

Trabajo Fin de Master

> The Watcher...

Sistema SIEM usando tecnologías
Blockchain y Open Source

Master en Ciberseguridad IMF Business School – Universidad
Camilo José Cela

Autor: Sergio Muñoz Gamarra

Tutor: Manuel Carpio Cámara

Tabla de contenido

1.	Introducción	6
1.1-	Motivación	6
1.2-	Objetivos del proyecto	6
1.3-	¿Qué es SIEM?	8
1.4-	¿Qué Blockchain?	8
2.	Requisitos del sistema	11
2.1	Requisitos técnicos.....	11
2.2	Requisitos funcionales.....	12
3.	Arquitectura.....	15
3.1-	Arquitectura a alto nivel.....	16
3.2-	Diseño de recolección de datos – Syslog-ng	17
3.3-	Diseño de colas de datos – Apache Kafka	18
3.4-	Diseño de motor de Eventos – Talend	19
3.5-	Diseño de datos – Elastic Search.....	20
3.6-	Diseño de reporting y cuadros de mando – Kibana	22
3.7-	Diseño de datos – Hyperledger Fabric	23
4.	Test del sistema.....	25
4.1.	Creación del entorno de prueba	25
4.1-1.	Instalación y configuración del recolector	27
4.1-2.	Instalación y configuración del centralizador de logs	31
4.1-3.	Instalación y configuración de la base de datos.....	34
4.1-4.	Desarrollo de proceso de normalización e inserción	35
4.1-5.	Configuración e instalación del visualizador	41
4.2-	Visualización de los eventos.....	42
4.3-	Creación de reglas	43
5.	Conclusiones.....	47
5.1-	Mejoras	47
5.2-	Comparativa con otros sistemas	48
5.3-	Conclusiones finales	54
6.	Referencias	56

Lista de tablas

Figure 1 - Modelo colaborativo.....	7
Figure 2 - Primer requisito funcional.....	12
Figure 3 - Primer requisito funcional - Segundo requisito funcional	13
Figure 4 - Diagrama de la arquitectura	15
Figure 5 - Diagrama mapa red.....	25
Figure 6 - Diagrama de entorno simulado	26
Figure 7 - Captura ejecución Zookeeper	32
Figure 8 - Captura ejecución servidor Kafka	33
Figure 9 - Captura prueba consumidor de eventos Kafka.....	34
Figure 10 - Ejecución servidor Elastic Search	34
Figure 11 - Contexto job de Talend	36
Figure 12- Job de Talend completo	36
Figure 13 - Componente consumidor de Kafka en Talend.....	37
Figure 14 - Componente tKafkaInput.....	38
Figure 15 - Parseo de evento de Kafka.....	39
Figure 16 - Convertiendo el evento en JSON	39
Figure 17 - Llamada a la API de Elastic Search	40
Figure 18 - Job lector de Kafka en ejecución.....	40
Figure 19 - Indices creados.....	41
Figure 20 - Dashboard Kibana	42
Figure 21- Filtro en syslog-ng	49
Figure 22 - Filtro en rsyslog	49
Figure 23 - Comparativa de motores de búsqueda Elastic Search [8]	51
Figure 24 - Cuadrante Mágico de herramientas de integración de datos [9]	52
Figure 25 - Comparativa entre Ethereum, Hyperledger Fabric y R3 Corda [10]	53
Figure 26- Comparativa entre diversos mecanismos de consenso.....	53

Abstract

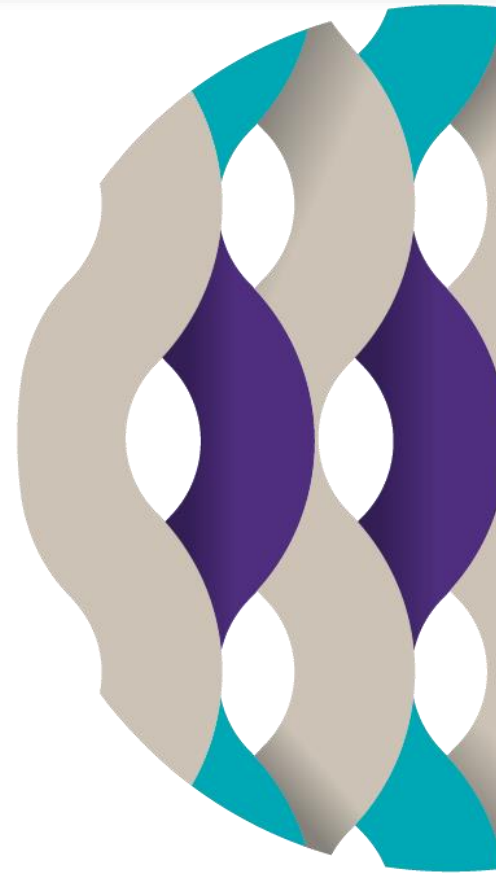
Las tecnologías SIEM (Security Information and Event Management) son ampliamente utilizadas en muchas organizaciones para detectar y gestionar eventos de seguridad en todos sus sistemas de manera global y práctica [1].

En este proyecto se expondrán los requisitos, arquitectura, desarrollo e implementación de un sistema SIEM basado en tecnologías de código abierto (Syslog, Talend, Kafka, Elastic Search, Kibana) y tecnologías Blockchain (Hyperledger Fabric, Hyperledger Composer). También se mostrarán ejemplos del uso del sistema y qué mejoras ofrece respecto a otros sistemas no basados en Blockchain (como el clásico ELK Stack[2]: Elasticsearch, Logstash, Kibana). Por último se comentarán las futuras mejoras y características pendientes de implementar en el sistema presentado.

Este proyecto está englobado en dos TFM, el presente y el TFM de Francisco Serrano. En este documento se expondrá la parte desarrollada por mí, en la que el objetivo es la creación, propagación, ingesta y visualización de eventos. Para más detalle sobre la parte de creación de alertas a partir de información propagada en Blockchain por favor diríjase a la memoria del TFM de Francisco Serrano.

***Nota:** Todo el material desarrollado para este TFM se puede encontrar en el siguiente repositorio de Github: <https://github.com/frandai/watcher>

I. Introducción



1. Introducción

1.1- Motivación

En la actualidad las grandes corporaciones, tanto a nivel nacional como internacional, reciben cientos de ciberataques diarios, por lo que todas las grandes empresas tienen robustos sistemas de seguridad, los cuales están preparados para recibir casi cualquier tipo de ataque minimizando los riesgos de los mismos, pero en ocasiones los ataques recibidos por estas grandes corporaciones son ataques poco frecuentes para los que no están preparados. Un claro ejemplo de esta situación es el reciente ataque 'Wannacry' [1], basado en ransomware, el cual aprovechaba una vulnerabilidad en sistemas Windows para cifrar un gran porcentaje de ficheros del sistema operativo y pedir un rescate por ellos. Lo particular de este caso es que el ciberataque se hizo público y sirvió para que el resto de grandes corporaciones se preparasen para minimizar los riesgos de este tipo de ataque y parchearan todos sus sistemas Windows.

Pero, ¿cuántos miles de ataques que reciben estas grandes empresas no se hacen públicos para no dañar la imagen corporativa?, ¿mejoraría la ciberseguridad de todas las empresas si hubiese un entorno colaborativo en el que todas las empresas pudiesen publicar anónimamente los ataques que reciben para que el resto de las empresas que formasen este consorcio se preparasen para recibir los mismos ataques? En ese punto es donde se encuentra la parte débil y que proponemos potenciar con 'The Watcher'.

1.2- Objetivos del proyecto

El objetivo de este proyecto es crear una plataforma unificada de detección de ciberataques que sea lo más flexible y dinámica posible para poder dar servicio a todas las empresas que se unan al consorcio, de forma que la plataforma se pueda seguir desarrollando y evolucionando sin que suponga un gran impacto para ninguno de los integrantes de este grupo de colaboración.

Una vez definido y desarrollado el sistema de monitorización, basado en tecnologías para la ingesta y captación de eventos, entraría en juego la anonimación de las empresas que reciben los ciberataques a través de Blockchain, de forma que todos los ciberataques puedan ser compartidos entre todos los miembros del grupo de forma anónima sin el riesgo reputacional que supondría hacerlos públicos. Adicionalmente todos los miembros del grupo de colaboración podrán recibir qué tipos de ataque están recibiendo sus compañeros de consorcio de forma que podrán prepararse para minimizar los efectos de futuros ataques similares a su entidad. Para llegar a dicho objetivo se requiere de la implantación de una plataforma SIEM, que además será anonimizada y distribuida a través de Blockchain.

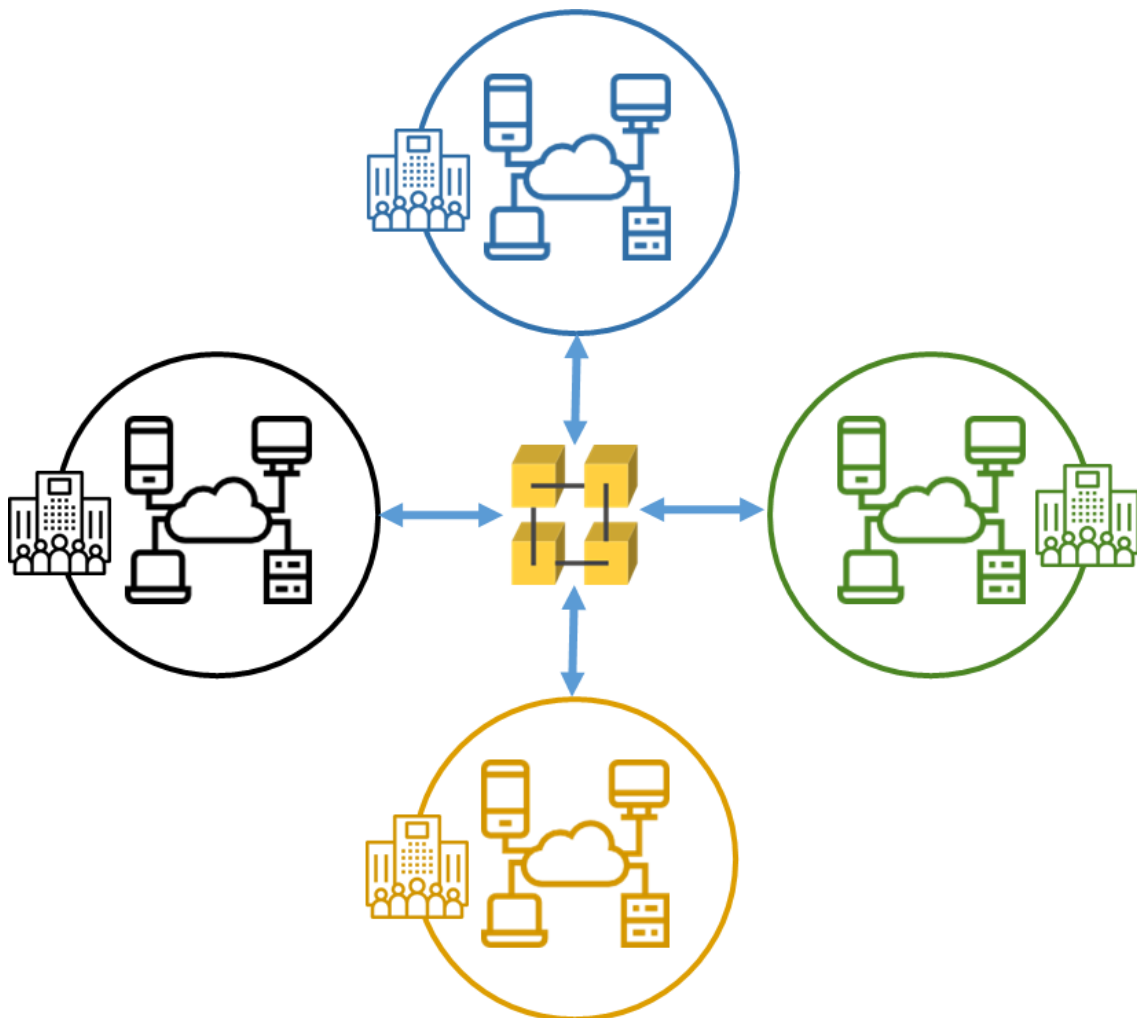


Figure 1 - Modelo colaborativo

1.3- ¿Qué es SIEM?

Un SIEM (Security Information Event Management) es un sistema, que desde el punto de vista empresarial, busca la centralización de la información sobre potenciales amenazas, a través de la estandarización de datos y la priorización de las amenazas. Todo esto es posible a través de un análisis centralizado de los datos de seguridad, obtenidos desde múltiples sistemas. [3]

Las funcionalidades básicas de una solución SIEM son:

- Centralizar la vista de potenciales amenazas
- Determinar qué amenazas requieren resolución y cuáles son solamente ruido
- Escalar temas a los analistas de Seguridad apropiados, para que puedan tomar una acción de manera ágil
- Documentar, en un registro de auditoría, los eventos detectados y cómo fueron resueltos
- Cumplir con las regulaciones de la industria en un formato de reporte sencillo

1.4- ¿Qué Blockchain?

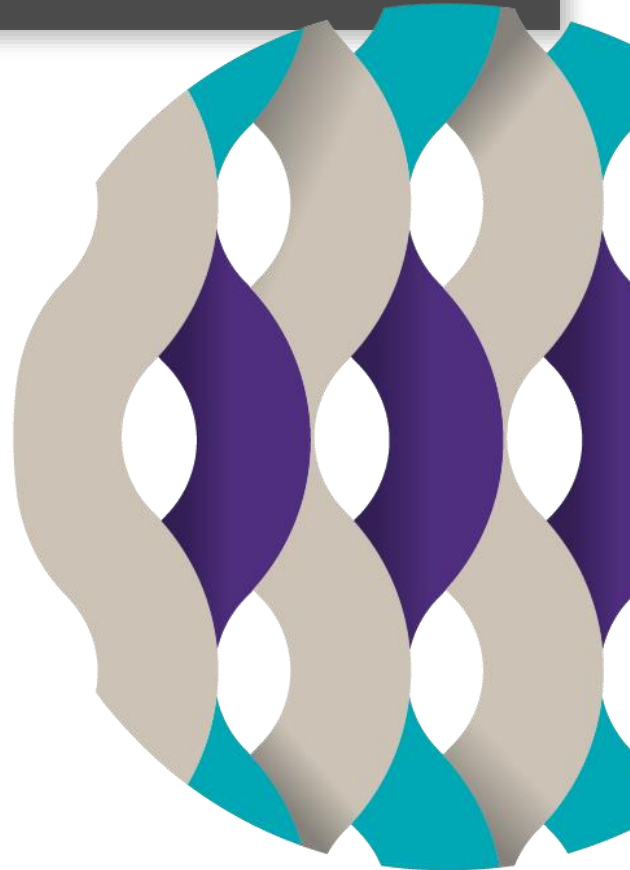
Es esencialmente una cadena de bloques en vivo que registra las transacciones de un 'token' organizadas en lotes de datos llamados 'bloques' que utilizan la validación criptográfica para unirse entre sí. En pocas palabras, cada bloque de datos hace referencia e identifica el bloque anterior mediante una función 'hash', formando una cadena ininterrumpida, de ahí el nombre.

