# Sistema SIEM usando tecnologías Blockchain y Open Source: “Watcher”

Francisco Serrano Carmona

Máster en Ciberseguridad IMF Business School – Universidad Camilo José Cela

28/10/2018

Instructor: Manuel Carpio Cámara

## 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[2], 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[3]: ElasticSearch, Logtash, Kibana). Por último se comentarán las futuras mejoras y características pendientes de implementar en el sistema presentado.

Sistema SIEM usando tecnologías Blockchain y Open Source: “Watcher”

# Índice

# Tablas y Figuras

# Requisitos Funcionales y Técnicos

El desarrollo de este proyecto parte de los siguientes requisitos funcionales:

- Este sistema está orientado a consorcios o conjuntos de organizaciones que deseen compartir una parte de sus eventos relacionados con la ciberseguridad de la organización para que se creen sinergias entre las distintas organizaciones tanto en información disponible como atención y ayuda entre los distintos agentes del sistema. Podría definirse como un SOC[4]/CERT[5] descentralizado en el que varias organizaciones pueden colaborar sin necesidad de un ente central de coordinación del conjunto.
- El sistema debe ser capaz de capturar eventos de diversas fuentes de datos (la mayor parte, desde los propios sistemas de las organizaciones), filtrarlos y guardarlos en un repositorio privado de la organización.
- A partir de este repositorio privado, la organización debe ser capaz de generar alertas que podrán ser compartidas con otras organizaciones del consorcio.
- Una organización también puede ser capaz de utilizar reglas generadas por otras organizaciones y así poder gestionar el intercambio de información de forma óptima. Estas reglas generan las alertas a partir del repositorio privado de la organización.
- Todas las organizaciones pertenecientes al consorcio deben compartir una forma de presentar alertas estándar, con formatos conocidos y definidos por todos los miembros del consorcio.

- La información de los datos alertados y compartidos con otras organizaciones deben ser trazables en todo momento e inmutables (evitando así posibles intentos de ocultar rastros por parte de implicados en eventos de ciberseguridad).
- Además el sistema debe ofrecer a cada organización las funcionalidades de cualquier otro SIEM del mercado:
  - Gestión de Logs y Eventos desde equipos y sistemas de la organización.
  - Búsquedas y reportes de dichos eventos.
  - Generación de informes y cuadros de mando.
  - Gestión de alertas e incidentes.

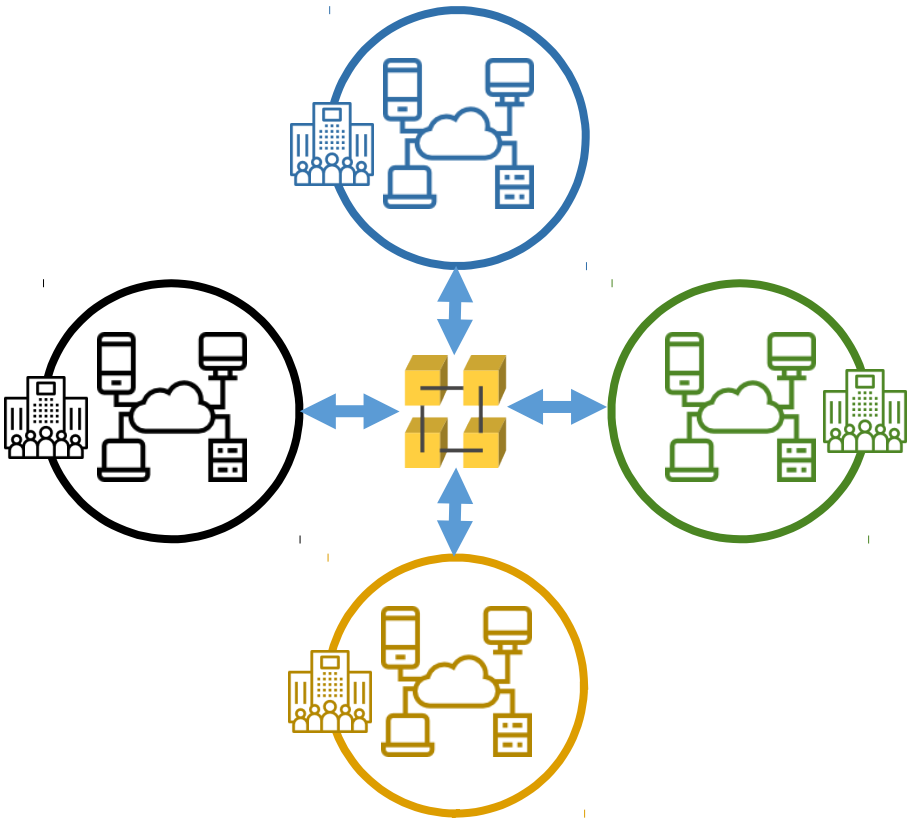


*Figura 1: Intercambio de información entre varias organizaciones de forma descentralizada*

