de ciberseguridad y aplicar todos lo aprendido durante el grado de informática a un campo relativamente nuevo para mí.

Capítulo 2

Planificación

Este proyecto se desarrollará en cinco bloques principales, estos serán:

2.1 Establecer el entorno de trabajo

Este bloque comprende la preparación del entorno en Ubuntu 16.04 así como la instalación y configuración de las herramientas de ELK Stack. También aprender el funcionamiento de estas por tal de agilizar futuro trabajo que requiera del uso de algunas de sus funciones.

2.2 Planificación de una red ficticia

Dada la falta de recursos para utilizar una red real este bloque tiene como finalidad crear una pequeña red ficticia que recreará el funcionamiento de una real.

2.3 Desarrollo de scripts

Por tal de crear una red ficticia será necesario desarrollar scripts que hagan esta función, en este bloque además de desarrollar estos scripts se investigará qué formato tienen las líneas de log de los diferentes dispositivos de la red por tal de tratarlos en futuros bloques.

2.4 Tratamiento de los datos

El objetivo de este bloque será el de utilizar Logstash en profundidad para realizar un tratamiento inicial a todos los datos generados anteriormente a través de los distintos scripts.

2.5 Desarrollo de una aplicación adicional

Por tal de complementar todo este sistema y dar una visualización extra de lo que haría una empresa real, se desarrollará una aplicación a modo de demostración de las posibilidades que permite el desarrollo de un SIEM propio.

2.6 Diagrama de Gantt

A continuación, se muestra una aproximación de los tiempos que requerirá cada una de las tareas:

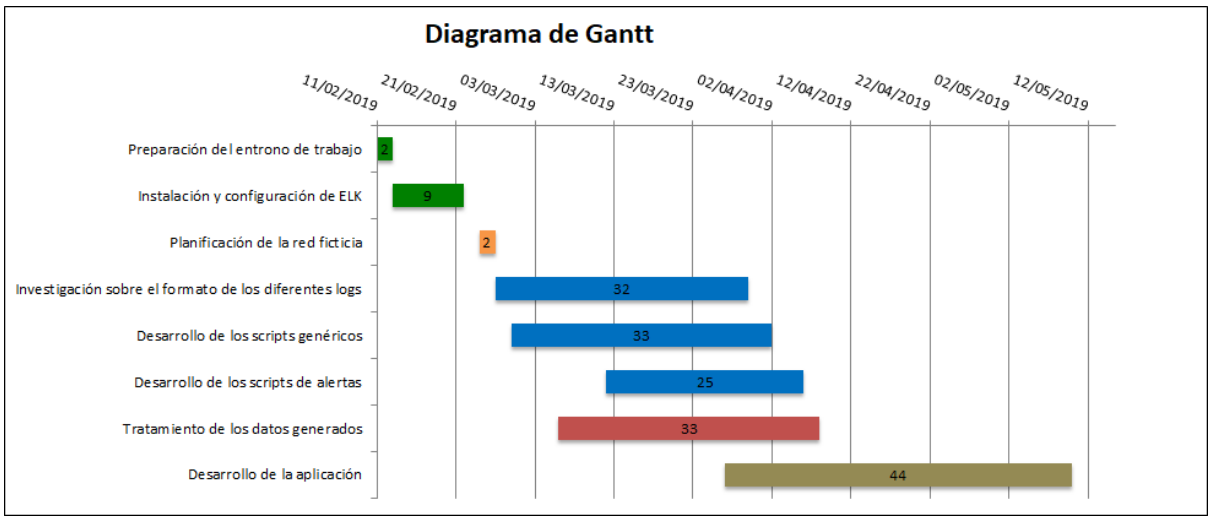

Figura 2.1: Diagrama de Gantt

Capítulo 3

Costes

En este apartado se analizarán los costes del proyecto.

3.1 Hardware

El único hardware necesario para este proyecto ha sido un ordenador portátil donde se desarrollará todo el proyecto mencionado anteriormente.

Ítem	**Precio**	**Cantidad**	**Total**
HP NoteBook 250 G5	438 €	1	438 €
Total			438 €

Tabla 3.1: Costes de hardware

3.2 Software

Dado que todo el software utilizado es gratuito, incluyendo ELK Stack que son herramientas Open Source, los costes de software serán nulos. Además de ELK también se incluye el sistema operativo Ubuntu 16.04 y el IDE IntelliJ de Idea.

Ítem	**Precio**	**Cantidad**	**Total**
ELK Stack	0 €	1	0 €
Ubuntu 16.04 LTS	0 €	1	0 €
Idea IntelliJ	0 €	1	0 €
Total			0 €

Tabla 3.2: Costes de software

Capítulo 4

Que es un SIEM

4.1. Contexto

Actualmente toda empresa necesita de redes para disponer de todos los recursos cuando sea necesario y, para ello, es necesaria una red conectada tanto interna como externamente. Esto conlleva abrir las puertas a nuevos riesgos y exponerse a ataques cibernéticos que podrían llegar a comprometer los datos de la empresa y de muchas personas. Para ello es necesario desarrollar sistemas que ayuden a la protección de estas redes, uno de estos sistemas, entre muchos otros, son los SIEMs.

4.2. ¿Qué es un SIEM?

Para responder esta pregunta lo primero que debemos saber es que un SIEM es la combinación de dos conceptos, estos son SIM (Security Information Management) y SEM (Security Event Management) de manera que un SIEM es un sistema de seguridad que se encarga de tratar eventos y notificaciones de una red, así como almacenar los datos de esta y analizarlos.

4.3. Usos en la actualidad

Actualmente los SIEM son una parte esencial de la seguridad en cualquier red de una empresa relativamente grande, permiten a los equipos de seguridad controlar todo lo que está pasando en la red a tiempo real y reaccionar rápidamente a posibles ataques y/o vulnerabilidades de la red. También, no solo se controla el tráfico externo de la red sino también interno, por ejemplo de los propios trabajadores, que webs se visitan, que inicios de sesión se producen y si han sido víctimas de algún ataque como por ejemplo el Phishing.

Las empresas tienen en el mercado varias alternativas a utilizar como SIEM, algunas de ellas son: QRadar de IBM, RSA enVision o Security MARS. Aun así, al ser sistemas muy complejos y un producto para grandes empresas, estos suponen un gran coste para la empresa, si vemos el ejemplo de QRadar, en su versión normal, empieza en unos 10.000 $ y en su versión Cloud empieza a 800$ mensuales, este precio varía según los EPS (Eventos Por Segundo) contratados.

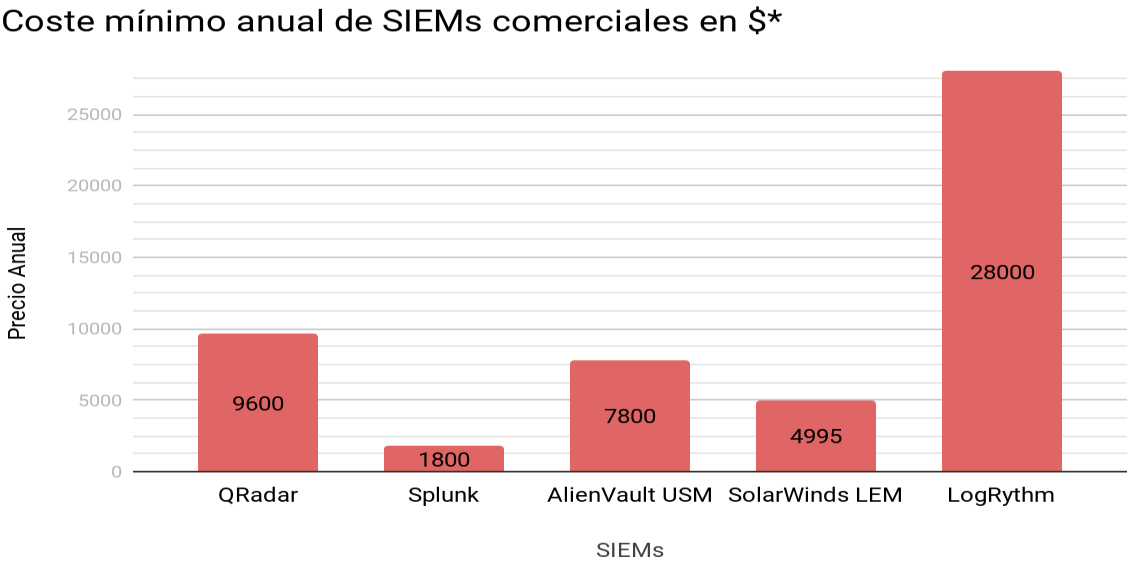

Figura 4.1: Coste mínimo anual de SIEMs comerciales[1]

Debido a esto, muchas veces se opta por utilizar ELK para desarrollar un sistema propio.