Este enfoque para registrar datos ofrece una ventaja significativa sobre las cadenas de bloques y las bases de datos tradicionales. Al hacer que cada bloque de datos valide su predecesor directo, la cadena cada vez más largo es estrictamente secuencial y permanente: no es posible modificar, enmascarar o eliminar

transacciones que ya se han registrado. Cualquier intento de hacerlo rompería la cadena criptográfica y se marcaría de inmediato a todos los participantes.

En principio, un Ledger blockchain es una base de datos a prueba de manipulaciones con validación incorporada. [4]

II. Requisitos



2 – Requisitos del sistema

Antes de entrar al detalle de la arquitectura cabe destacar que el desarrollo de este proyecto en conjunto tiene dos partes, la captación de la información y su normalización, que es la parte en la que se centra mi TFM, y por otra parte la creación de alertas y posterior distribución a través de la Blockchain, desarrollado en el TFM de Francisco Serrano.

2.1 Requisitos técnicos

Antes de definir la arquitectura que debemos utilizar se debe definir las características que debe poseer nuestra herramienta:

- Debe detectar ciberataques en tiempo real
- Debe estar preparado para eventos con baja latencia, near realtime
- Alta eficiencia en consultas
- Integración nativa con una herramienta de reporting, que nos permita visualizar la información contenida en la base de datos
- Sistema Blockchain distribuido, en el que se puedan crear redes privadas y tenga unos ratios de propagación cercanos a 100% y que nos permita realizar inserciones y consultas en tiempo real de manera eficiente.

A partir de estos requisitos podemos ver que en nuestro proyecto entran en juego tres importantes ramas de las ciencias de la computación:

- **Big Data:** Parte fundamental para la extracción, ingesta y procesamiento de los datos de todas las fuentes que se definirán.
- **Ciberseguridad:** Imprescindible los conocimientos sobre posibles ataques, así como cómo detectarlos y que fuentes de datos obtener para enriquecer la información sobre el ataque. Adicionalmente y de forma conjunta con Big Data, las tecnologías SIEM jugarán un papel fundamental en la captación de la información de logs, necesarios para detectar amenazas.
- **Blockchain:** Dentro de la blockchain se parametrizarán todas las reglas de conversión de eventos en alertas, así como todas las alertas para poder propagarlas al resto de nodos, haciendo que lleguen así a todas las corporaciones que formen para del grupo de colaboración.

2.2 Requisitos funcionales

Desde un punto de vista funcional los datos que manejará el sistema deberán seguir los siguientes pasos:

- Desde el punto de vista del generador de eventos y alertas:

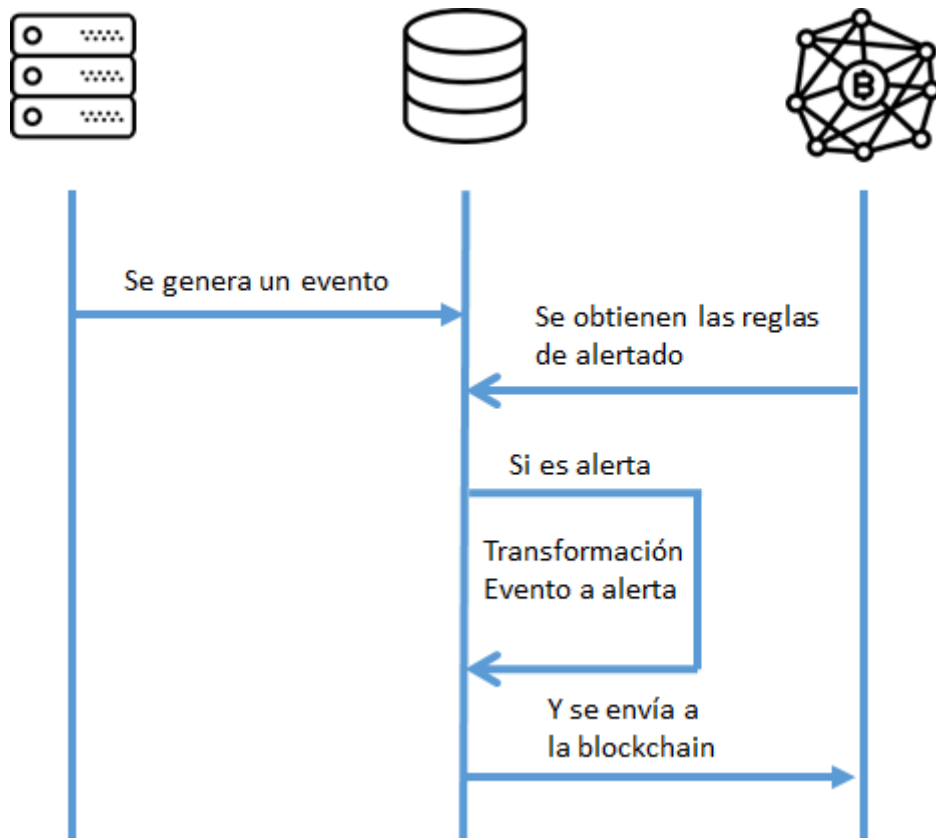


Figure 2 - Primer requisito funcional

- Desde el punto de vista del receptor de eventos



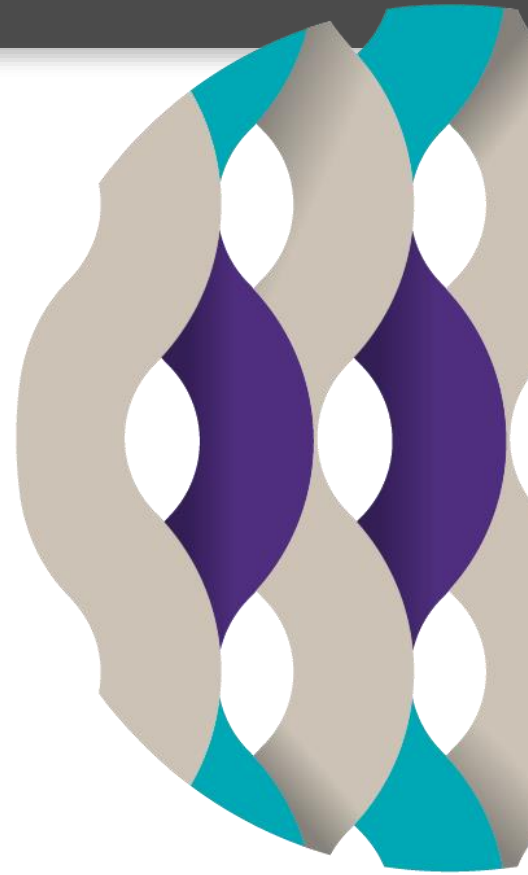
Figure 3 - Primer requisito funcional - Segundo requisito funcional

En este punto cabe destacar que se va a trabajar sobre dos conceptos independientes:

- Eventos: unidad atómica de nuestro sistema. Se refiere a cualquier acción que se quiera monitorizar sin que sea necesariamente un ciberataque. La agregación de varios eventos puede conllevar en una alerta.
- Alerta: evento o grupos de eventos que a través de una agregación o regla se consideran preocupantes siendo posible que se trate de un ciberataque.

Expuestos los requisitos funcionales cabe destacar que los eventos se quedarán en el entorno corporativo. Una vez que el evento cumpla alguna de las reglas de alertado se generará una alerta que si será propagada al resto de corporaciones que forman parte del consorcio. En resumen, los eventos se quedarán en el ámbito privado de cada empresa y la única información que se compartirá con el resto de empresas serán las alertas de ciberataques de forma anónima y sin exponer ni IPs ni información sobre usuarios.

III. Diseño del sistema



3. Arquitectura

- Para cubrir dichos requisitos, tanto técnicos como funcionales se ha propuesto la siguiente arquitectura:

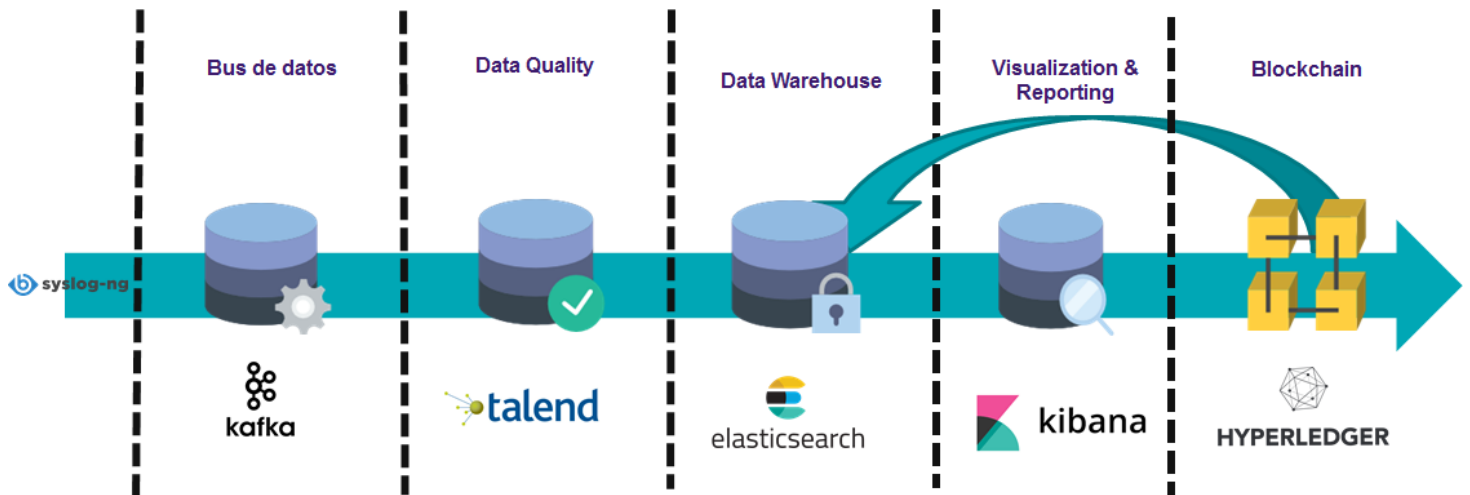


Figure 4 - Diagrama de la arquitectura

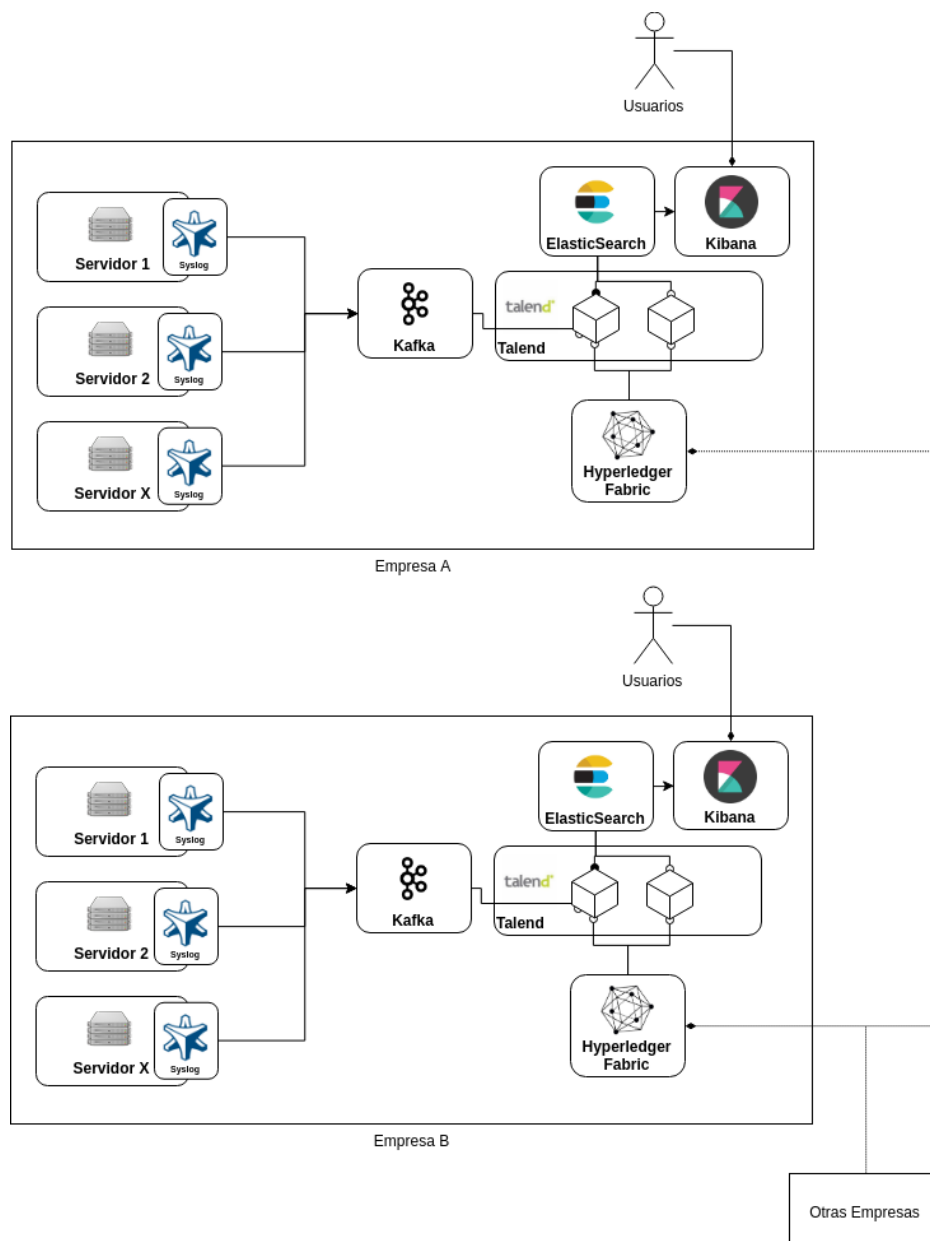
Dicha arquitectura se ha basado en 6 capas:

- **Recolección de logs:** Donde se tendrán agentes en los distintos equipos a monitorizar y se enviarán a un centralizador de información. La herramienta que se utilizará para tal fin es Syslog-ng.
- **Bus de datos:** Donde se recibirá la información del agente desplegado en las máquinas y se pondrá a disposición para ser consumida en tiempo real. La herramienta seleccionada es Apache Kafka.
- **Data Quality:** En este punto se recolectará la información y se realizará un procesamiento masivo para la normalización y estandarización de la información de cara a la generación de alertas. Para dicho fin se ha seleccionado Talend.
- **Data Warehouse:** Donde persistirá la información, con el fin de almacenar un histórico de datos para posibles análisis posteriores, así como para la creación de reporting. En este caso la herramienta que se ha utilizado es Elasticsearch.