# Diseño del Sistema

Para cumplir los requisitos contemplados en el apartado anterior se ha optado por utilizar las siguientes tecnologías:

- Syslog-ng[6] como agente de los sistemas implicados. Encargado de recolectar información (en forma de logs o eventos de sistema) y enviarla uno o varios servidores para su posterior procesamiento.
- Kafka[7] es la herramienta encargada de conectar con los agentes de Syslog (gracias a su servicio de publicadores y subscriptores de colas), hacer de *buffer* y enviar esta información a otro sistema de la forma más eficiente posible.
- Talend[8] como herramienta de normalización, transformación, enriquecimiento, *enrutamiento* y limpiado de datos. Talend es una herramienta destinada a desarrollar procesos ETL[9] (Extraer, Transformar y Cargar) basados en Java[10]. Se han desarrollado procesos usando Talend en este proyecto para subscribirse a la cola de eventos de Kafka y comunicarse con otros sistemas de almacenamiento del SIEM.
- Elasticseach[11] como repositorio privado de información del SIEM para una organización. Elasticsearch es un motor de almacenamiento y búsqueda de información basado en tecnologías Lucene y NoSQL. ElasticSearch dispone de Kibana[12] para la consulta, visualización y manipulación de los datos.
- Hyperledger[13] (Fabric/Composer) como repositorio compartido con otras organizaciones del consorcio. Hyperledger Fabric es un sistema Blockchain, es decir: un sistema transaccional de registro de datos inmutable y secularizado a través de técnicas criptográficas y mecanismos de consenso entre todos los actores del sistema.

# 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:

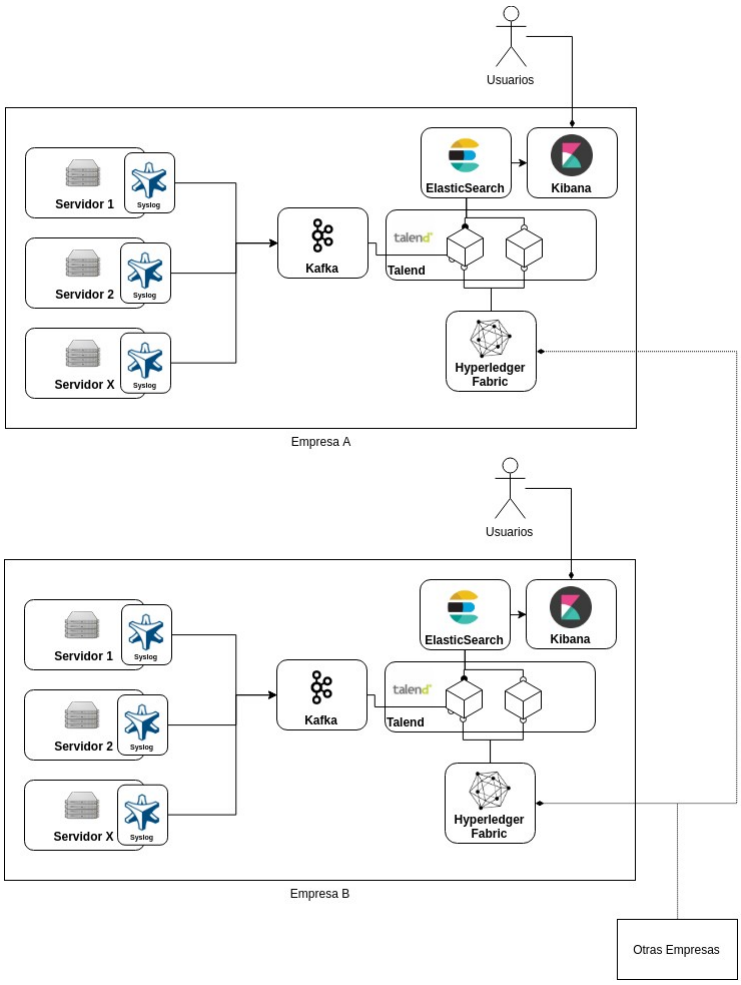



*Figura 2: Arquitectura del sistema a alto nivel*

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).

## Diseño de recolección de datos en Syslog

Como se ha comentado anteriormente, syslog-ng es un sistema encargado de recolectar información (en forma de logs o eventos de sistema) y enviarla uno o varios servidores para su posterior procesamiento.

El diseño y configuración de esta tecnología en el sistema se contempla en el documento de Sergio Muñoz Gamarra.

## Diseño de colas de datos en Kafka

Como se ha comentado anteriormente, Kafka es un servicio en envío de mensajes distribuido usado para leer los eventos generados por los diversos agentes de Syslog en la organización.

El diseño y configuración de esta tecnología en el sistema se contempla en el documento de Sergio Muñoz Gamarra.