4.4 Casos de uso

Algunos de los casos de uso más comunes de los SIEMs son:

- **Gestionado de logs:** Un SIEM permite, como una de sus funcionalidades principales, gestionar todos los logs generados por una red de dispositivos y software de forma cómoda para un equipo de especialistas. Además de la capacidad de realizar investigaciones de, por ejemplo, la procedencia de logs concretos.

- **Detección y análisis de incidencias:** Junto con la gestión de logs, los SIEMs tienen la capacidad de detectar posibles incidencias mediante la correlación de eventos de la red realizando análisis de los logs recibidos. Además, permiten analizar estas incidencias por tal de aislar la procedencia del problema y poder actuar eficazmente.

Es importante remarcar que no debe tenerse a los SIEMs como una fiabilidad total de la seguridad de una red, es una gran herramienta que nos facilita mucho la seguridad de la empresa gracias a sus funciones y la capacidad de detectar problemas por sí solo, pero, es necesario tener un equipo preparado detrás.

1 Aproximación, cada compañía tiene sus propios planes en base a la cantidad de datos, eventos por segundo, etc. Estos valores pueden cambiar drásticamente en función de cada plan.

Pongamos un ejemplo simple:

Nuestro SIEM hace saltar una alerta si un usuario intenta iniciar sesión más de diez veces en cinco minutos, pero, resulta que hay un dispositivo que realiza nueve intentos cada diez minutos, en este caso, el SIEM no haría saltar ninguna alerta, pero un equipo preparado debería ver algo extraño en el conjunto de logs que aparecen y detectarlo ellos mismos.

Este es un ejemplo en el que podemos ver que no es bueno depender completamente de un SIEM, sino que también puede ser simplemente una herramienta complementaria para el equipo de técnicos encargados de la seguridad.

4.5 Comerciales o de desarrollo propio

Como he mencionado anteriormente, una de las razones por las que a menudo se opta por el desarrollo de un SIEM propio es debido a los altos costes que tienen los SIEMs comerciales, aun así, esta no es la única razón, entran varios factores. Se trata, sobre todo, de equilibrar la balanza de nuestras necesidades, ¿Tenemos una gran infraestructura y necesitamos todas las funcionalidades posibles y la mayor fiabilidad? Dado que tenemos una gran infraestructura posiblemente seamos una gran empresa y por lo tanto nuestra mejor opción sea optar por un SIEM comercial.

Pero, ¿Y si tan solo necesitamos pequeñas funcionalidades como tener un pequeño control de lo que pasa en la red? Una solución sería buscar un SIEM que cumpla con esta necesidad y a un coste razonable. Otra opción sería desarrollar nuestro propio sistema, si bien no será de una gran complejidad comparándolo a grandes como QRadar, satisfará nuestras necesidades y a un coste mucho menor.

Capítulo 5

ELK

5.1. Introducción a ELK Stack

ELK será la base de este proyecto de desarrollo de un SIEM, corresponde a las siglas de Elasticsearch-Logstash-Kibana, lo cual son tres herramientas con diferentes funcionalidades que se complementan unas con otras.

5.1.1. Elasticsearch

Elasticsearch es un motor de búsqueda basado en Lucene[1] que permite búsqueda de texto completo, dispone de una interfaz HTTP y está basado en documentos JSON para el almacenado de los datos. Está desarrollado en JAVA lo cual lo hace compatible con gran cantidad de dispositivos. Dadas estas características, además de un motor de búsqueda *full-text*, lo podemos considerar un sistema de almacenamiento NoSQL, concretamente un sistema de documentos.

5.1.2. Logstash

Logstash es un procesador de textos, su función consiste en, a partir del texto de entrada que recibe, procesarlo mediante plugins, y enviar el resultado a la salida que especifiquemos. Un procesado del texto puede ser por ejemplo dividirlo en diferentes campos y conformar un JSON que enviaríamos a Elasticsearch.

5.1.3. Kibana

Se trata de una herramienta web que permite visualizar todos los datos de Elasticsearch mediante filtros y diferentes tipos de representaciones de datos que el propio usuario puede elegir. También tiene la posibilidad de administrar Elasticsearch modificando los índices y realizando *queries*.

[1] API de código abierto para recuperación de información

5.2. Instalación y configuración

Lo primero corresponde a instalar estas herramientas, para ello se hará uso de las guías proporcionadas por los propios desarrolladores de Elastic siguiendo también el orden recomendado [1]. Aun tratándose de un conjunto de herramientas que se tratan como una sola, deben instalarse por separado. Se ha decidido realizar esta instalación en Linux, concretamente Ubuntu 16.04 LTS, pero las herramientas de ELK están disponibles también para Windows y MAC OS.

5.2.1. Elasticsearch

Se instalará Elasticsearch mediante la función apt de Ubuntu de la manera que se explica en la web de los desarrolladores [2].

Una vez instalado se configurará, por comodidad y siguiendo también las recomendaciones de los desarrolladores, que Elasticsearch se ejecute al iniciar el equipo mediante los comandos:

```
sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable elasticsearch.service
```

Hay otras alternativas a este comando en la página mencionada por si este método no funcionara.

Para comprobar que todo funciona correctamente se hace una solicitud HTTP al puerto 9200 que es el predeterminado en la configuración de Elasticsearch.

```
curl -X GET "localhost:9200/"
```

Esto retorna como respuesta una salida en formato JSON similar a esta:

```
{
    "name": "BEtqUIs",
    çluster_name": .elasticsearch",
    çluster_uuid": "dH9_j2FOQM2vF_P5UyKX4g",
    "version": {
        "number": "6.4.0",
        "build_flavor": "default",
        "build_type": "deb",
        "build_hash": "595516e",
        "build_date": "2018-08-17T23:18:47.308994Z",
```

```
    "build_snapshot": false,
    "lucene_version": "7.4.0",
    "minimum_wire_compatibility_version": "5.6.0",
    "minimum_index_compatibility_version": "5.0.0"
  } ,
  "tagline": "You Know, for Search"
}
```


Aunque con esta configuración básica se dice que todo funciona correctamente, se nos da una lista de las opciones más importantes a considerar una vez empecemos a utilizarlo [3], estas opciones serán valoradas en adelante una vez vistas las necesidades del proyecto.

5.2.2. Kibana

Se instalará Kibana de la misma forma que Elasticsearch siguiendo también la explicación de la web de los desarrolladores [4].

Se utilizará Kibana con la configuración mínima necesaria, de igual manera que con Elasticsearch se iniciará Kibana al iniciar el equipo con los siguientes comandos:

sudo /bin/systemctl daemon-reload
sudo /bin/systemctl enable kibana.service

Una vez hecho esto e iniciado Kibana, se pasa a testear su correcto funcionamiento, para ello accedemos a *localhost* en el puerto 5601 ya que es el predeterminado en la configuración de Kibana.