- **Visualizacion y reporting:** Herramienta donde se pondrán ver tanto los eventos como las alertas que se generen. Para la visualización de datos se ha elegido Kibana.
- **Blockchain:** Capa donde se anonimizará y se distribuirá toda la información. Para tal fin se ha seleccionado Hyperledger Fabric.

3.1- Arquitectura a alto nivel

Una vez comentadas las tecnologías a usar en el apartado anterior, se puede explorar cómo interaccionan a través de este esquema:



Usando el diagrama se puede ver que cada servidor o dispositivo que se desee monitorizar tendrá instalado un agente de Syslog que estará configurado para capturar eventos del sistema y enviarlos al topic definido en Kafka. Este topic es leído por procesos desarrollados en Talend y desde Talend la información es distribuida en ElasticSearch (repositorio privado) y Hyperledger Fabric (Sistema Blockchain compartido con otras organizaciones).

Vamos a proceder ahora a explicar más en profundidad la funcionalidad de cada tecnología dentro del sistema y por qué ha sido elegida.

3.2- Diseño de recolección de datos – Syslog-ng



Syslog-ng es una herramienta de captura y envío de logs. La forma en la que funciona es a través de un servidor central al que se envía toda la información que se recolecta en el resto de dispositivos monitorizados a través del despliegue de un agente, en el cuál simplemente tendremos que señalar que logs queremos enviar y a que máquina.

En dicho fichero de configuración tan solo hay que definir 4 propiedades:

- Source: en el que hay que señalar el fichero que queremos que sea monitorizado.
- Filter: en el que definimos cada línea que queremos que nos llegue a partir de un patrón o expresión regular.
- Destination: Donde queremos que se envíe la información que estamos compartiendo con el servidor syslog.
- Log: En este componente lo único que hay que definir es la relación entre sources, filters y destinations.

Esta tecnología ha sido elegida por los siguientes motivos:

- Dispone de una versión open source para servidores UNIX que no conlleva ningún cargo en licencias. Respecto a la centralización y control de logs hay pocas herramientas en el mercado que sean Open Source.
- Envío de datos en tiempo real.
- Fácil de implementar en cualquier servidor, ya que tan solo hay que desplegarlo y modificar un fichero de configuración estandarizado para que envíe la información al servidor donde se centraliza la información.
- Dispone de un largo listado de conectores para destinos, entre ellos el de Kafka, que pocas herramientas similares poseen.

La versión que se está utilizando es la 3.13.

3.3- Diseño de colas de datos – Apache Kafka



Apache Kafka es un bus para la centralización de información. Básicamente funciona como un gestor de información que pone a disposición de consumidores la información que, desde varios puntos, se le envía.

Kafka dispone de dos modalidades:

- Información en tiempo real, que básicamente basa su funcionamiento en colas FIFO, de modo que cuando el evento más antiguo que hay en la cola es consumido desaparece de la misma y no puede ser recuperado.
- Pool, en el que se pueden guardar ciertos archivos que estarán disponibles de forma íntegra durante un tiempo, que debe haber sido definido en la política de retención de la información.

En nuestro caso la modalidad de uso de Kafka que se ajusta a las necesidades es la ingesta y procesamiento de la información en tiempo real, de modo que lo que se hará será habilitar un puerto de entrada para que Kafka reciba información desde todos los agentes de syslog-ng desplegados y se creará un topic en el que se podrá encontrar dicha información lista para el consumo.

Adicionalmente es habitual que se realice una réplica de la cola en producción para garantizar la alta disponibilidad, pero en este caso y por temas de recursos no se ha replicado la cola y se dispone únicamente de una cola.

3.4- Diseño de motor de Eventos – Talend



Talend Open Studio es una herramienta ETL donde se podrán llevar a cabo las siguientes tareas:

- Extracción: Con Talend consumiremos los eventos que están encolados en Kafka. A partir del conector de Kafka podremos realizar la conexión a los diferentes nodos de Kafka en tiempo real, donde el delay de procesamiento está casi despreciable.
- Transformación: Una vez se consume el evento desde Talend realizaremos el parseo de la información para obtener la información que necesitamos para generar los eventos orientados a alertas. Además se realizarán procesos de enriquecimiento de información.
- Load: Finalmente se cargarán todos los eventos en nuestra base de datos, Elasticsearch, a través de peticiones a la API de ingesta de Elasticsearch.

3.5- Diseño de datos – Elastic Search



Como base de datos se ha utilizado Elastic Search. Esta base de datos ha sido seleccionada por las siguientes razones:

- Es un sistema de almacenamiento distribuido, lo cual nos permite asegurar la disponibilidad de los datos.
- Multitenencia de datos: Lo cual nos permite trabajar con diferentes índices a la vez independientemente de sus campos, lo cual agiliza y hace más eficientes las búsquedas.
- Acceso en tiempo real: Está preparada para operar en transmisiones de baja latencia, por lo que podremos consultar la información en tiempo real.
- Escalabilidad horizontal: A partir de esta propiedad siempre se podrá escalar el sistema de forma sencilla cuando la cantidad de datos sea demasiado grande para los recursos actuales del sistema.
- Integración nativa con Kibana: Al tener una herramienta de reporting integrada de forma nativa facilita la visualización de la información.

Dentro de esta base de datos tendremos guardada la siguiente información:

- Eventos:
 - o A través de Syslog-ng se recogerán los eventos que los diferentes sistemas generen, lo cuales serán transformados por Talend e insertados en Elastic Search. Estos eventos son eventos simples, que pueden ser o no una amenaza, es decir son eventos atómicos, donde desde la agregación de los mismos se podrán generar las alertas. Los eventos tiene un identificador de evento compuesto que se genera a partir de dos características:
 - Aplicación: Se refiere a la fuente de los datos o la naturaleza de los mismos. Durante el desarrollo de este TFM se han generado eventos sobre las siguientes aplicaciones:

- OS: Operation System. Eventos que se generan directamente desde el sistema operativo, por ejemplo la creación de un fichero.
- SN: SNMP (Simple Network Management Protocol). Eventos generados a través del protocolo SNMP.
- RO: Root. Eventos relacionados a operaciones que se ejecutan con el usuario root de la máquina.
- Tipo de evento:
 - AC: Acceso a ficheros con información sensible. En este caso se monitoriza la ruta /etc.
 - LA: Login Attempt. Intento de login erróneo en la aplicación.
 - CR: Creación de un fichero sospechoso.
 - SE: Sesión. Inicio de sesión con un usuario con privilegios altos, en este caso root.
- A partir de la información citada podemos encontrar los siguientes eventos en nuestra base de datos de eventos:
 - ROSE: Inicio de sesión con root.
 - SNLA: Intento erróneo de login, monitorizado a través del protocolo SNMP.
 - OSCR: Creación de un fichero sospechoso en sistema operativo.
 - OSAC: Lectura de un fichero con información sensible.
- Para guardar toda la información de la manera más eficiente posible el modelo de datos que se va a crear es el siguiente:
 - Index: Para cada tipo de evento existe un index, y el que contendrá en el nombre la fecha de monitorización del evento, de este modo tendremos un índice nuevo cada día por cada tipo de evento donde el nombre seguirá el siguiente patrón: 'event-yyyy-mm-dd-event-type' (por ejemplo: event-2018-10-25-osac).
 - Type: Todos los índices dispondrán de un tipo común, que será 'event'.

- Documents: Todos los documents, o lo que en SQL se conoce como filas, tendrán los siguientes campos en común:
 - Event_date: Fecha en la que se generó el evento
 - Monitorization_date: Fecha en la que se registro el evento en nuestro sistema.
 - Event_type: Tipo de evento.
 - App: Aplicación que ha generado el evento.
 - Source_host: Máquina en la que se ha generado el evento.

Además de estos campos comunes cada tipo de evento puede disponer de campos extra de información con información específica del evento.

- Alertas: A partir de los eventos se realizarán funciones de agregación, que darán como resultado alertas relativas a posibles ataques. Para más detalle puede verlo en el TFM de Francisco Serrano.

3.6- Diseño de reporting y cuadros de mando – Kibana



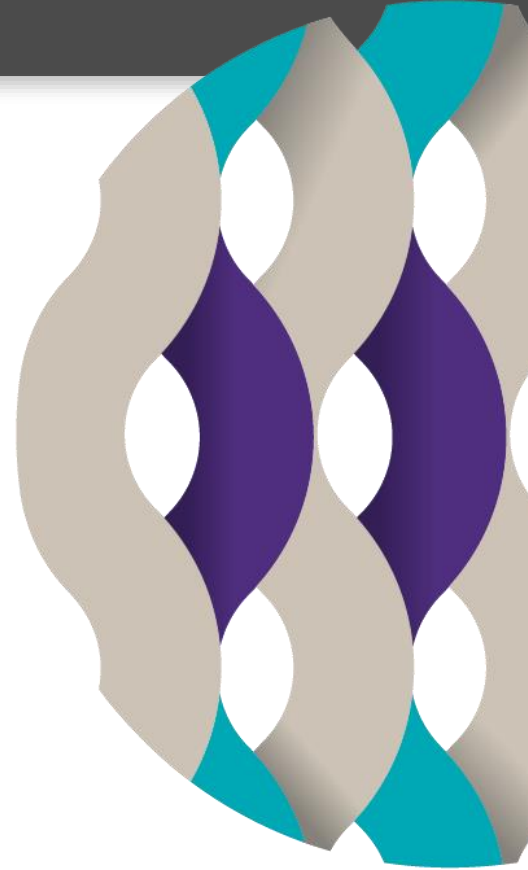
Como herramienta de reporting y creación de informes la seleccionada ha sido Kibana, principalmente por su integración nativa con Elastic Search y su facilidad para crear reportes. Como resultado de la parte de generación de eventos se ha desarrollado una cuadro de mandos donde se podrá ver cómo han ido evolucionando los eventos creados en el tiempo, así como los eventos más frecuentes y su detalle.

3.7- Diseño de datos – Hyperledger Fabric

La creación de alertas se ha centralizado principalmente en Hyperledger, para que las reglas de alertado puedan ser comunes y cada corporación pueda elegir cuales de ellas les son más útil o preocupantes y cuáles de ellas no aplican para su caso. A partir de esas reglas desde Talend se realizarán consultas a las mismas y se lanzarán de forma recursiva cada minuto un job que procese los nuevos eventos y si son resultantes en una alerta.

Para más detalle puede verlo en el TFM de Francisco Serrano.

IV. Test del sistema



4. Test del sistema

4.1. Creación del entorno de prueba

Con el fin de probar dicho sistema se ha montado y desarrollado un entorno de prueba, donde se ha simulado el comportamiento real de varias máquinas de una empresa para la generación de eventos y alertas que se puedan ajustar a la realidad. En esta simulación se han instalado y configurada todas las herramientas anteriormente citadas, de forma que se pudiese realizar el proceso end-to-end.

La arquitectura física que tendría la red de máquinas que vamos a simular sería similar a la siguiente:

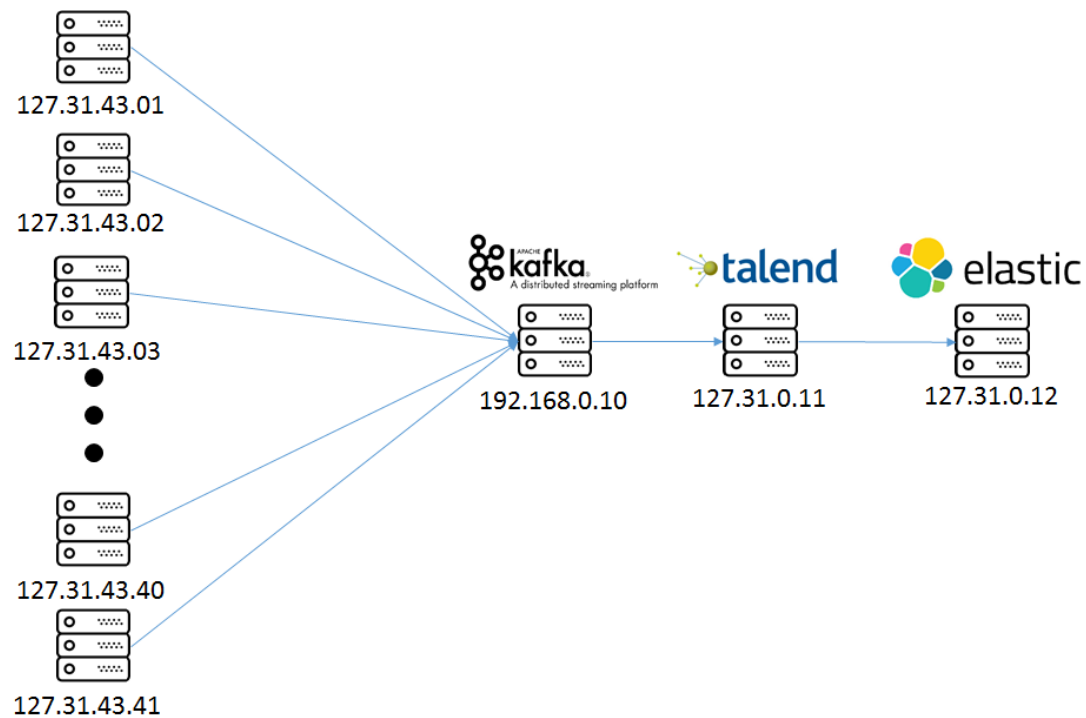


Figure 5 - Diagrama mapa red

En este caso tan solo se han simulado el comportamiento de algunas de las máquinas, siendo su comportamiento extensible a cualquier máquina de la red.