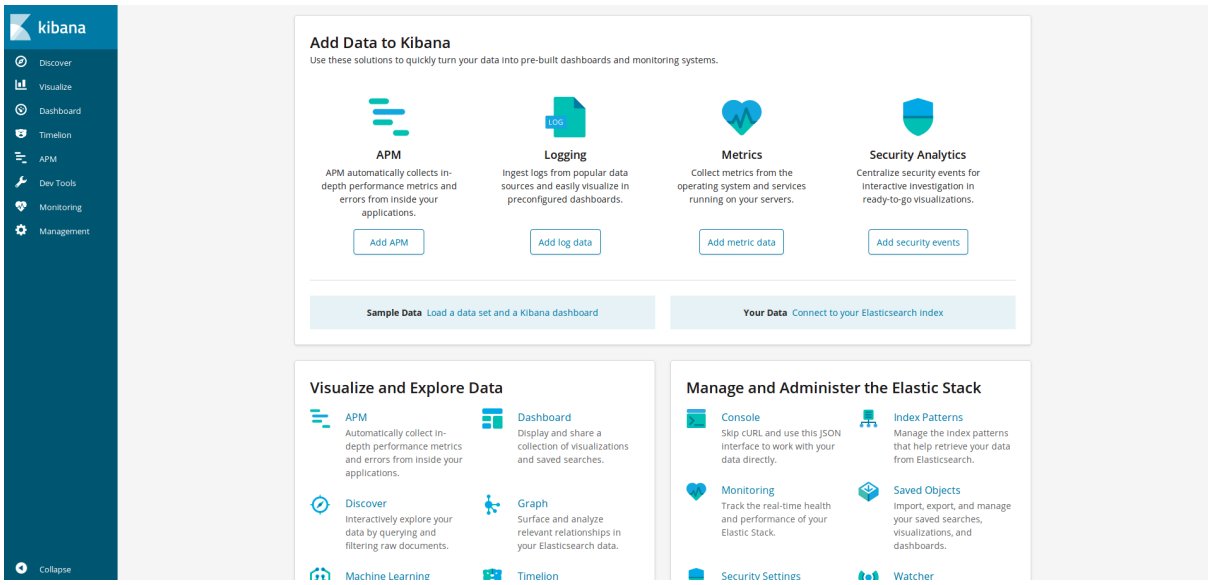

Figura 5.1: Página principal de Kibana

Más adelante se consultará, si es necesario, como modificar la configuración de Kibana siguiendo la documentación de la web [5].

5.2.3. Logstash

Una vez más se instalará Logstash utilizando la función apt [6]. Logstash es la última de las herramientas correspondiente a la pila principal de ELK. Como anteriormente, se iniciará Logstash al iniciar el equipo mediante los comandos mostrados en la web [7].

Logstash se configurará de tal manera que el puerto por el que reciba los datos de Beats, herramienta explicada a continuación, sea el 5044, para hacer esto se debe crear un archivo .conf que se utilizara para crear un *pipeline* entre los diferentes Beats instalados y Logstash. Este archivo tiene la siguiente forma final:

```
input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => "localhost:9200"
    manage_template => false
    index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
  }
}
```

Figura 5.2: Archivo .conf de Logstash

Esta es la configuración inicial, a este archivo se puede añadir un apartado filter para tratar los diferentes datos, el cual se utilizará en un futuro.

Lo que estamos diciendo a Logstash en esta configuración es que debe esperar los datos de Beats por el puerto 5044 y que la salida de estos una vez tratados será a Elasticsearch, poniendo como índice el nombre del beat y la fecha del momento.

5.2.4. Beats

Beats son pequeños programas que envían logs a Logstash, estos son instalados en cada uno de los servidores de los que queremos recibir datos y se dispone de diferentes productos que se pueden instalar por separado.

- *Auditbeat*: Se encarga de recopilar datos de las actividades de los usuarios y los procesos del sistema.
- *Filebeat*: Recolecta logs de los eventos de un sistema y los envía a Logstash.
- *Heartbeat*: Nos da información de la latencia entre dos puntos.

- *Metricbeat*: Recopila datos del sistema operativo, como la memoria utilizada.
- *Packetbeat*: Guarda los paquetes que transitan por la red.
- *Winlogbeat*: Recoge información de los eventos de Windows.

En este caso se utilizará Filebeat ya que es el que mejor se ajusta a las necesidades del proyecto.

Aun así, se empezará con Metricbeat ya que es un Beat más simple y me permitirá ver el funcionamiento a través de la guía de los desarrolladores [8].

5.2.4.1. Metricbeat

Metricbeat permite visualizar datos básicos del equipo que lo esté ejecutando como pueden ser procesos o memoria utilizada.

Se instalará Metricbeat siguiendo la guía, y la configuración de este es realmente simple.