Esta arquitectura es la arquitectura común de cualquier corporación y es extrapolable a cualquier otra empresa. Por lo que el entorno simulado de pruebas tendría una estructura real como la que aparece en la siguiente figura:

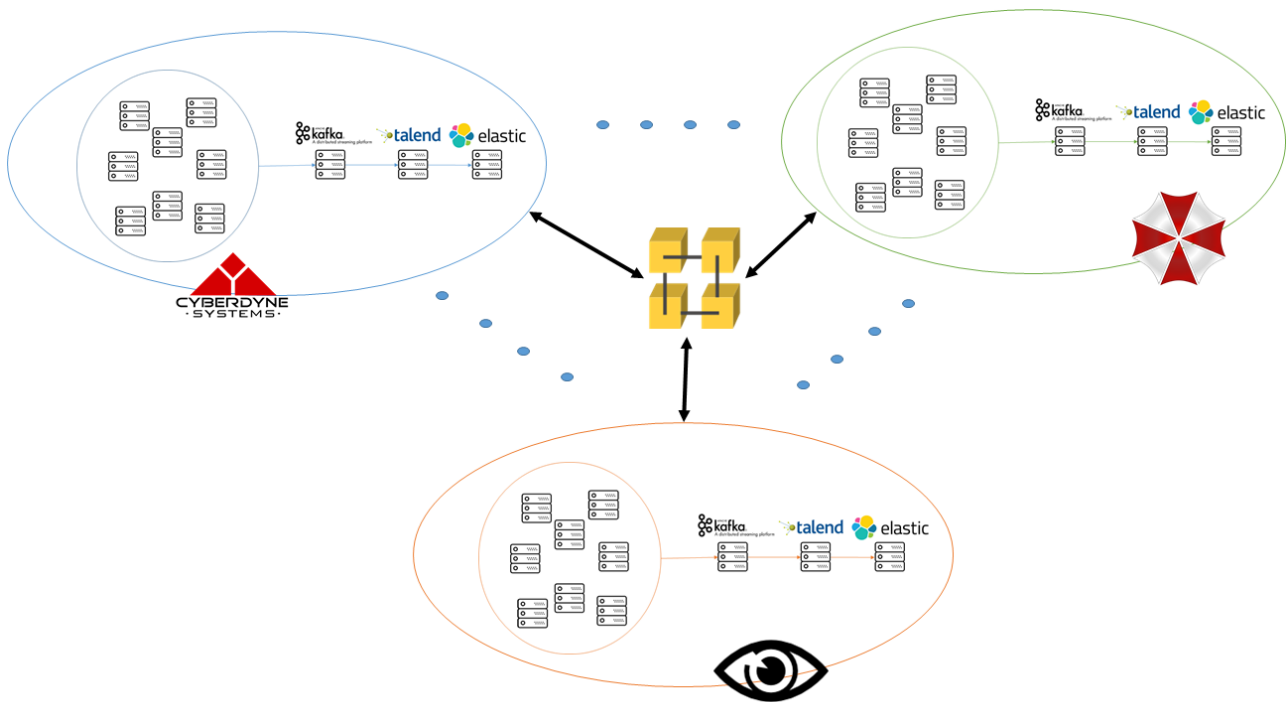


Figure 6 - Diagrama de entorno simulado

En ella se van a simular los sistemas de 'Watcher', Cyberdyne Systems y Umbrella Corp.

4.1-1. Instalación y configuración del recolector

La primera pieza de nuestro sistema es Syslog, el cuál funciona como recolector de logs para centralizarlos, pero en primer lugar se han desarrollado algunos scripts y técnicas de generación de logs con para poder generar los eventos. Dichos métodos de creación de eventos son los siguientes, según los eventos listados en la sección de diseño de datos en Elastic Search:

- ROSE: Inicio de sesión con root.

Para recolectar esta información no ha sido necesario desarrollar ningún proceso de creación de log, ya que el SO ya genera el log donde aparece esta información. Dicho log se encuentra en la ruta `/var/log/auth.log` en los sistemas UNIX. De este modo la configuración de la sonda de syslog-ng quedaría de la siguiente manera:

```
source auth_log_s {
    file("/var/log/auth.log");
};

destination d_kafka_auth {
    python(
        class("syslogng_kafka.kafkadrivers.KafkaDestination")
        on-error("fallback-to-string")
        options(
            hosts("192.168.0.10:9092")
            topic("watcher_topic")
        )
    );
};

filter watcher_root_access {
    match("session opened for user root" value("MESSAGE"));
};

log {
    source(auth_log_s);
    destination(syslog_to_kafka);
    filter(watcher_root_access);
};
```

- SNLA: Intento erróneo de login, monitorizado a través del protocolo SNMP.

En este caso tan solo hay que habilitar la escritura de los SNMP traps a un fichero. Esta configuración viene por defecto en la instalación de SNMP y tan solo se tendría que activar en caso de hubiese sido desactivada previamente. El agente de syslog-ng quedaría de la siguiente manera:

```
source snmp_log_s {
    file("/var/log/messages");
};
destination d_kafka_snmp {
    python(
        class("syslogng_kafka.kafkadriver.KafkaDestination")
        on-error("fallback-to-string")
        options(
            hosts("192.168.0.10:9092")
            topic("watcher_topic")
        )
    );
};
filter watcher_root_access {
    match("session opened for user root" value("MESSAGE"));
};
log {
    source(auth_log_s);
    destination(syslog_to_kafka);
    filter(watcher_root_access);
};
```

- o OSCR: Creación de un fichero sospechoso en el sistema operativo.

En este caso habrá que crear un proceso de monitorización a través de 'inotify', que se configurará para ser lanzar en el arranque del sistema. El script simplemente detectará la creación de un fichero y lo volcará en un log.

El script para dicha tarea es el siguiente:

```
#!/bin/bash

inotifywait -m / -e create -e moved_to |
while read path action file; do
    echo "SO;$file;$path;$action;" > /var/log/watcher_so.log
done
```

A partir de dicho script se genera el log en la ruta indicada, quedando el agente de la siguiente manera:

```
source create_file_s {
    file("/var/log/messages");
};

destination d_kafka_create_file {
    python(
        class("syslogng_kafka.kafkadrivers.KafkaDestination")
        on-error("fallback-to-string")
        options(
            hosts("192.168.0.10:9092")
            topic("watcher_topic")
        )
    );
};

filter f_danger_file {
    match(".wnc" value("MESSAGE"));
};
```

```
log {
    source(create_file_s);
    destination(d_kafka_create_file);
    filter(f_danger_file);
};
```

- OSAC: Lectura de un fichero con información sensible.

Finalmente para la detección de lectura de un fichero se ha creado un proceso similar al anteriormente citado, pero habilitando los eventos de lectura únicamente, y se monitoriza el contenido de la carpeta /etc, el cuál es sensible. El script queda de la siguiente forma:

```
#!/bin/bash

inotifywait -m /etc -e access |

while read path action file; do
    echo "S0;$file;$path;$action;" > /var/log/watcher_read.log
done
```

Y el agente del syslog-ng será el siguiente:

```
source read_file_s {
    file("/var/log/watcher_read.log");
};

destination d_kafka_read_file {
    python(
        class("syslogng_kafka.kafkadrivere.KafkaDestination")
        on-error("fallback-to-string")
        options(
            hosts("192.168.0.10:9092")
            topic("watcher_topic")
        );
    };
};
```

```
log {  
    source(read_file_s);  
    destination(d_kafka_read_file);  
};
```

Tras tener los procesos creados e implantados en las máquinas a monitorizar lo único que faltaría sería instalar syslog-ng en las máquinas, a través del comando `apt-get install syslog-ng`. Tras la instalación el último paso será modificar el fichero `/etc/syslog-ng/syslog-ng.conf` y añadir al final del mismo los componentes del agente que anteriormente hemos listado, de forma que todos los sources, filter, destinations y logs estén contenidos en el mismo.

4.1-2. Instalación y configuración del centralizador de logs

Tras la configuración de los recolectores pasamos a la instalación del centralizado de logs, Kafka. En este caso vamos a utilizar un único cluster, desde el que se centralizará toda la información. Para la instalación del mismo tan solo hemos seguido los siguientes pasos:

- Descargar los ficheros de Kafka:
https://www.apache.org/dyn/closer.cgi?path=/kafka/2.0.0/kafka_2.11-2.0.0.tgz
- Tras descargar los ficheros encontraremos dos componentes que vamos a utilizar:
 - o Zookeeper: Servicio utilizado para el control de servicios distribuidos. En este caso el papel de Zookeeper es residual, ya que en este momento el servicio distribuido al que deber coordinar, Apache Kafka, se encuentra en una instalación monoclustero. Pero es una herramienta importante para futuras mejoras del sistema, como es la clusterización de Kafka.
 - o Kafka: Aquí se encuentran los scripts para inicializar y arrancar el servicio. Además se encuentra un script que inicializa un

consumidor, lo cual es de gran ayuda para comprobar que todos los eventos de syslog están llegando correctamente.

Vamos a proceder ahora a lanzar el servicio:

- 1) El primer paso tras extraer los ficheros es crear el directorio en /opt/kafka, y mover dentro de la carpeta todos los archivos que había dentro del fichero comprimido. Este paso no es obligatorio, solo se ha realizado con el fin de mantener orden.
- 2) Procedemos a lanzar zookeeper. Al lanzarlo le indicaremos que fichero de configuración queremos que utilice. En este caso utilizaremos config/zookeeper.properties. Dentro de este fichero la única propiedad importante es el puerto, que será el que zookeeper utiliza por defecto, el puerto 2181.

El comando que se utiliza para lanzar zookeeper es el siguiente:

`bin/zookeeper-server-start.sh config/zookeeper.properties`

```
logger@logger-VirtualBox: /opt/kafka$ sudo ./bin/zookeeper-server-start.sh config/zookeeper.properties
[2018-10-26 21:38:17,168] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.QuorumPeerConfig)
[2018-10-26 21:38:17,176] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DataDirCleanupManager)
[2018-10-26 21:38:17,177] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DataDirCleanupManager)
[2018-10-26 21:38:17,177] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DataDirCleanupManager)
[2018-10-26 21:38:17,177] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.QuorumPeerMain)
[2018-10-26 21:38:17,252] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.QuorumPeerConfig)
[2018-10-26 21:38:17,252] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2018-10-26 21:38:17,275] INFO Server environment:zookeeper.version=3.4.13-2d71af4dbe22557fda74f9a9b4309b15a7487f03, built on 06/29/2018 00:39 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2018-10-26 21:38:17,275] INFO Server environment:host.name=logger-VirtualBox (org.apache.zookeeper.server.ZooKeeperServer)
[2018-10-26 21:38:17,276] INFO Server environment:java.version=1.8.0_181 (org.apache.zookeeper.server.ZooKeeperServer)
[2018-10-26 21:38:17,276] INFO Server environment:java.vendor=Oracle Corporation (org.apache.zookeeper.server.ZooKeeperServer)
[2018-10-26 21:38:17,276] INFO Server environment:java.home=/usr/lib/jvm/java-8-openjdk-amd64/jre (org.apache.zookeeper.server.ZooKeeperServer)
[2018-10-26 21:38:17,278] INFO Server environment:java.class.path=/opt/kafka/bin/../libs/activation-1.1.1.jar:/opt/kafka/bin/../libs/aopalliance-repackaged-2.5.0-b42.jar:/opt/kafka/bin/../libs/args4j-0.7.0.jar:/opt/kafka/bin/../libs/audience-annotations-0.5.0.jar:/opt/kafka/bin/../libs/commons-lang3-3.5.jar:/opt/kafka/bin/../libs/connect-api-2.0.0.jar:/opt/kafka/bin/../libs/connect-basic-auth-extension-2.0.0.jar:/opt/kafka/bin/../libs/connect-file-2.0.0.jar:/opt/kafka/bin/../libs/connect-json-2.0.0.jar:/opt/kafka/bin/../libs/connect-runtime-2.0.0.jar:/opt/kafka/bin/../libs/connect-transforms-2.0.0.jar:/opt/kafka/bin/../libs/guava-20.0.jar:/opt/kafka/bin/../libs/hk2-api-2.5.0-b42.jar:/opt/kafka/bin/../libs/hk2-locator-2.5.0-b42.jar:/opt/kafka/bin/../libs/hk2-utils-2.5.0-b42.jar:/opt/kafka/bin/../libs/jackson-annotations-2.9.6.jar:/opt/kafka/bin/../libs/jackson-core-2.9.6.jar:/opt/kafka/bin/../libs/jackson-databind-2.9.6.jar:/opt/kafka/bin/../libs/jackson-jaxrs-base-2.9.6.jar:/opt/kafka/bin/../libs/jackson-jaxrs-json-provider-2.9.6.jar:/opt/kafka/bin/../libs/jackson-module-jaxb-annotations-2.9.6.jar:/opt/kafka/bin/..
```

Figure 7 - Captura ejecución Zookeeper

- 3) Una vez lanzado Zookeeper se procede a arrancar Kafka. El siguiente paso es lanzar el servidor Kafka. Al igual que Zookeeper se lanza a partir de un script y un fichero de configuración. En este caso la única modificación que debemos hacer es la de cambiar la IP de Zookeeper (zookeeper.connect=192.168.0.10:2181), para que en caso de clusterizar en


```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic watcher_topic
```

```
logger@logger-VirtualBox: /opt/kafka$ bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic watcher_topic
2018-10-24 17:54:09 127.31.42.23 50:Document1.wnc;/home/logger/test/;CREATE
2018-10-24 17:54:09 127.31.42.23 50:Document2.wnc;/home/logger/test/;CREATE
2018-10-24 17:54:10 127.31.42.23 50:Document3.wnc;/home/logger/test/;CREATE
```

Figure 9 - Captura prueba consumidor de eventos Kafka

Como se puede ver ya estamos recibiendo información desde la máquina 127.31.24.23. Por lo que ya podemos proceder a consumirlos desde Talend.