Por defecto los Beats envían los datos directamente a Elasticsearch pero es interesante que estos pasen por Logstash, para ello hay que modificar la configuración en el archivo metricbeat.yml, en el apartado Outputs se eliminará todo lo relativo a Elasticsearch y añadiremos las líneas de Logstash:

```
#================================ Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

#-------------------------- Elasticsearch output ------------------------------
#output.elasticsearch:
  # Array of hosts to connect to.
  #hosts: ["localhost:9200"]

  # Optional protocol and basic auth credentials.
  #protocol: "https"
  #username: "elastic"
  #password: "changeme"

#----------------------------- Logstash output --------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

Figura 5.3: Archivo de configuración de los Beats

Hecho esto Metricbeat ya está mínimamente configurado.

5.2.4.1.1. Prueba

Para comprobar que este funciona correctamente ejecutaremos Logstash para que escuche los datos entrantes y accederemos a Kibana. Seleccionando *metricbeat* veremos los datos recibidos.

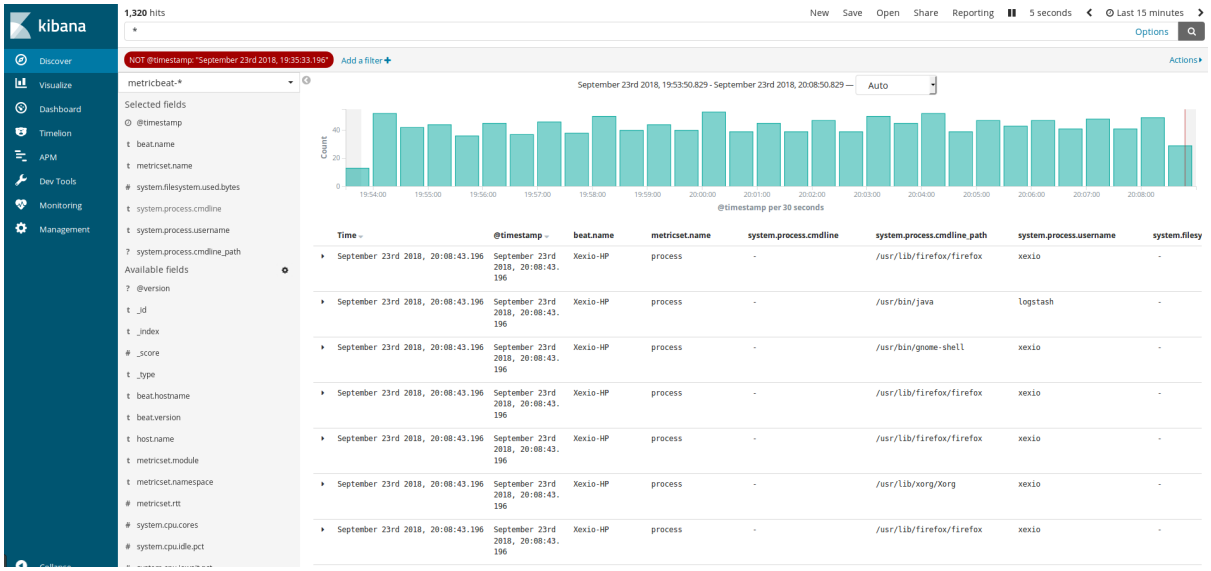

Figura 5.4: Kibana en funcionamiento

5.2.4.2. Filebeat

En la página de los desarrolladores tenemos instrucciones para la instalación de cada uno de los Beats, en el caso de Filebeat el proceso es el siguiente [9].

Primero de todo se instalará Filebeat mediante los comandos nombrados en la página y se configurará.

El archivo de configuración se encuentra en el directorio /etc/filebeat/filebeat.yml, este será modificado para configurar Filebeat.

El primer paso es decidir en qué directorio leerá Filebeat en busca de logs, como directorio predeterminado tenemos /var/log/, se mantendrá así ya que es la localización donde UNIX guarda los logs de forma predeterminada.

De la misma forma que con Metricbeat se modificara el archivo de configuración para enviar los datos a Logstash y no a Elasticsearch.

Una vez hecho esto iniciamos Filebeat.

Capítulo 6

Simulación, recolección y filtrado de datos

6.1. Datos utilizados

Como datos se utilizarán logs simulados mediante scripts, estos scripts generan logs de firewall, SSH, router y similares que tendrán el mismo formato que logs reales. Los scripts están programados mediante Python 3.6 por la rapidez y facilidad con la que permite programar este lenguaje.

6.2 Simulación de los datos

Para simular los logs de la red será necesario investigar cómo están formados los logs que producen todos los elementos de esta, en concreto nuestra red está formada por:

- Nueve equipos Linux.
- Tres equipos Windows.
- Un servidor web Nginx.
- Un router Cisco.
- Un firewall Cisco ASA.
- Una base de datos PostgreSQL.
- Dos switchs.
- Un punto de acceso inalámbrico.

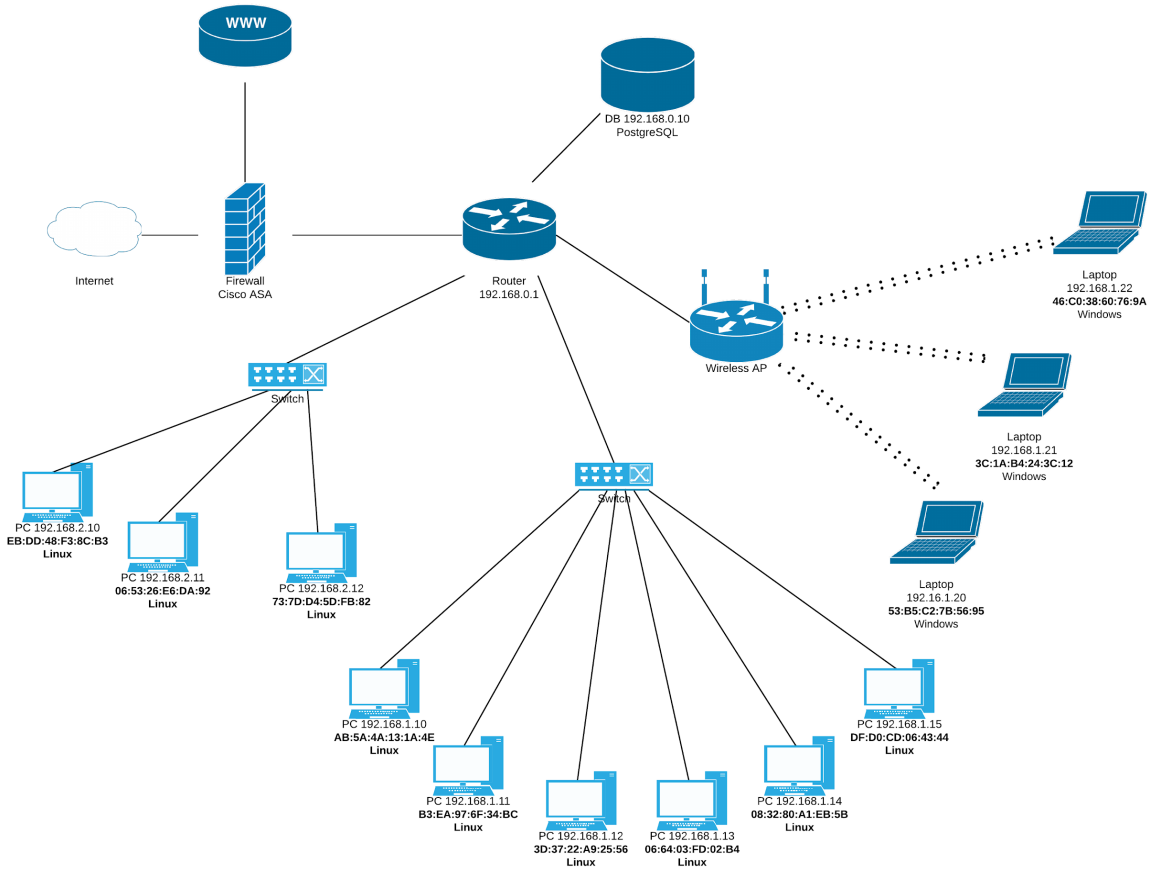

Figura 6.1: Diagrama de la red ficticia

Concretamente se simularán logs de los siguientes elementos:

- SSH con doce usuarios distintos pertenecientes a la red.
- Firewalls de Ubuntu y Windows, uno para cada equipo.
- Eventos del servidor web Nginx.
- Logs de ambos dispositivos Cisco.
- Logs de la base de datos SQL

La mayor dificultad en la simulación de logs es para los elementos físicos de la red y no software, ya que resulta más complicado obtener ejemplos reales y suelen ser más complicados de simular debido a la gran cantidad de variantes. Se intentará que sean lo más realistas posibles pero sin dejar que nos consuma un tiempo excesivo, ya que no es el objetivo del proyecto.

Como ya hemos dicho, serán generados mediante scripts, de manera que se estarán ejecutando continuamente y generando datos y "ruido" al sistema. Estos serán los logs genéricos, que aportan información del estado de la red, pero nada realmente importante, simplemente una red funcionando con normalidad.

Se crearán otros scripts que generen líneas de logs más específicas e interesantes, como puede ser inicios de sesión extraños dentro de la misma red, ataques externos o accesos a páginas web potencialmente peligrosas, que ejecutaremos nosotros mismos por tal de ver la reacción de nuestro sistema.

6.2. Recolección de los datos

Los datos serán recolectados por Filebeat, Beat explicado en el capítulo anterior. Los datos obtenidos por este están sin procesar, de manera que pasarán por Logstash para su tratamiento y de esta manera disponer de una mejor organización en los datos de Elasticsearch.

6.3. Filtrado de los datos

Para el filtrado de los datos utilizaremos la configuración de la pipeline entre los Beats y Logstash, mencionada anteriormente. Dentro de esta configuración se dispone de un campo *filter* que permite utilizar plugins que reconocen patrones y crean nuevos campos en los datos de Elasticsearch, en este proyecto se utilizará el plugin Grok ya que es muy versátil y se adapta perfectamente a las necesidades con gran cantidad de *Regular Expressions*. En la siguiente imagen se puede observar el filtrado de un log del firewall de Ubuntu.