* Con el fin de lanzar los servidores y servicios comentado en segundo plano, los comandos para lanzar el servidor de Zookeeper y Kafka serán lanzados en segundo plano con: `nohup 'comando' < 2/dev/null &`

4.1-3. Instalación y configuración de la base de datos

Como ya se comentó durante el desarrollo de esta documentación la base de datos que vamos a utilizar es Elastic Search. Al igual que Kafka la instalación de Elastic Search se basa en la descarga y utilización de ficheros binarios y scripts, por lo que se seguirá el mismo proceso que con Kafka, descargándolo de la página oficial de Elastic Search (<https://www.elastic.co/downloads/elasticsearch>) y extrayéndolo en la carpeta ~/elasticsearch (en este caso no podemos guardarlo en /opt ya que Elastic Search no puede ser lanzado como root).

Una vez descargado se lanza desde línea de comandos con el siguiente comando:

~/elasticsearch/bin/elasticsearch

```
logger@logger-VirtualBox:~$ elasticsearch/bin/elasticsearch
[2018-10-27T00:00:53,027][INFO ][o.e.n.Node ] [] initializing ...
[2018-10-27T00:00:53,370][INFO ][o.e.n.NodeEnvironment ] [iKId2xa] using [1] data paths, mounts [[ /dev/sda1]], net usable_space [603.5m
b], net total_space [17.6gb], types [ext4]
[2018-10-27T00:00:53,371][INFO ][o.e.n.NodeEnvironment ] [iKId2xa] heap size [1015.6mb], compressed ordinary object pointers [true]
[2018-10-27T00:00:53,601][INFO ][o.e.n.Node ] [iKId2xa] node name derived from node ID [iKId2xaTS865Rk6UB2reJg]; set [node.name]
to override
[2018-10-27T00:00:53,602][INFO ][o.e.n.Node ] [iKId2xa] version[6.4.2], pid[10352], build[default/tar/04711c2/2018-09-26T13:34:0
9.098244Z], OS[Linux/4.15.0-36-generic/amd64], JVM[Oracle Corporation/OpenJDK 64-Bit Server VM/1.8.0_181/25.181-b13]
[2018-10-27T00:00:53,603][INFO ][o.e.n.Node ] [iKId2xa] JVM arguments [-Xms1g, -Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatin
gOccupancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djna
.nosys=true, -XX:-OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPe
rThread=0, -Dlog4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.io.tmpdir=/tmp/elasticsearch.IbNtCQGS, -XX:+HeapDumpOnOutOfMem
oryError, -XX:HeapDumpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -XX:+PrintGCDetails, -XX:+PrintGCDateStamps, -XX:+PrintTenuringDistribut
ion, -XX:+PrintGCApplicationStoppedTime, -Xloggc:logs/gc.log, -XX:+UseGCLogFileRotation, -XX:NumberOfGCLogFiles=32, -XX:GCLogFileSize=64m, -De
s.path.home=/home/logger/elasticsearch, -Des.path.conf=/home/logger/elasticsearch/config, -Des.distribution.flavor=default, -Des.distribution
.type=tar]
```

Figure 10 - Ejecución servidor Elastic Search

Una vez que tanto la base de datos como Kafka están ejecutándose correctamente vamos a proceder ahora al desarrollo de los Jobs de Talend para normalizar e insertar la información en base de datos.

4.1-4. Desarrollo de proceso de normalización e inserción

Vamos a proceder ahora al desarrollo del job de Talend que tiene dos componentes básicos:

- 1- Consumidor de Kafka: El primer elemento que debemos tener es el consumidor de eventos. Para esta tarea Talend cuenta con un conector, el cual nos permitirá consumir eventos de nuestro Kafka. Una de las cosas a tener en cuenta que, por defecto y sin posibilidad de ser modificado, el consumidor devuelve cada evento que hay en Kafka como un string completo, sin columnas, por lo que lo primero que tenemos que hacer es parsear dicho string.
- 2- Inserciones en BD: El método de comunicación con Elastic Search es a través de la API de Elastic Search, en la que básicamente tendremos que hacer llamadas siguiendo el siguiente patrón, que se ajusta al modelo de datos explicado en el apartado 3.4:

<http://192.168.0.12:9200/index/type>

El método para insertar la información a través de la API será 'POST'.

Toda la información de las conexiones a los diferentes componentes se parametriza a través del context de Talend, en el que definiremos los diferentes entornos: Desarrollo, Preproducción y Producción.

				Desarrollo			Preproduccion		Produccion	
	Name	Type	Comment	Value			Value		Value	
1	port_es	int Integer		9200	<input type="checkbox"/>		9200	<input type="checkbox"/>		<input type="checkbox"/>
2	host_es	String		localhost	<input type="checkbox"/>		192.168.0.12	<input type="checkbox"/>		<input type="checkbox"/>
3	port_kafka	String		"9092"	<input type="checkbox"/>		"9092"	<input type="checkbox"/>		<input type="checkbox"/>
4	host_kafka	String		localhost	<input type="checkbox"/>		192.168.0.10	<input type="checkbox"/>		<input type="checkbox"/>
5	port_zoo	String		"2181"	<input type="checkbox"/>		"2181"	<input type="checkbox"/>		<input type="checkbox"/>
6	host_zoo	String		localhost	<input type="checkbox"/>		192.168.0.10	<input type="checkbox"/>		<input type="checkbox"/>

Figure 11 - Contexto job de Talend

De este modo y teniendo esta información el job ha quedado de la siguiente forma:

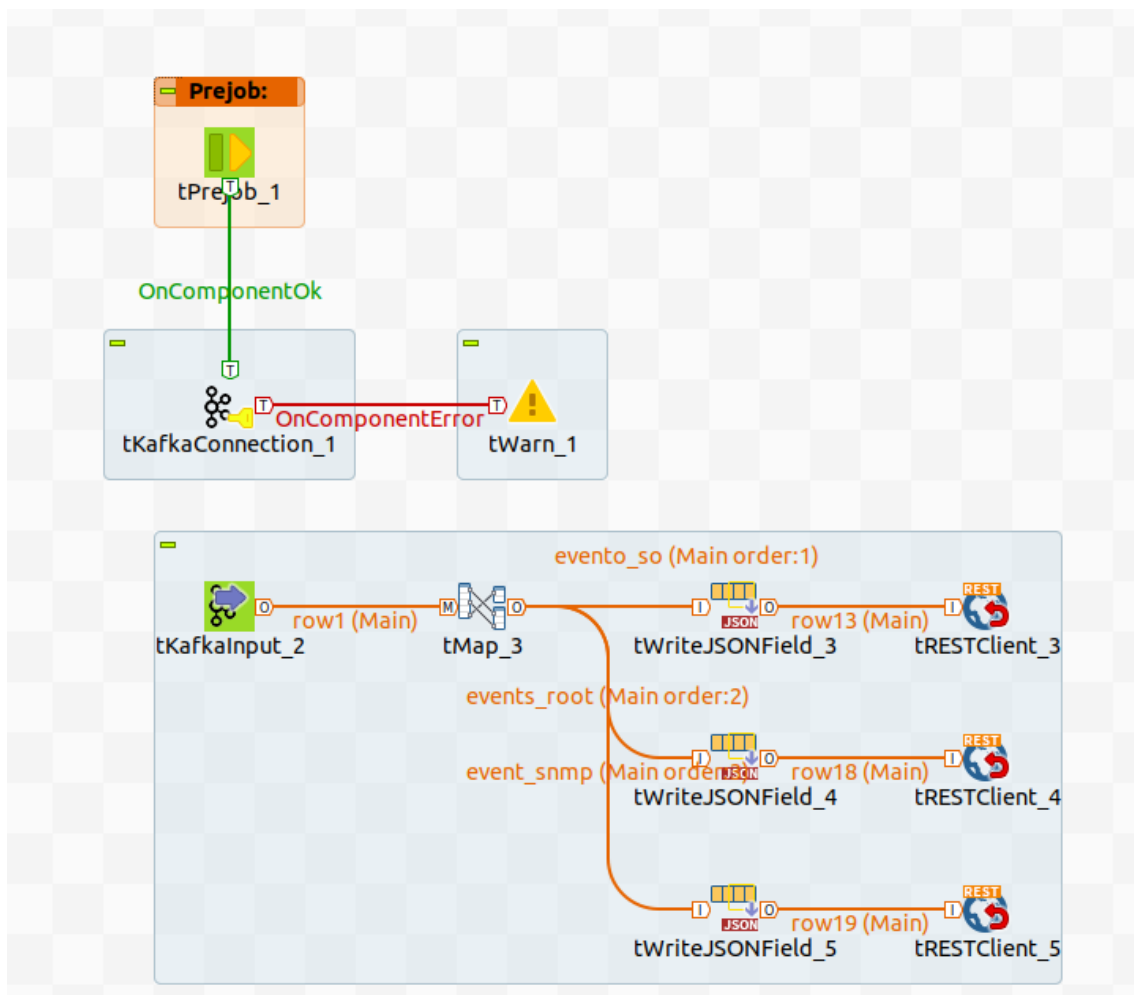


Figure 12- Job de Talend completo

Procedemos ahora a despiezarlo brevemente:

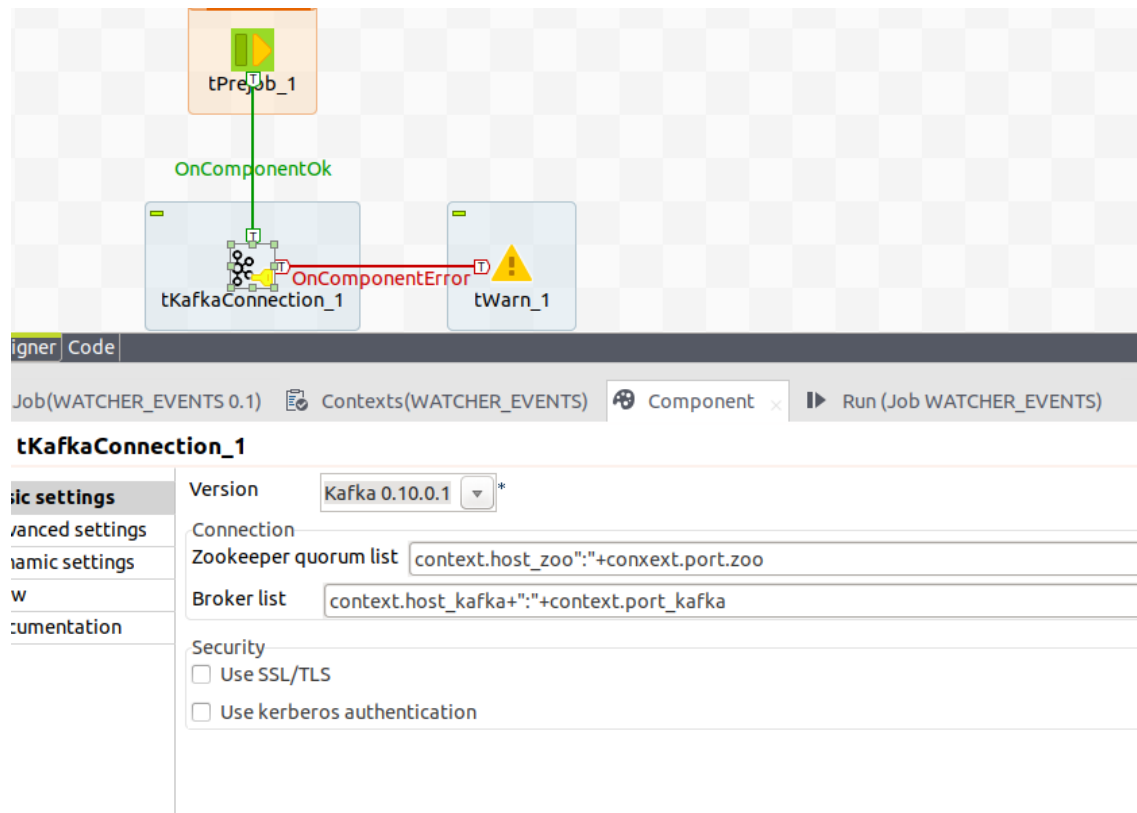


Figure 13 - Componente consumidor de Kafka en Talend

En este primer componente lo único que hacemos es abrir la conexión con el servidor de Kafka.

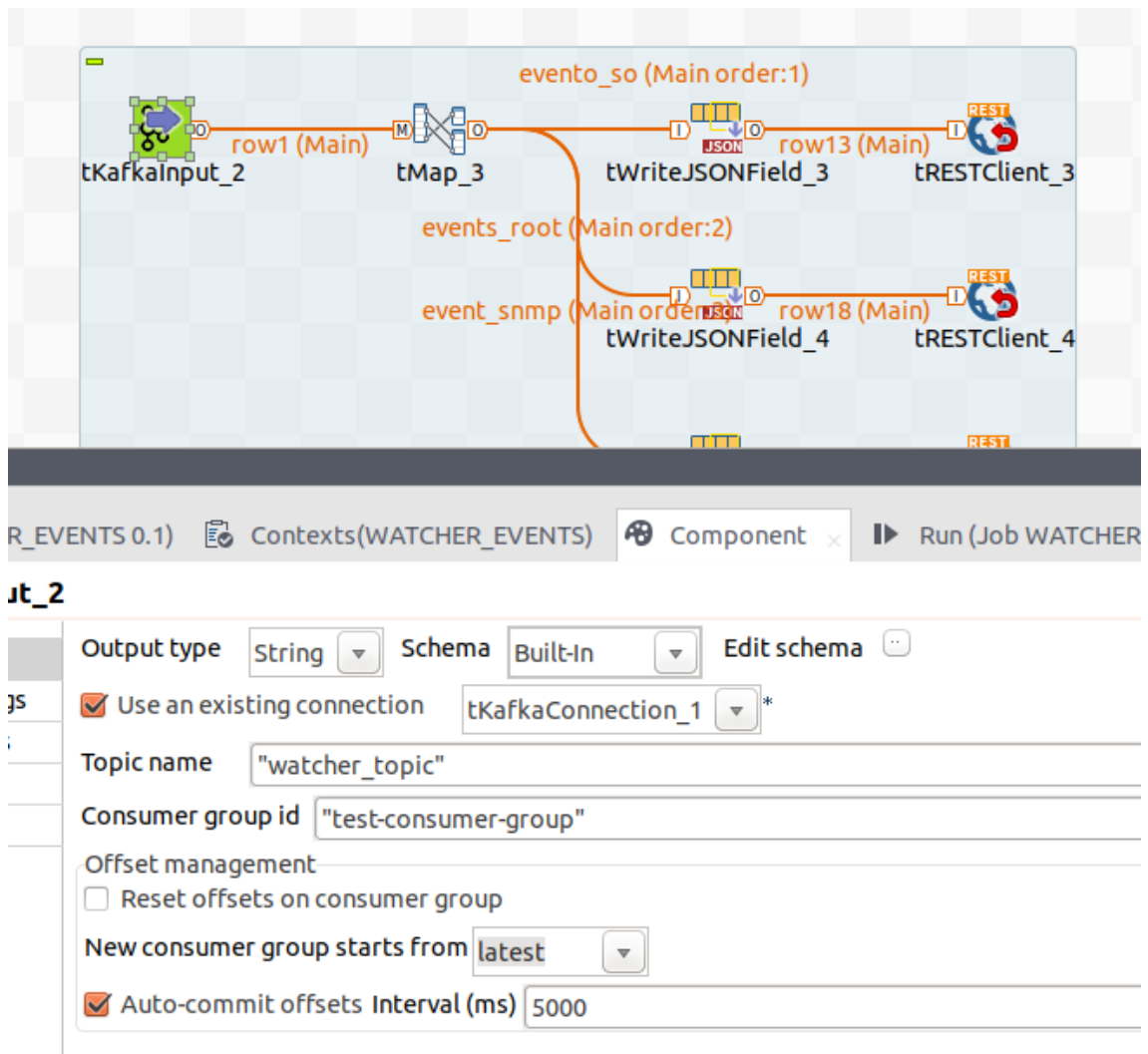


Figure 14 - Componente tKafkaInput

En el siguiente paso abrimos un consumidor de Kafka, que se conecta a partir de la sesión abierta con el servidor Kafka en el primer paso. En este componente ya le señalamos a que topic debe conectar.

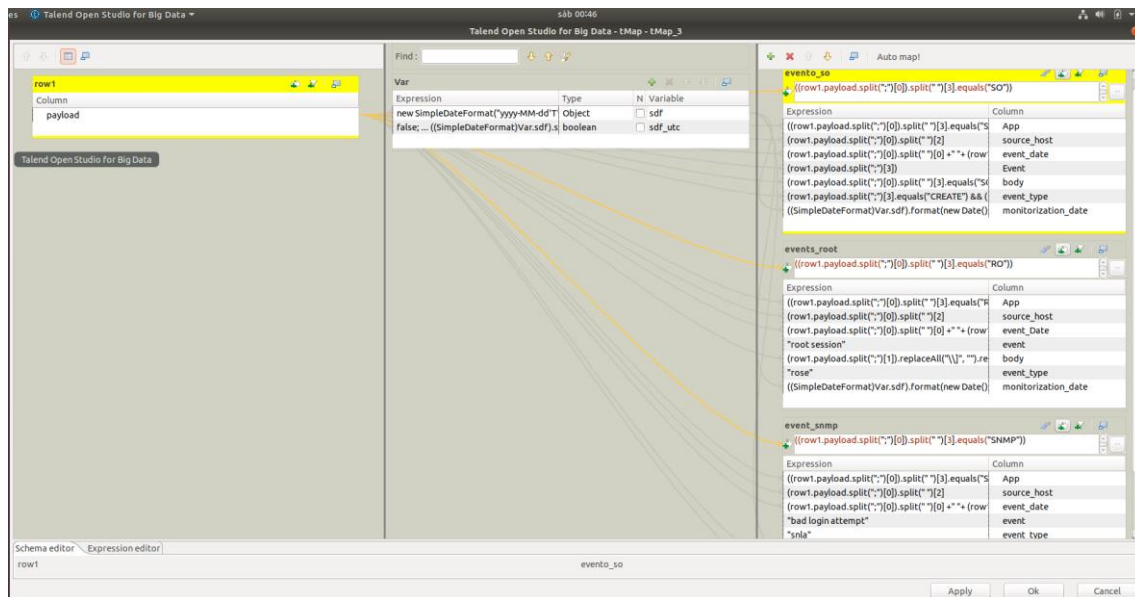


Figure 15 - Parseo de evento de Kafka

Una vez tenemos el evento el siguiente paso es normalizado, ya ello utilizamos el componente tMap, en el que parseamos el string que nos devuelve el consumidor de Kafka. En este caso debemos ponerle especial atención a la normalización de la fecha de monitorización, ya que tiene que estar en UTC y en un formato específico para que más tarde Elastic Search lo cree como tipo Date.

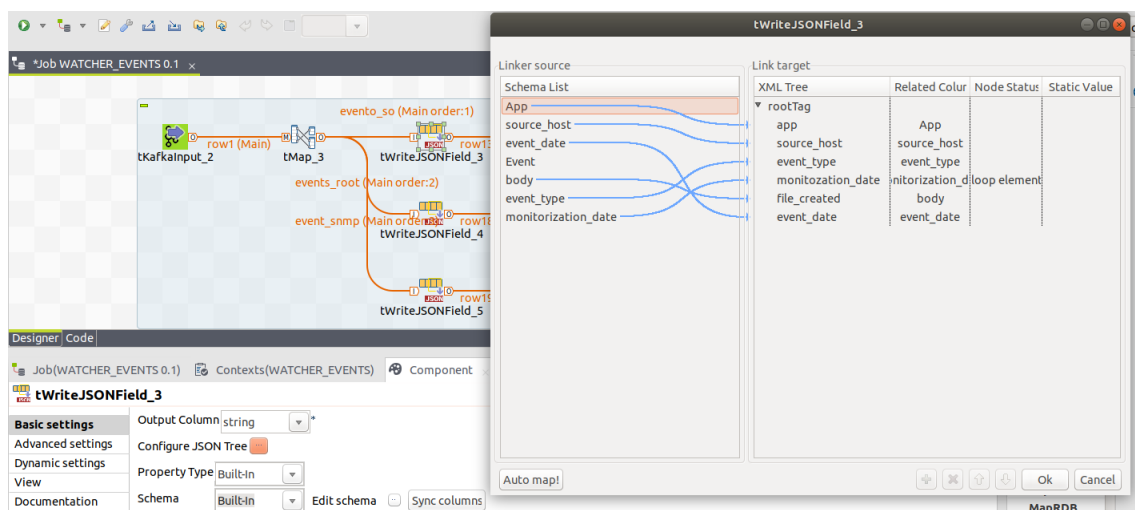


Figure 16 - Convertiendo el evento en JSON

Una vez normalizado el evento procedemos a la transformación del evento a JSON para la posterior inserción en Elastic Search.

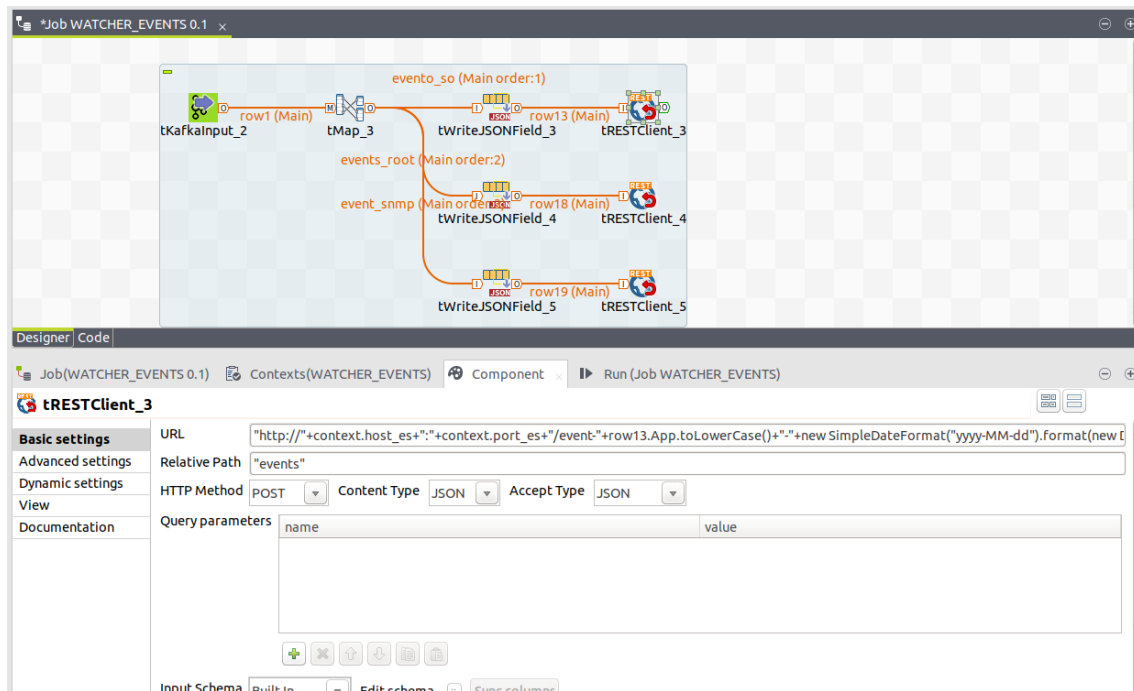


Figure 17 - Llamada a la API de Elastic Search

Finalmente procedemos a la inserción de los eventos a través de la API de Elastic Search con el patrón que mencionamos anteriormente.

Una vez terminado el job lanzamos una prueba del mismo, donde ya obtenemos información de diferentes fuentes y registramos los eventos en Elastic Search:

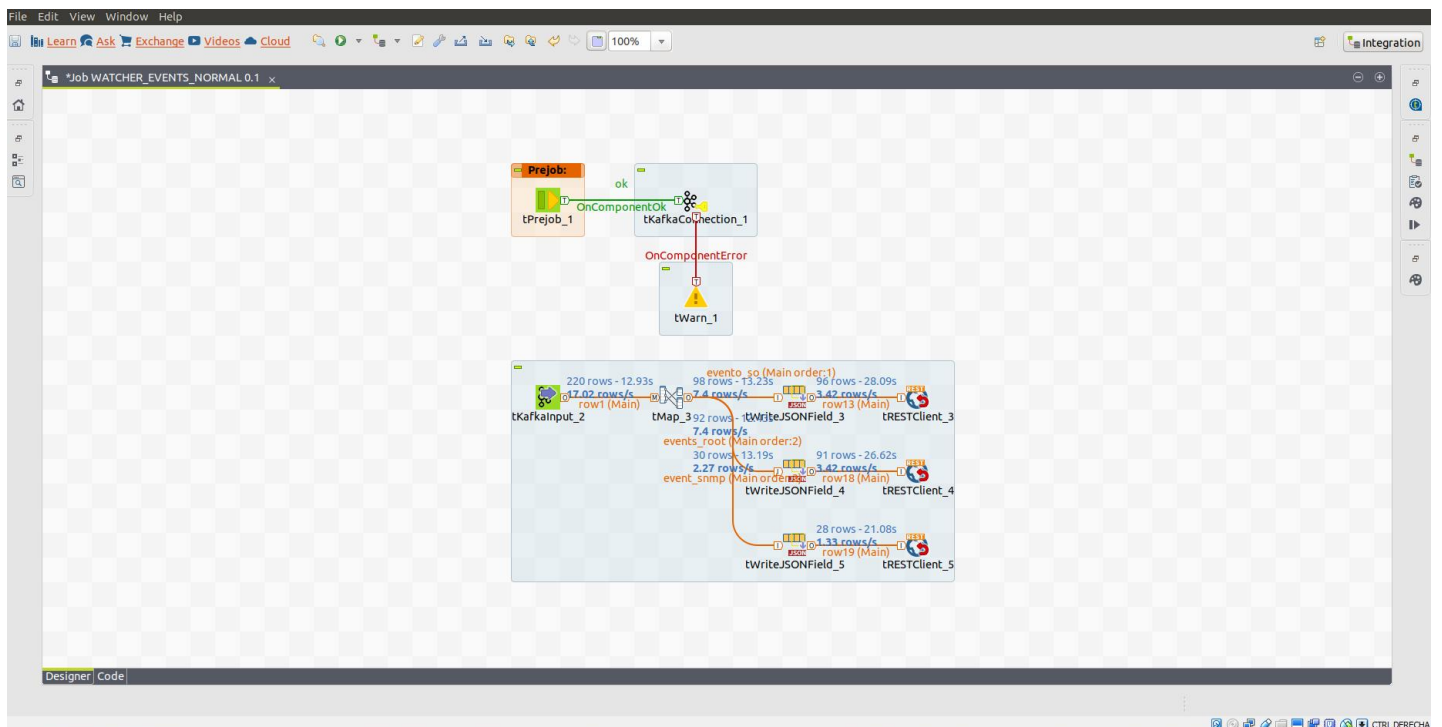


Figure 18 - Job lector de Kafka en ejecución

Con el fin de comprobar que toda la información se ha guardado correctamente en Elastic Search vamos a proceder ahora con la instalación de Kibana para poder hacer consultas de manera visual y sencilla.

Este job será la fuente de datos para el resto de Jobs en los que se crean las alertas. Para más información dirijase al TFM de Francisco Serrano.

4.1-5. Configuración e instalación del visualizador

La instalación de Kibana se realiza a partir de la agregación de la definición de un nuevo repositorio en el directorio de repositorios del sistema `sources.list.d`. Una vez se agrega se puede proceder a la instalación a partir del comando `apt-get install`. Una vez está instalado se lanza el servicio con el comando:

Sudo service kibana start

Una vez que kibana está arrancado vamos a dirigirnos al management de índices para comprobar que los índices se han creado bien y que contienen documents:

Index management							
Update your Elasticsearch indices individually or in bulk							
<input type="checkbox"/> Include system indices							
<input type="text" value="Search"/>							
<input type="checkbox"/> Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Primary storage size
<input type="checkbox"/> event-os-2018-10-26-oscr	● yellow	open	5	1	14	46.1kb	46.1kb
<input type="checkbox"/> event-sn-2018-10-26-snlr	● yellow	open	5	1	28	72kb	72kb
<input type="checkbox"/> event-os-2018-10-26-osac	● yellow	open	5	1	82	164.4kb	164.4kb
<input type="checkbox"/> event-ro-2018-10-26-rose	● yellow	open	5	1	91	148.3kb	148.3kb
Rows per page: 10 ▾							

Figure 19 - Indices creados

Como se puede comprobar ya hay eventos en nuestros índices.