```
filter {
  if [source] =~ "firewall.logtest" {
    grok{
      match => { "message" => "\[UFW%{SPACE}%{GREEDYDATA:flag}\] IN=%{GREEDYDATA:IN} OUT=%{GREEDYDATA:OUT}
      SRC=%{IP:origen} DST=%{IP:destino}" }
    }
  }
```

Figura 6.2: Filtro de Logstash con Grok

El formato de los patrones de Grok viene dado por la forma % { *SYNTAX:SEMANTIC*} donde *SYNTAX* será el patrón del *String* deseado y *SEMANTIC* el nombre del campo en Elasticsearch. Se dispone de un amplio abanico de patrones ya definidos por los desarrolladores [12] y también se aceptan expresiones con el formato Oniguruma [13] para patrones personalizados.

Es importante resaltar que debe realizarse este proceso para cada una de las distintas líneas de log que nos interese de nuestra red, de manera que esta es una parte de vital importancia y que requiere de un trabajo humano ya que cada línea de log puede ser muy distinta dependiendo del dispositivo que la genere.

6.4 Organización de los datos

Por tal de mantener un buen orden de los logs y facilitar la consulta de estos, se almacenarán en distintos índices de Elasticsearch, estos tendrán un nombre de la forma *<Fuente-del-log>-<Año.Mes.Dia>*. Para mantener esta organización simplemente añadiremos unas líneas a la configuración de Logstash de manera que, dependiendo de donde provenga el log, este se insertará, una vez tratado, a su correspondiente índice.

La línea se añadirá en el apartado *output* del archivo de configuración de la *pipeline* de la siguiente forma:

index =>" % { [@metadata][fuente]} - % { +YYYY.MM.dd} "

Capítulo 7

Ataques

7.1 ¿Qué se considera un ataque cibernético?

Se considera un ciberataque cualquier tipo de ofensiva que tiene como objetivo información de sistemas, infraestructuras, redes o dispositivos personales. De manera que cualquier acceso a una red de forma no autorizada es considerado un ataque cibernético.

Por lo tanto un ciberataque comprende, entre otras, cualquiera de las siguientes acciones:

- Robo de identidad
- Uso de cualquier tipo de malware
- Robo de hardware
- Ataques de denegación de servicio
- Robo de credenciales
- Infiltración en sistemas

7.1 ¿Por qué simular ataques?

El hecho de simular ataques en nuestra propia red puede parecer algo innecesario en un primer momento, pero es muy importante saber cómo reaccionará nuestro sistema ante cierto tipo de ataques.

El conocimiento de esta respuesta permite, por una parte, detectar mucho más rápido la causa del problema, y por otra, permite al equipo encargado de tratar con los incidentes en la empresa, reaccionar rápidamente y gracias a ello, que el daño causado a la red comprometida sea mínimo o incluso nulo.

Gracias a realizar estas pruebas se pueden acordar protocolos internos de la empresa para tratar con diferentes problemas, además permite entrenar a los equipos en nuevos tipos de ataques que hayan surgido. Todo este proceso puede compararse a un simulacro de incendios.

7.2 Ataques simulados

Para este proyecto los ataques que simularé no serán realmente complejos pero sí muy comunes, serán, de la igual manera que los logs genéricos, generados mediante scripts en Python que escribirán líneas concretas en los mismos archivos de logs. Los scripts desarrollados generarán las siguientes alertas:

- **Múltiples *drops* de los firewalls**, tanto de Ubuntu como de Windows, un caso como este puede indicar que se está intentado realizar un acceso a cierto equipo de forma no autorizada y por lo tanto los firewalls están bloqueando la conexión entrante.
- **Inicios de sesión de Secure SHell (SSH) en distintos equipos en un corto plazo de tiempo**, esto resulta sospechoso sobre todo en redes empresariales donde se trabaja a distancia desde distintas ciudades, de manera que si un mismo usuario que está trabajando en Madrid inicia sesión, y, pasados unos minutos, se produce un login del mismo usuario en Barcelona, es muy probable que se haya vulnerado la seguridad de alguna manera. Ya sea por un intruso externo o por un usuario que ha vulnerado la seguridad de la empresa.
- **Múltiples inicios de sesión de SSH en un corto plazo de tiempo**, si la cantidad de intentos fallidos es considerable muy probablemente nos encontremos ante un ataque de fuerza bruta.
- **Escaneo de puertos**, si una dirección concreta está enviando conexiones a varios puertos de nuestra red puede ser señal de que se está realizando un escaneo en busca de puertos vulnerables que puedan ser objetivo de otros ataques, si bien como tal no supone un peligro el escaneo por sí solo, puede ser precedente de un ataque mayor.
- **Acceso a URLs críticas**, dado que en una empresa no es interesante que se acceda a ciertos tipos de páginas como podrían ser páginas de descargas u otras similares potenciales de contener algún tipo de malware, se necesita controlar si en algún momento alguno de los usuarios de la red intenta acceder o acceder a alguna de estas, para así notificarlo y/o añadir dicha URL a una lista negra.
- **ARP Poisoning o ARP Spoofing**, es un ataque muy peligroso ya que su objetivo es el de enviar mensajes ARP[1] falsos con el fin de asociar la dirección MAC del atacante con la IP de un nodo de la red y de esta manera "formar parte de ella" .

7.3 El papel de ELK

El papel más importante de ELK en esta parte, obviando por supuesto que es el encargado de la recolección de los datos, es la posibilidad que nos da de realizar *queries* realmente detalladas gracias a la búsqueda full-text de Elasticsearch. Con este sistema podremos realizar queries, que, dependiendo de su resultado, pueden provocar una alerta bien si