4.2- Visualización de los eventos

Para poder sacar partido a la información que está creando nuestro sistema cada corporación podrá tener la visualización que considere oportuna, pero el sistema vendrá con un dashboard preconstruido y abierto a cambios. Para la creación de este dashboard se ha creado a partir una consulta en que se recogen todas las tablas que empiecen por event con la expresión regular event-*. A partir de esta consulta vamos a construir 4 visualizaciones que lo compondrán:

- Diagrama de barras por días agrupado por los eventos.
- Tabla con todo el detalle de cada evento.
- Gráfico de líneas para seguir la evolución en cuanto a número de eventos de cada tipo de evento.
- Diagrama de tarta agrupado por aplicación.

De este modo el dashboard tiene el siguiente aspecto:

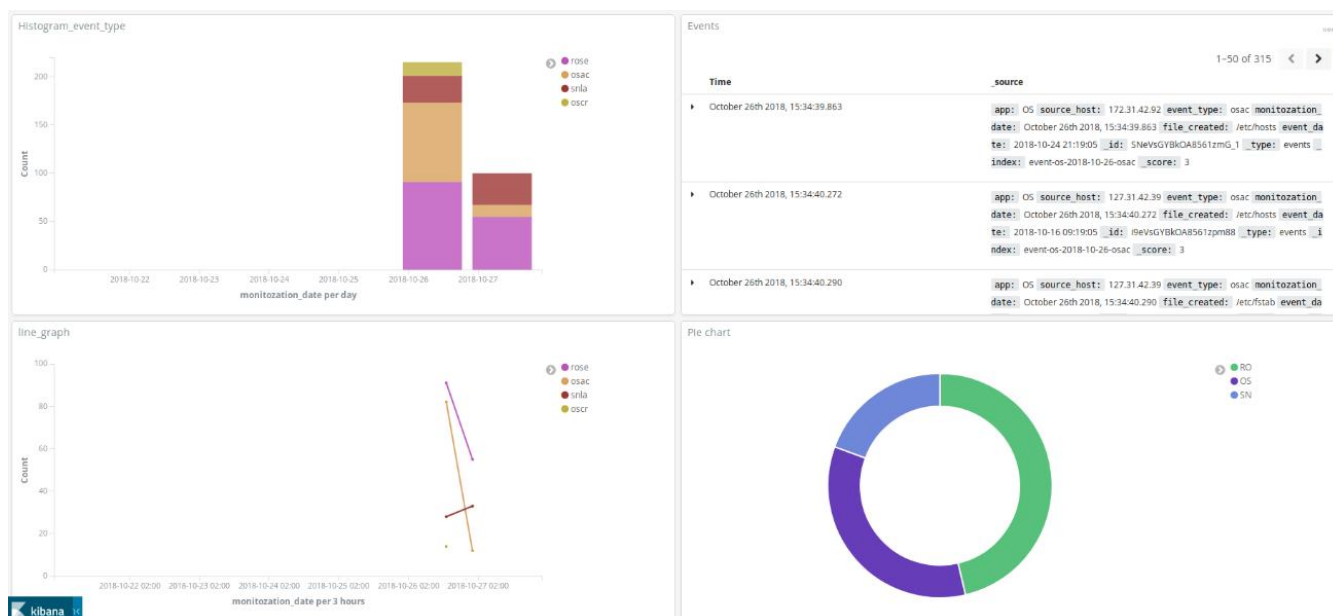


Figure 20 - Dashboard Kibana

En dicho dashboard además hay filtros para modificar el periodo de tiempo que queremos consultar.

A partir de este dashboard se podrá realizar el seguimiento en tiempo real de todos los eventos que tengan origen dentro de cada corporación.

4.3- Creación de reglas

Una vez los eventos están insertados en base de datos se procede a la creación de alertas, para ello es necesario primeramente crear unas reglas sobre los eventos. Estas reglas serán agregaciones o filtros sobre los eventos que hagan al evento o conjunto de eventos que harán que los eventos se transformen en alertas por la naturaleza preocupante de los mismos.

Estas reglas serán centralizadas en la Blockchain, con el fin de que todas las organizaciones puedan utilizar las reglas. A partir de estas reglas se generarán las reglas que vayan a la Blockchain para la propagación entre las organizaciones.

Las reglas que se han implementado en el sistema de pruebas son las siguientes:

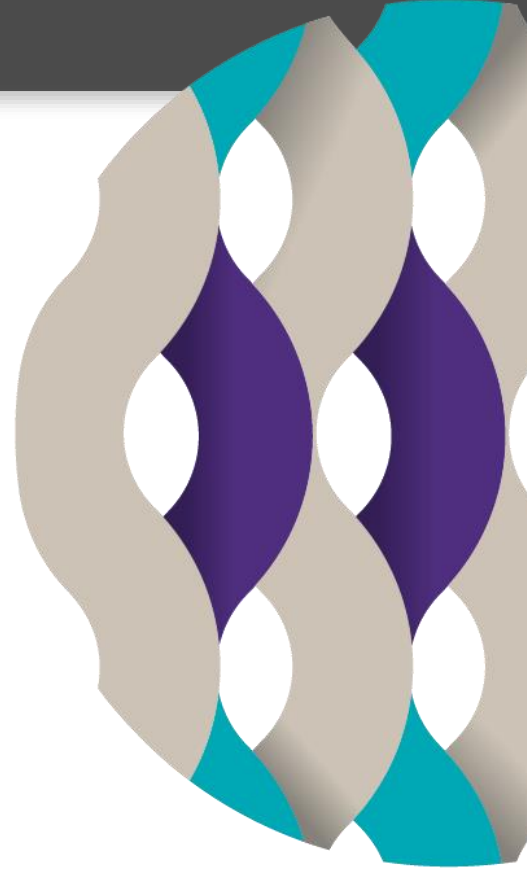
ID	Nombre	URLs	DSL	Org.
accesspasswd	Acceso indebido a /etc/passwd	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":[{"term":{"file_created.keyword":"/etc/passwd"}], "must_not":{"term":{"internal_ruleId":"accesspasswd"} }}}}	Watcher, Umbrella
accessshadow	Acceso indebido a /etc/shadow	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":[{"term":{"file_created.keyword":"/etc/shadow"}], "must_not":{"term":{"internal_ruleId":"accessshadow"} }}}}	Watcher, Umbrella
accessgroups	Acceso indebido a /etc/groups	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":[{"term":{"file_created.keyword":"/etc/groups"}], "must_not":{"term":{"internal_ruleId":"accessgroups"} }}}}	Watcher, Umbrella
accesslogin	Acceso indebido a /etc/login.defs	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":[{"term":{"file_created.keyword":"/etc/login.defs"} }], "must_not":{"term":{"internal_ruleId":"accesslogin"} }}}}	Watcher, Umbrella
accessshells	Acceso indebido a /etc/shells	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":[{"term":{"file_created.keyword":"/etc/shells"}], "must_not":{"term":{"internal_ruleId":"accessshells"} }}}}	Watcher, Umbrella

accesssecurety	Acceso indebido a /etc/securetty	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"term":{"file_created.keyword":"/etc/securetty"}}, "must_not":{"term":{"internal_ruleid":"accesssecurety"}}}}}	Watcher, Umbrella
ransom1	Detectado cifrado Ransomware eecc	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*ecc"}}, "must_not":{"term":{"internal_ruleid":"ransom1"}}}}}	Watcher, Cyberdyne, Umbrella
ransom2	Detectado cifrado Ransomware ezz	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	2{"query":{"bool":{"must":{"regexp":{"file_created":".*ezz"}}, "must_not":{"term":{"internal_ruleid":"ransom2"}}}}}	Watcher, Cyberdyne, Umbrella
ransom3	Detectado cifrado Ransomware exx	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*exx"}}, "must_not":{"term":{"internal_ruleid":"ransom3"}}}}}	Watcher, Cyberdyne, Umbrella
ransom4	Detectado cifrado Ransomware zzz	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*zzz"}}, "must_not":{"term":{"internal_ruleid":"ransom4"}}}}}	Watcher, Cyberdyne, Umbrella
ransom5	Detectado cifrado Ransomware xyz	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*xyz"}}, "must_not":{"term":{"internal_ruleid":"ransom5"}}}}}	Watcher, Cyberdyne, Umbrella
ransom6	Detectado cifrado Ransomware aaa	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*aaa"}}, "must_not":{"term":{"internal_ruleid":"ransom6"}}}}}	Watcher, Cyberdyne, Umbrella
ransom7	Detectado cifrado Ransomware abc	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*abc"}}, "must_not":{"term":{"internal_ruleid":"ransom7"}}}}}	Watcher, Cyberdyne, Umbrella
ransom8	Detectado cifrado Ransomware ccc	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*ccc"}}, "must_not":{"term":{"internal_ruleid":"ransom8"}}}}}	Watcher, Cyberdyne, Umbrella
ransom9	Detectado cifrado Ransomware vvv	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*vvv"}}, "must_not":{"term":{"internal_ruleid":"ransom9"}}}}}	Watcher, Cyberdyne, Umbrella
ransom10	Detectado cifrado Ransomware xxx	event-os- {TODAY}- osac,event-os- {TODAY-1}- osac/events	{"query":{"bool":{"must":{"regexp":{"file_created":".*xxx"}}, "must_not":{"term":{"internal_ruleid":"ransom10"}}}}}	Watcher, Cyberdyne, Umbrella
rootnonworkinghours	Acceso con usuario ROOT fuera de horas de trabajo	event-ro- {TODAY}- rose/events	{"query":{"bool":{"must":{"range":{"_source.event_date":{"gt":{"TODAY}T18:00:00.00"}}, "must_not":{"term":{"internal_ruleid":"rootnonworkinghours"}}}}, {"query":{"bool":{"must":{"range":{"_source.event_date":{"lt":{"TODAY}T8:00:00.00"}}, "must_not":{"term":{"internal_ruleid":"rootnonworkinghours"}}}}}	Watcher, Cyberdyne

snmpincorrect admin	Log-in incorrecto de usuario "admin"	event-snm- {TODAY}- snla/events	{"query":{"bool":{"must":[{"term":{"user_name.keyword":"admin"}],"must_not":{"term":{"internal_ruleId":"snmpincorrectadmin"}}}}}}	Watcher, Cyberdy ne
------------------------	--	---------------------------------------	---	---------------------------

Todo el detalle sobre esta sección y de la visualización de las alertas se puede encontrar en el TFM de Francisco Serrano.

V. Conclusiones



5. Conclusiones

5.1- Mejoras

Este sistema distribuido SIEM se muestra en este documento como una Prueba de Concepto. Para su uso en producción en un consorcio real se proponen las siguientes mejoras y vías de desarrollo futuro:

- Securización de comunicaciones en el sistema: Sería un punto positivo integrar las comunicaciones con Kafka a través de Kerberos para evitar fuga de información y posibles ataques man-in-the-middle, además de implementar comunicaciones con HTTPS, SSL y Certificados Digitales para la comunicación con las siguientes API REST usadas por los distintos componentes del sistema.
- Estandarización de implantación de agente de syslog-ng: Sería conveniente la creación de un espacio compartido, en el que todas las organizaciones pudieran acceder de forma segura para la compartir y estandarizar del desligue del agente de syslog-ng en sus sistemas.
- Sistema para actualizar tipos de evento y aplicaciones a analizar de forma dinámica: si un actor del sistema modifica o añade nuevos tipos de eventos o aplicaciones, estos datos deberían ser actualizados de forma automática en los agentes implicados.
- Pruebas de estrés: realizar grandes cargas de datos en todos los componentes del sistema y en el sistema en general para detectar cuellos de botella y medir tiempos de respuesta bajo condiciones adversas (alto tráfico de red, alto número de ataques en determinado momento).
- Crear una interfaz de usuario amigable para gestionar acciones en la Blockchain (creación y aplicación de reglas, consultas al histórico de transacciones en la Blockchain, Exploración de organizaciones, aplicaciones y tipos de evento).
- Mejorar procesos de ETL:
 - o Basar la entrada de nuevas alertas de otras organizaciones en WebSockets[28].

- Pruebas de estabilidad al lanzar procesos de ETL de forma distribuida en varios servidores a la vez.
- Mejorar sistema de Reglas:
 - Integrar un “Marketplace” de reglas para que las organizaciones del consorcio puedan compartir sus reglas de forma sencilla y efectiva.
 - Cambios en la estructura de las reglas: Actualmente internamente las reglas se guardan como una lista de DSL sin integración entre ellas. Para mejorarlo, las reglas pueden integrar una lista ordenada de conjuntos de DSL, URLs y FileOutputs que se apliquen desde el primero hasta el último, usando la entrada de valores de la siguiente consulta con los datos obtenidos de la consulta anterior.
 - Activar proceso de validación de nuevas reglas: una regla puede no ser válida en el sistema y aun así puede ser añadida al sistema por ser sintácticamente correcta. Es necesario crear nuevos actores en el sistema que se encarguen de validar nuevas reglas y rechazarlas con comentarios si estas no cumplen ciertos requisitos (de eficiencia o de estabilidad) o aceptarlas si pueden ser óptimas para su explotación.
 - Crear un nuevo tipo de actores “data miners” que puedan ser capaces de explotar los datos de los repositorios privados y la Blockchain de un conjunto de organizaciones del consorcio para poder crear reglas avanzadas usando mecanismos de Machine Learning y Data Mining en grandes conjuntos de eventos y alertas de varias organizaciones de forma conjunta (detección de vulnerabilidades, heurísticas, detección de anomalías, etcétera).

5.2- Comparativa con otros sistemas

- Syslog-ng

Para la captación de los logs de todas las máquinas monitorizadas es necesario un sistema que envíe en tiempo real los registros de cada log y que además, en nuestro caso, tenga un conector de Kafka donde se puedan enviar todos los registros.

Se ha elegido syslog-ng por los siguientes motivos:

- o Posee un conector para destinos Kafka, lo cual nos permite enviar en tiempo real cada registro de los logs. A diferencia de su competencia directa el conector de Kafka lleva tiempo implantado y es una versión estable, mientras que el resto de herramientas de la misma naturaleza poseen implantaciones muy recientes y poco estables.
- o Posee la capacidad de realizar filtros de forma sencilla y con un proceso estandarizado, mientras que el proceso de creación de filtros en otras herramientas es mucho más complejo, como se puede ver en las siguientes capturas. [5]

```
filter f_iptables { facility(kern) and message("IN=") and message("OUT=");  
};  
filter f_console { level(warn) and facility(kern) and not  
filter(f_iptables) or level(err) and not  
facility(authpriv); };  
log { source(src); filter(f_console); destination(console); };  
log { source(src); filter(f_console); destination(xconsole); };
```

Figure 21- Filtro en syslog-ng

```
if ( \  
/* kernel up to warning except of firewall */ \  
($syslogfacility-text == 'kern') and \  
($syslogseverity <= 4 /* warning */ ) and not \  
($msg contains 'IN=' and $msg contains 'OUT=') \  
) or ( \  
/* up to errors except of facility authpriv */ \  
($syslogseverity <= 3 /* errors */ ) and not \  
($syslogfacility-text == 'authpriv') \  
) \  
then /dev/tty10  
& | /dev/xconsole
```

Figure 22 - Filtro en rsyslog

- Posee la capacidad de crear plantillas para cada mensaje, es decir, nos permite personalizar el formato con el que syslog-ng enriquecerá el envío del mensaje, por ejemplo: añadir la fecha de envío, un string, la ip de origen, etc.
 - Es posible instalar el agente en cualquier sistema operativo.
 - Además posee la comunidad más amplia de este tipo de herramientas.
- Apache Kafka

Para centralizar los eventos generados se necesita un bus o cola que nos permita recibir y servir los eventos en tiempo real. Para esta tarea existen principalmente tres alternativas Apache Kafka, Flume y colas RabbitMQ.

Se ha elegido Kafka por las siguientes razones [6]:

- El volumen que puede procesar Kafka es mayor que el de sus competidores, ya que puede procesar 100K eventos/segundo, mientras que el resto de las herramientas de centralización de logs procesan unos 20K eventos/segundo como máximo.
- Escala de manera horizontal, por lo que es una escalabilidad mucho más sencilla y viable. Flume y RabbitMQ escalan verticalmente.
- Posee integración de un balanceador de carga, Zookeeper, que puede actuar de amortiguador en momentos de picos de carga.

Por estas razones Kafka ha sido la herramienta seleccionada, ya que es la que mayor recorrido permitirá a nuestro sistema.

- ElasticSearch

Para el repositorio privado de cada organización se necesita un sistema que permita guardar y gestionar gran volumen de datos y permita consultas avanzadas de dichos datos de forma rápida y efectiva.

Se ha elegido ElasticSearch por los siguientes motivos [7]:

- Elasticsearch es capaz de indexar datos que cambian rápidamente casi instantáneamente (en menos de 1 segundo). Por ejemplo, en Uber, Elasticsearch agrega métricas de negocio sobre precios dinámicos y posicionamiento de la oferta, en tiempo real. Es capaz de atender más de 1.000 consultas por segundo en hora punta.
- ElasticSearch es capaz de aumentar su rendimiento cuando la base de datos crece, de modo que la velocidad de búsqueda no se ralentiza.
- Además se puede encontrar que ElasticSearch es el motor de búsqueda con más puntuación en los rankings de relevancia de motores de búsqueda:



17 systems in ranking, October 2018									
Rank			DBMS	Database Model	Score				
Oct 2018	Sep 2018	Oct 2017			Oct 2018	Sep 2018	Oct 2017		
1.	1.	1.	Elasticsearch 	Search engine	142.33	-0.28	+22.09		
2.	2.	↑ 3.	Splunk	Search engine	76.90	+2.87	+12.54		
3.	3.	↓ 2.	Solr	Search engine	61.31	+1.11	-9.82		
4.	4.	4.	MarkLogic	Multi-model 	12.63	+1.10	+0.82		
5.	5.	5.	Sphinx	Search engine	7.55	+0.48	+1.52		
6.	6.	6.	Microsoft Azure Search	Search engine	5.00	+0.21	+1.32		
7.	7.	↑ 8.	Algolia	Search engine	3.69	+0.12	+1.05		

Figure 23 - Comparativa de motores de búsqueda Elastic Search [8]

- Kibana

Para la explotación de la información registrado y generada se necesitaba una herramienta de visualización que permitiese mostrar cada registro en tiempo real.

Se ha elegido Kibana principalmente porque posee una integración nativa con Elastic Search, formando parte del stack ELK, lo cual nos reporta los siguientes beneficios:

- Facilita la comunicación entre las dos herramientas evitando delays innecesarios entre ambos.
- Creación de dashboard a partir de consultas con expresiones regulares, lo cual nos facilita la tarea de creación de visualizaciones a partir de consultas sencillas en una sistema complejo de tablas buscando la eficiencia en el almacenamiento.
- Instalación y configuración sencilla.

- Talend

En este proyecto se usa Talend como herramienta de ETL para sincronizar datos con ElasticSearch, Kafka e Hyperledger. Existen muchas herramientas ETL en el mercado, aunque el siguiente Cuadrante Mágico demuestra que Talend es líder dentro de herramientas dentro del Open Source:

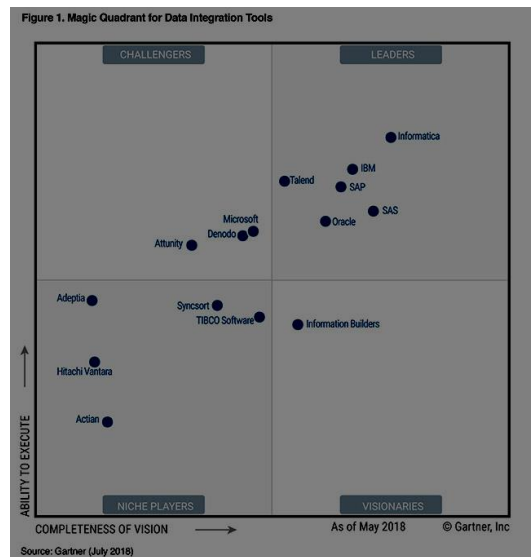


Figure 24 - Cuadrante Mágico de herramientas de integración de datos [9]

Además Talend ayuda a desarrollar de forma rápida y eficaz gracias a su interfaz gráfica y modelo de componentes, incluyendo un amplio número de características que facilitan la integración con un diverso abanico de sistemas.

- Hyperledger Fabric/Composer

A la hora de elegir una Blockchain para este proyecto se han tenido en cuenta las siguientes características:

- o Debe de permitir gran número de transacciones por segundo que deben de ser validadas en el menor tiempo posible.
- o Debe de ser versátil y permitir el desarrollo de procesos y transacciones propias (Smart Contracts).
- o Debe de permitir gestionar y añadir nuevos nodos y participantes de forma dinámica.
- o Debe de ser estable y maduro para ser usado de forma productiva.

- o Debe ser eficiente, no tener algoritmos de consenso costosos (desde el punto de vista de computación).

En esta comparativa podemos encontrar características de 3 sistemas Blockchain actualmente usados:

Characteristics	Ethereum	Hyperledger Fabric	R3 Corda
Programming Language	Solidity	Go, Java	Kotlin
Governance	Distributed among all participants	Linux foundation and organisation in the Chain	R3 and organisations involved.
Smart Contract	Not legally bounded	Not legally bounded	Legally bounded
Consensus Algorithm	PoW. Casper implementation PoS.	PBFT	Notary nodes can run several consensus algorithm
Scalability	Existing scalability issue	Not prevalent	Not prevalent
Privacy	Existing privacy issue	Not prevalent	Not prevalent
Currency	Ether	None Can be made using chaincode	None

Figure 25 - Comparativa entre Ethereum, Hyperledger Fabric y R3 Corda [10]

En la siguiente comparativa se puede estudiar la relación de PBFT (algoritmo de consenso usado por Hyperledger Fabric) en comparación con otros algoritmos de consenso, en el entorno de rendimiento:

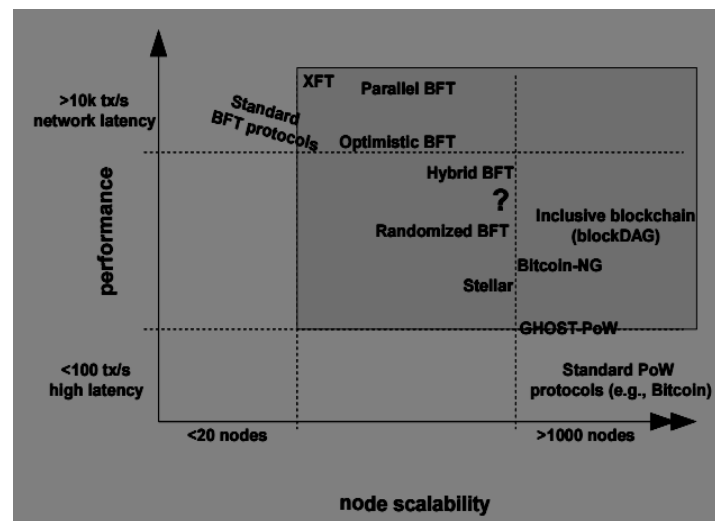


Figure 26- Comparativa entre diversos mecanismos de consenso

Tras estudiar estas comparativas queda patente que Hyperledger Fabric es un candidato óptimo para desarrollar el sistema Blockchain de este proyecto.

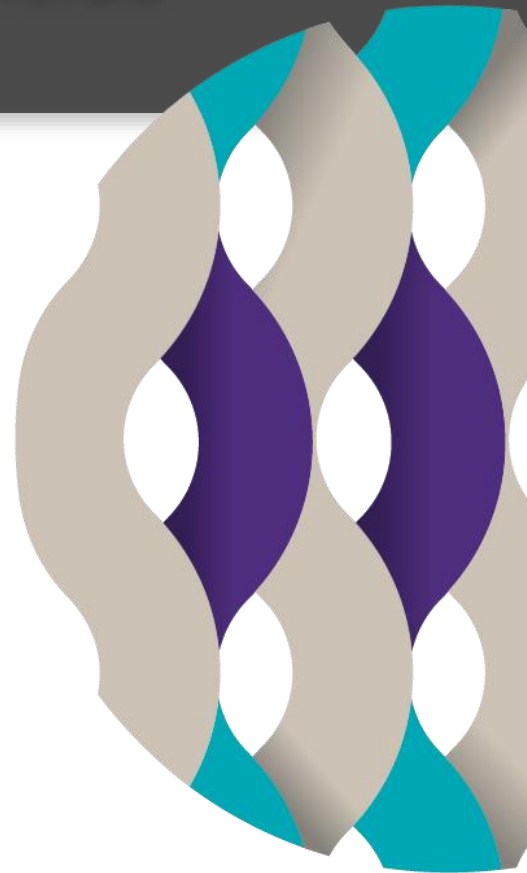
5.3- Conclusiones finales

Tras el desarrollo de esta plataforma podemos encontrar varias reflexiones sobre el producto final:

- Se ha alcanzado el objetivo de la creación de una plataforma de compartición de alertas de ciberseguridad que solvento el gran problema actual de la falta de comunicación entre grandes corporaciones para evitar el daño reputacional. Además añadiendo un plus de monitorización interna de todos los eventos que se definan para cada corporación, lo que le añade valor al producto final. Teniendo en cuenta que actualmente la tecnología blockchain está en pleno auge pero que hay muy pocos proyectos productivos este sería un buen ejemplo si se llega a productivizar en algún momento, ya que es un gran ejemplo del gran potencial de blockchain en lo referente a la anonimizacion de eventos.
- El resultado del desarrollo es una plataforma completamente funcional que con un tiempo de desarrollo puede ser un producto final perfectamente implantable en grandes corporaciones.
- Es una herramienta exportable a otros sectores, como el fraude en seguros y banca, fraude bancario entre países, etc.

Como conclusión final creo que ha sido un proyecto muy completo en el que se han mezclado tecnologías de las diferentes ramas de la computación actual (Big Data, Blockchain y ciberseguridad) que lo hacen un proyecto complejo a la par que útil para cualquier corporación y que serviría como punto de expansión para otros sectores empresariales.

VI. Referencias



6. Referencias

- [1] Dr.Dobb's Journal (Octubre 2018). SIEM: A Market Snapshot. Recuperado de <http://www.drdobbs.com/siem-a-market-snapshot/197002909>
- [2] Urfeena Elahi (Julio 2018). Elastic Stack — A Brief Introduction- Recuperado de: <https://hackernoon.com/elastic-stack-a-brief-introduction-794bc7ff7d4f>
- [3] Margaret Rouse (Agosto 2018). Gestión de eventos e información de seguridad (SIEM). Recuperado de: <https://www.helpsystems.com/es/blog/que-es-un-siem>
- [4] Comunidad de Wikipedia. Recuperado de: https://es.wikipedia.org/wiki/Cadena_de_bloques
- [5] Anónimo - syslog-ng vs. rsyslog comparison. Recuperado de: <https://www.syslog-ng.com/products/open-source-log-management/syslog-ng-rsyslog-comparison.aspx>
- [6] Diego Calvo (Julio 2018) - Comparativa Kafka, Flume y RabbitMQ. Recuperado de: <http://www.diegocalvo.es/comparativa-kafka-flume-rabbitmq/>
- [7] Greenice (Febrero 2018). Best Open Source Search Platform Comparison. Recuperado de: <https://greenice.net/elasticsearch-vs-solr-vs-sphinx-best-open-source-search-platform-comparison/>
- [8] DB-Engines (Octubre 2018). DB-Engines Ranking of Search Engines. Recuperado de: <https://db-engines.com/en/ranking/search+engine>
- [9] Gartner (Julio 2018). Magic Quadrant for Data Integration Tools. Recuperado de: <https://www.gartner.com/doc/3883264/magic-quadrant-data-integration-tools>
- [10] Shourya Shirsha Nandi (Marzo 2018). Technical difference between Ethereum, Hyperledger fabric and R3 Corda. Recuperado de: <https://medium.com/@micobo/technical-difference-between-ethereum-hyperledger-fabric-and-r3-corda-5a58d0a6e347>