[1] Address Resolution Protocol o protocolo de resolución de direcciones, se trata de un protocolo de comunicación de la capa de red que se encarga de encontrar la dirección MAC asociada a una IP concreta.

la realizamos desde Kibana y vemos un resultado preocupante o, si nuestra aplicación desarrollada detecta que hay algún problema con el resultado obtenido de esta.

Capítulo 8

Cliente

Dado que el objetivo es desarrollar un SIEM, es interesante también el desarrollar una aplicación con una interfaz gráfica que facilite el uso de este. Esta aplicación se ha decidido programarse en JAVA, por varios factores. Uno es que es el lenguaje al que más adaptado estoy y del que más conocimientos tengo en el campo de lenguajes orientados a objetos, y otro, es la gran facilidad con la que se pueden desarrollar interfaces gráficas en JAVA gracias a *Swing*[1].

Si bien esta aplicación se desarrollará completamente una vez establecida la base de los datos, se desarrollará una primera parte que me facilite hacer pruebas y que posteriormente se incluirá en la aplicación final.

8.1.1 Primeros pasos

La primera base de la aplicación será, como ya he dicho, una utilidad personal que ayudará a un desarrollo más eficiente, esta constará de una pequeña interfaz en la que se pueden ejecutar los scripts que generan los logs así como también logs especiales que simularán situaciones de alerta.

Está desarrollada de manera que se dispone de dos menús desplegables, uno para logs y otro para alertas. La ventaja de estos menús es que leen los scripts almacenados en carpetas concretas y por lo tanto permite ir añadiendo más scripts sin necesidad de desarrollar nuevo código, esto consta de una mejora de una idea anterior que consistía en tener un conjunto de *checkboxes*, pero eso provocaba que hubiese que reprogramar la aplicación a cada nuevo script y por lo tanto no resultaba útil. Además, también consta de un campo de texto donde incluir parámetros para estos scripts.

La idea de esta utilidad es que en un futuro se incluya en el SIEM como herramienta de *testing*, de manera que un usuario del SIEM, pueda desarrollar un script que simula, por ejemplo, un nuevo tipo de vulnerabilidad, añadirlo, y ver como el SIEM reacciona a esta.

[1] Swing es una biblioteca gráfica para Java. Incluye widgets para interfaz gráfica de usuario tales como cajas de texto, botones, listas desplegables y tablas.

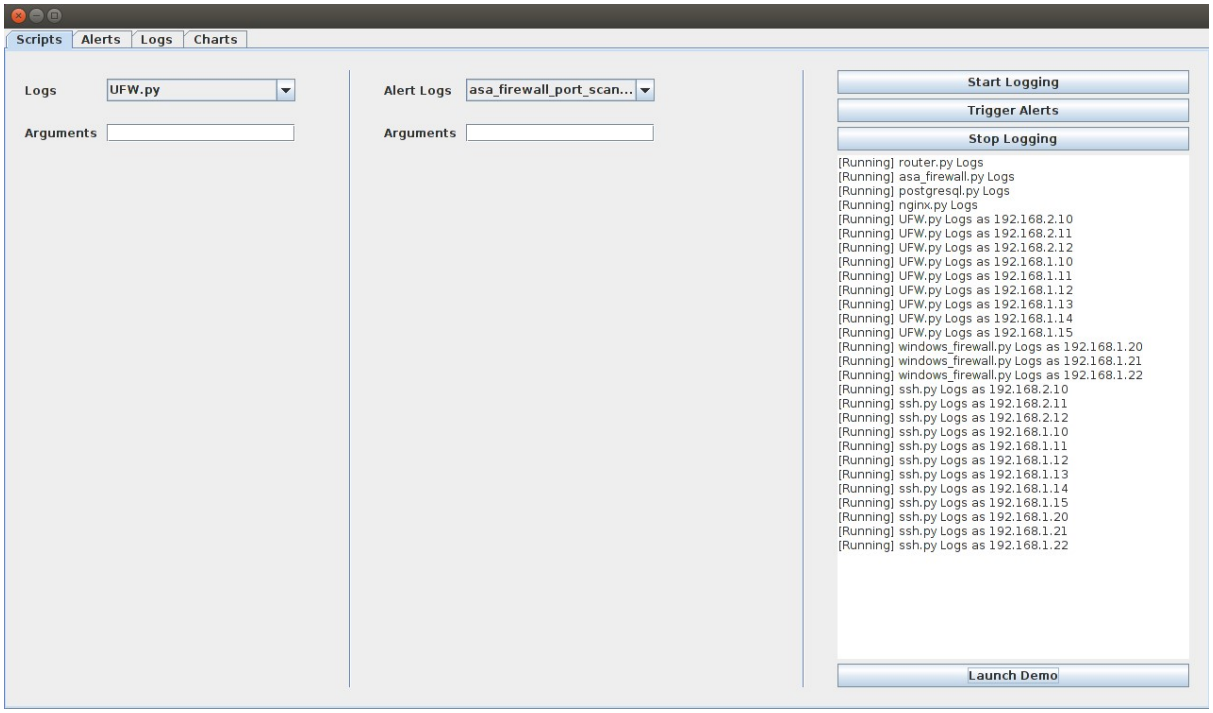

Figura 8.1: Ventana de ejecución de scripts de la aplicación

8.1.2 Configuración

Por tal de dar más libertad a la aplicación se ha añadido un archivo de configuración con un formato propio, de manera que se podrán modificar aspectos de la aplicación de manera mucho más sencilla, como por ejemplo, escoger donde se escribirán los logs simulados.

Este archivo tiene el nombre "SIEM.conf" el formato para la configuración tiene la forma *<atributo> : <valor>*. Además, acepta comentarios mediante el uso de almohadillas ("# ").

Sus principales atributos son:

- *logsPath*: Corresponde al URI absoluto donde serán escritos los logs
- *logScriptsPath*: Corresponde al URI absoluto donde se buscarán los scripts de logs, en el caso de que se introduzca un punto (" . ") se utilizará la carpeta predeterminada "logs" , situada en el directorio del ejecutable
- *alertScriptsPath*: Corresponde al URI absoluto donde se buscarán los scripts de alertas, en el caso de que se introduzca un punto (" . ") se utilizará la carpeta predeterminada "alerts" , situada en el directorio del ejecutable
- *python-exec*: Corresponde al comando de Python utilizado para ejecutar los scripts, ya que cada máquina puede tener su propia variante, por ejemplo, "python3" o "python2" , como valor predeterminado se utiliza "python"

8.1.3 Detección de alertas

El siguiente paso consiste en elaborar ya una función que haría un SIEM real, y esta será detectar alertas a partir de los logs que se vayan almacenando en Elasticsearch después de ser filtrados en Logstash. Para ello será necesario elaborar *queries* que permitan detectar estos problemas de la red.

Se utilizará una API de Java para realizar las queries a Elasticsearch desde la propia aplicación, de manera que resulte más cómodo tratar con ellas, esta API [14] está desarrollada también por Elastic Co, tan solo será necesario añadir las dependencias correspondientes a nuestra aplicación.

Añadiremos también un pequeño código de color que permita distinguir niveles de alerta, así como algunas funcionalidades que permitan investigar mínimamente la alerta y exportarla.

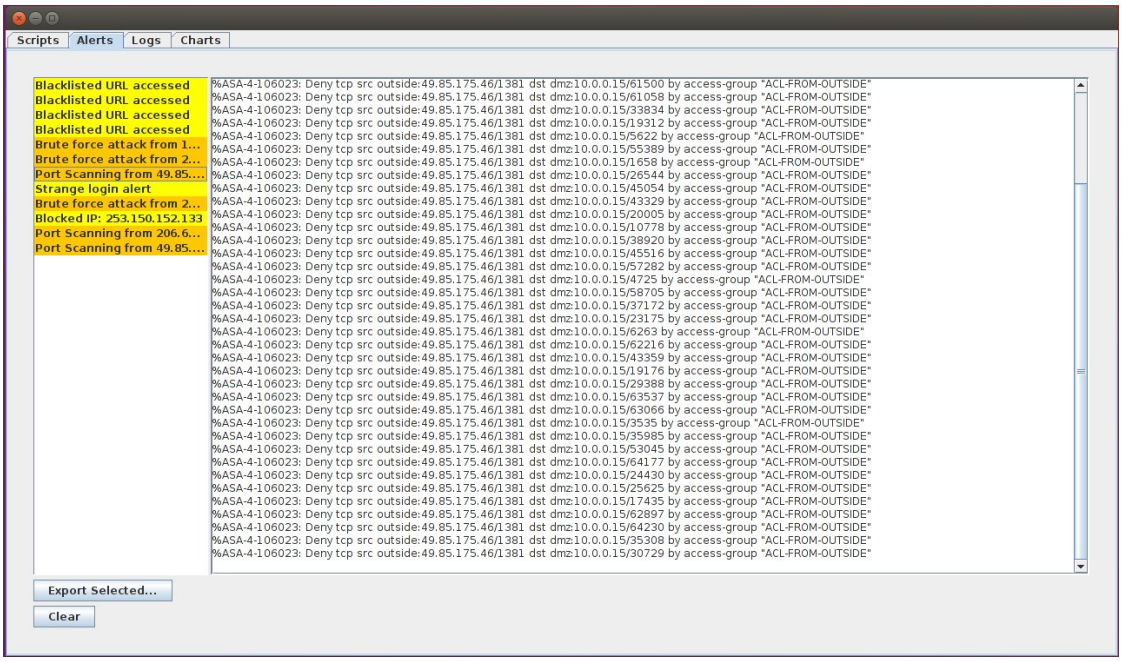

Figura 8.2: Ventana de muestra de alertas

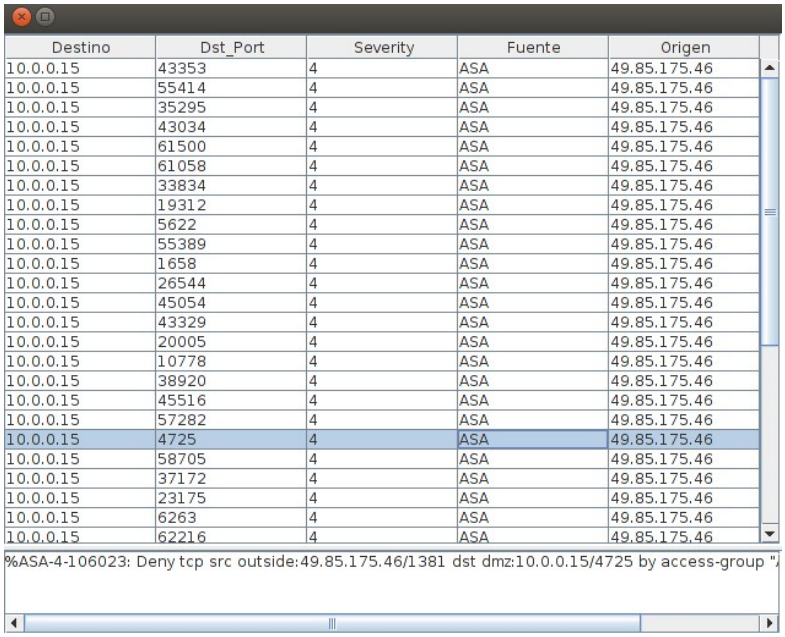

Figura 8.3: Ventana de información de alerta

8.1.3.1 Ejemplo de query

La primera query que haremos servirá para la detección de un inicio de sesión de un mismo usuario en diferentes equipos en un plazo de tiempo muy corto, lo cual resulta extraño en una red empresarial. Esta *query* se escribirá primeramente como una *query* de búsqueda normal, es decir en formato JSON que es el estándar de Elasticsearch, esta es la mejor manera de hacer pruebas y asegurarse de que funciona correctamente ya que resulta muy cómodo, el resultado de la *query* es el siguiente, y consta de dos partes:

La primera parte es la query en sí, es decir, qué condiciones tendrá en cuenta para escoger los *records*. Para este caso, se escogen aquellos *records* que contengan "sshd" en el campo "Fuente" y todos aquellos que estén en un rango de tiempo de quince segundos anteriores a la ejecución de la *query*, ya que estas *queries* se ejecutarán cada cierto tiempo:

```
{
        "size": 0,
        "query": {
                "bool": {
                        "must": [{
                                "match": {
                                        "Fuente": ”sshd”
                                }
                        } ,
```

```
                {
                    ”range”: {
                        "@timestamp": {
                            "gte": ”now-15s”
                        }
                    }
                }
            ]
        }
    } ,
```

La segunda parte consta de una agregación, donde se agrupan los *records* por nombre de usuario, y dentro de cada nombre de usuario, cada uno de los orígenes, de manera que si aparece un *record* con más de un origen en su usuario, detectaremos una alerta:

```
”aggs” : {
        "by_username": {
            "terms": {
                "field": ”username”
            } ,
        ”aggs” : {
            "by_ip": {
                "terms": {
                    "field": ”Origen”
                }
            }
        }
    }
```

Una vez sabemos que esta *query* funciona correctamente es momento de pasarla a la API de Java.

8.1.4 Mostrar las alertas

Para mostrar las alertas haremos uso del patrón de programación *Observer* que consiste en tener un objeto observable al que se la adjunta un observador, de manera que cuando se produce un cambio que queremos que el observador sepa, lo notificamos desde el objeto observable, resulta muy útil por tal de no romper la distinción entre la vista (GUI) y el resto de la lógica del proyecto.

Se ha optado por mostrar las alertas en una lista, de manera que cada alerta va acompañada de un mensaje y un grado de severidad que será mostrado según el color del elemento de la lista, de esta manera se puede detectar rápidamente qué alertas son prioritarias.

Si se hace un doble clic en alguno de los elementos se da información extendida sobre la alerta producida.

8.1.5 Logs en tiempo real

Si bien esto ya es una funcionalidad que nos ofrece Kibana de base, puede ser interesante desarrollarla para nuestra aplicación y de esta manera depender menos de Kibana y centrar toda la atención a la hora de utilizar el SIEM en la aplicación.

Para ello se ha desarrollado también una ventana donde se muestran los logs que se van generando en tiempo real, separados por campos y con posibilidad de cambiar el tiempo de actualización de la tabla.

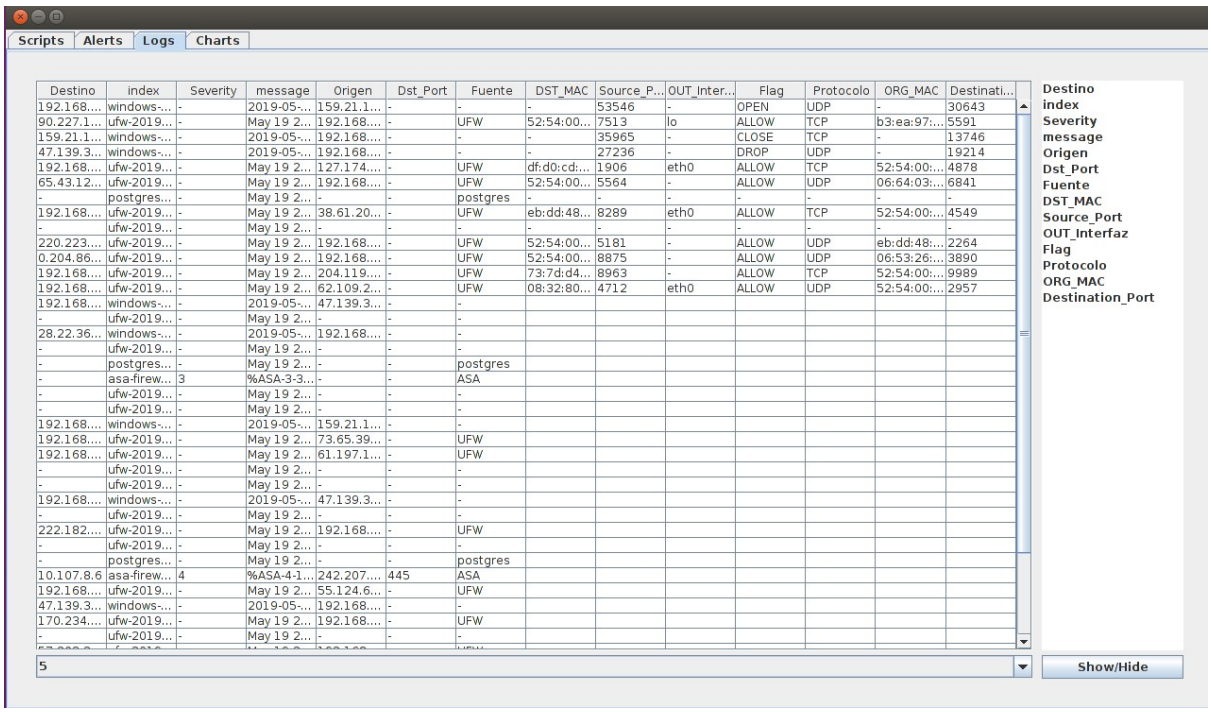

Figura 8.4: Ventana de visualización de logs entrantes

8.1.6 Gráficos

Como última funcionalidad añadiremos una pestaña con la capacidad de mostrar gráficos sobre ciertos datos, se ha desarrollado de manera muy simple ya que es algo que conllevaría mucho tiempo de desarrollar correctamente, pero su objetivo es el mostrar otra más de las muchas funcionalidades que se podrían añadir, en este caso se generan tres gráficos distintos que muestran, por ejemplo que URLs críticas han hecho saltar más alertas.

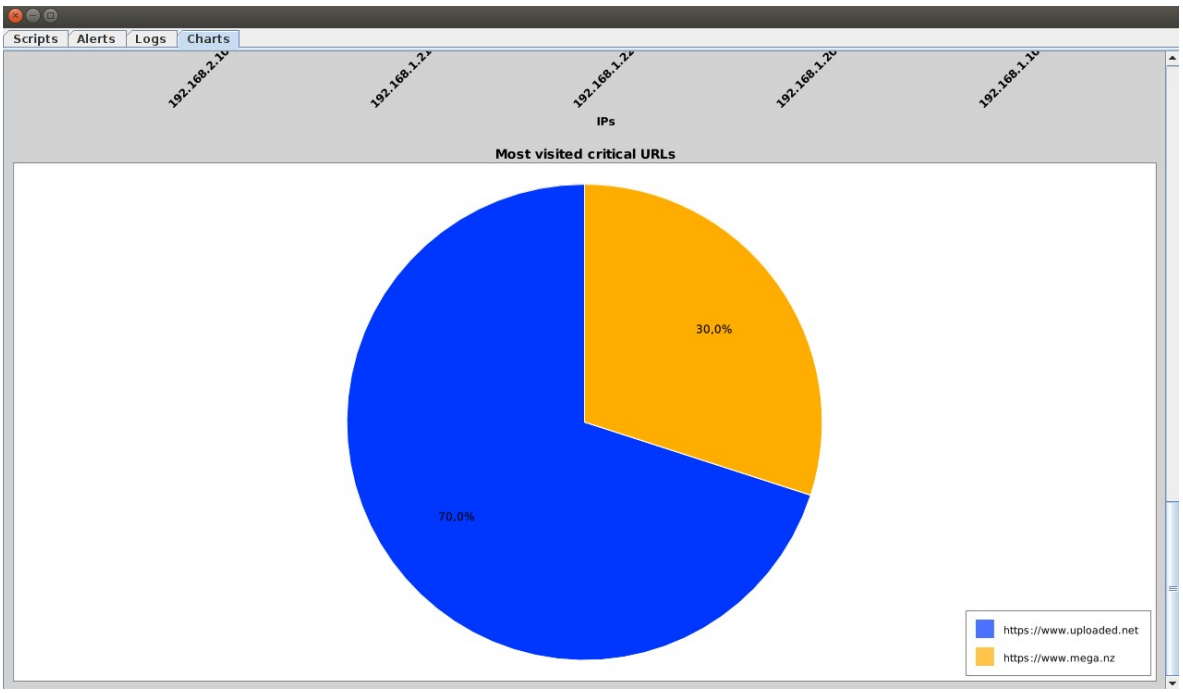

Figura 8.5: Ventana de visualización de gráficos

8.2 Mejoras

Como ya he mencionado, se trata de una aplicación que muestra las diferentes posibilidades que se podrían añadir, pero no se encuentra ni mucho menos al nivel que requeriría una empresa. Se deberían introducir muchas mejoras, algunas de ellas podrían ser:

- **Mejoras de UX**, de manera que resulte mucho más cómodo de utilizar y visualmente sea más intuitivo.
- **Control de errores mucho más alto**, por tal de evitar que la aplicación deje de funcionar correctamente.
- Capacidad de **generar informes mucho más detallados** sobre cada una de las alertas, añadiendo por ejemplo gráficos que acompañen el archivo PDF sobre las IPs relacionadas con la alerta.
- Dar la posibilidad de **añadir nuevas alertas por parte del usuario**, en este caso el equipo de seguridad, de manera que estas no se encontrasen *hardcoded* sino que

pudieran ser añadidas de una forma similar a los scripts. Esto permitiría una mejor escalabilidad de las alertas que el SIEM puede llegar a detectar.

- **Introducción de diferentes filtros y/o agregaciones** a la ventana de logs, de forma que sea mucho más cómodo leer lo que está pasando a tiempo real.
- **Permitir crear gráficos personalizados** y no fijos como actualmente. De esta manera se podría analizar el estado de la red de una forma más cómoda.

Capítulo 9

El conjunto

9.1 Sistema completo

Ahora que ya hemos visto los distintos elementos del proyecto, es decir de nuestro SIEM, hagamos un pequeño resumen del funcionamiento general.

Tenemos los siguientes elementos en nuestro sistema:

- Filebeat
- Elasticsearch
- Logstash
- Kibana
- Scripts
- Aplicación

Como he ido explicando a lo largo del proyecto, el flujo de datos resultante de este sistema empieza en los scripts, estos generarán unos logs (un archivo para cada uno de los elementos de la red) que serán recolectados por Filebeat, la única función de Filebeat será recogerlos y a continuación enviarlos a Logstash, el cual, se encargará de tratar estos logs utilizando el plugin Grok y dividiéndolos en varios campos con el formato de un JSON que será enviado a Elasticsearch. Una vez en Elasticsearch, estos datos pueden ser consultados o bien por Kibana o bien por nuestra propia aplicación. A continuación un diagrama simple que muestra este flujo de datos:

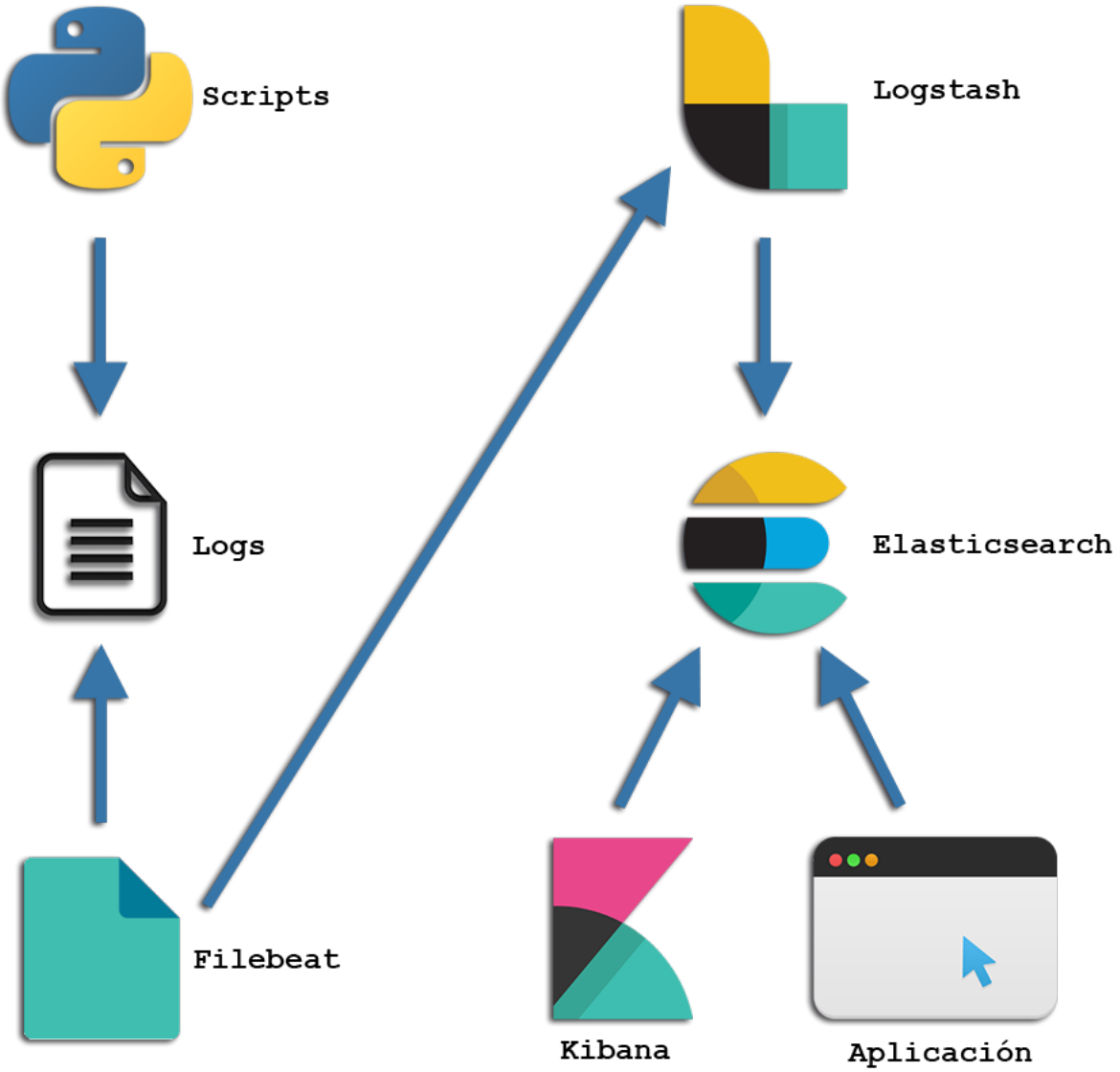

Figura 9.1: Diagrama del sistema

Todo este conjunto formaría nuestro SIEM, que como podemos ver, si bien se trata como una sola unidad, el SIEM es un sistema formado por varias herramientas que se complementan entre sí.

Capítulo 10

Conclusiones

Como vemos, disponer de un SIEM en cualquier empresa medianamente grande es algo necesario, ya que es una gran contribución a la seguridad de esta en un mundo en el que todo tiene la necesidad de estar conectado a la gran red que forma Internet.

Por esta misma razón, al ser algo realmente necesario, disponemos de gran cantidad de alternativas, como se ha explicado, tanto de pago como de desarrollo propio. En este proyecto hemos visto que el desarrollo de un SIEM propio, en gran parte gracias a ELK, es una opción realmente viable para conseguir unas funcionalidades mínimas necesarias. Este sistema, acompañado de un buen equipo de desarrollo y un buen equipo de especialistas en seguridad cumpliría con las expectativas para una empresa media a la que supone un gran coste la contratación de SIEMs como QRadar o ArcSight.

Se ha visto también como puede ir esto acompañado de una aplicación propia que extienda las posibilidades que nos da Kibana de base, si bien realmente esta es una parte del SIEM que requiere de mucho más desarrollo ya que son funcionalidades más complejas de desarrollar si se está buscando crear una aplicación a nivel empresarial, tanto en control de errores, porque no nos podemos permitir que haya fallos en una aplicación de esta índole, como en complejidad de desarrollo.

Bibliografía

[1] https://www.elastic.co/guide/en/elastic-stack/current/installing-elastic-stack.html

[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html

[3] https://www.elastic.co/guide/en/elasticsearch/reference/current/important-settings.html

[4] https://www.elastic.co/guide/en/kibana/6.4/deb.html

[5] https://www.elastic.co/guide/en/kibana/6.4/settings.html

[6] https://www.elastic.co/guide/en/logstash/6.4/installing-logstash.html

[7] https://www.elastic.co/guide/en/logstash/6.4/running-logstash.html

[8] https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-elastic-stack.html

[9] https://www.elastic.co/guide/en/beats/filebeat/6.4/filebeat-installation.html

[10] https://help.ubuntu.com/community/UFW

[11] https://qbox.io/blog/logstash-grok-filter-tutorial-patterns

[12] https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns

[13] https://grokconstructor.appspot.com/RegularExpressionSyntax.txt

[14] https://www.elastic.co/guide/en/elasticsearch/client/java-api/6.4/index.html

[15] https://www.techopedia.com/definition/27471/address-resolution-protocol-poisoning-arp-poisoning

[16] https://www.ibm.com/es-es/marketplace/ibm-qradar-siem

[17] https://www.splunk.com/

[18] https://www.alienvault.com/

Anexos

El siguiente código QR contiene un enlace a un video donde se muestra la configuración de ELK desde cero:

Glosario

ciberseguridad o seguridad de tecnología de la información, es el área relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional, especialmente, la información contenida en un equipo o circulante a través de una red de equipos. 1

drop En términos de Firewalls hace referencia a cuando una conexión entrante o saliente ha sido bloqueada. 19

log o "registro" en español, es un archivo de texto en el que constan cronológicamente los acontecimientos que han ido afectando a un sistema informático, así como el conjunto de cambios que estos han generado. 2, 16

Open Source Es una expresión de la lengua inglesa que pertenece al ámbito de la informática. Se califica como open source a los programas informáticos que permiten el acceso a su código fuente. 4

query o cadena de consulta, es un término informático que se utiliza para hacer referencia a una interacción con una base de datos. 24, 25

Siglas

HTTP HyperText Transfer Protocol. 8, 9

JSON JavaScript Object Notation. 8, 9, 24, 29

SIEM Security Information and Event Management. 1, 2, 5–8, 21, 23, 26, 28–31

SSH Secure SHell. 14, 15, 19

UX User Experience. 27