## Diseño de motor de eventos en Talend

Los procesos desarrollados en Talend son usados desde dos perspectivas en este proyecto: como motor de eventos (gestionando la entrada de eventos para que puedan ser consultados desde el repositorio privado en ElasticSearch) y como motor de reglas (es decir, como gestor de operaciones con el repositorio privado para analizar qué alertas pueden compartirse en la Blockchain del consorcio y bajo qué circunstancias).

El diseño y configuración de esta sección se contempla en el documento de Sergio Muñoz Gamarra.

# Diseño de motor de reglas en Talend

Desde la perspectiva de motor de reglas, se han desarrollado en Talend los siguientes procesos:

- WATCHER_FILL_COMMON

  Este proceso actualiza los datos auxiliares guardados en Hyperledger Fabric hacia ElasticSearch. Consulta el estado en la Blockchain de los siguientes objetos: Aplicaciones, Tipos de Evento, Organizaciones y Reglas (asociadas a la Organización que consulta la información)[1].

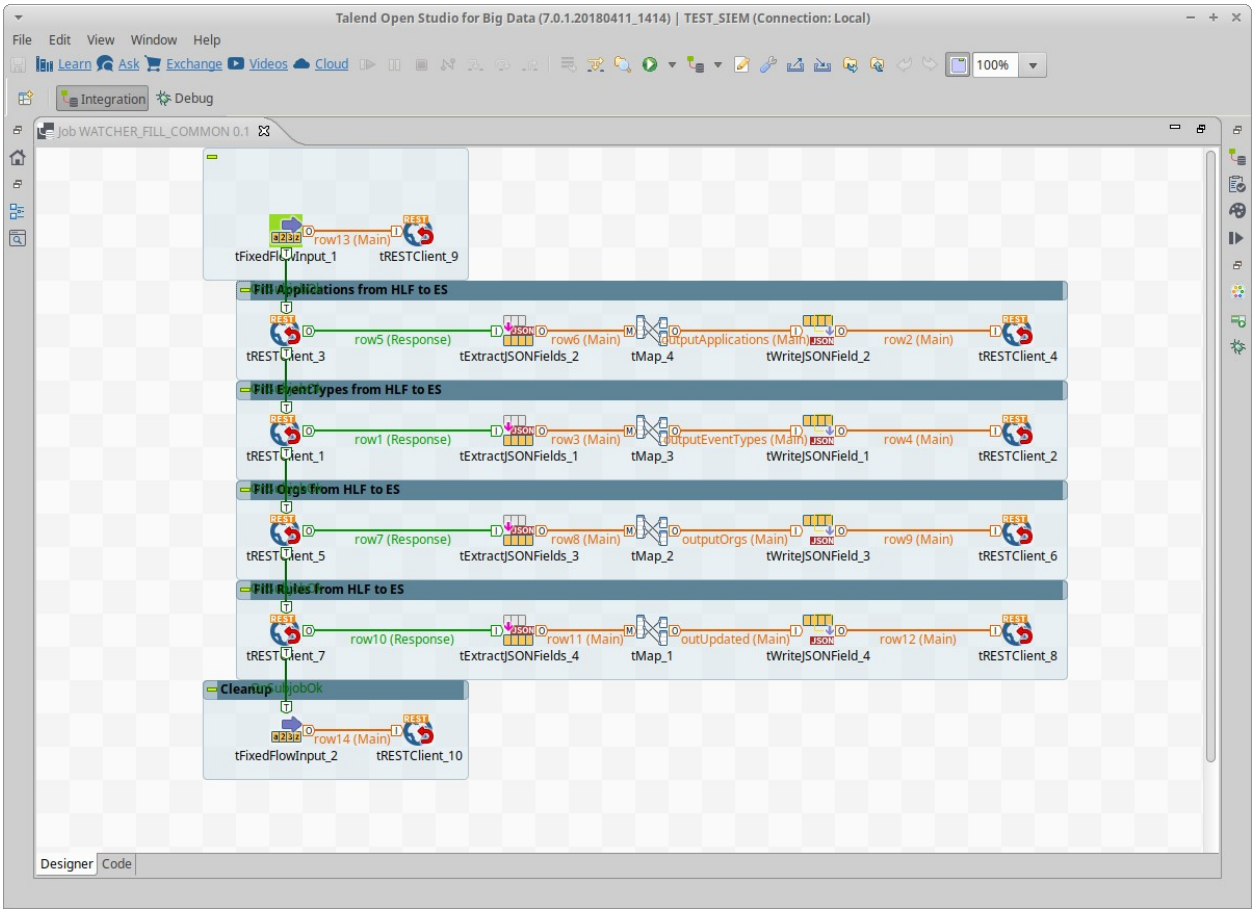



*Figura 3: Diseño de proceso de escritura de objetos comunes*

1 Para una información más extendida sobre estos datos, por favor consulte la sección “Diseño de datos en Blockchain con Hyperledger”

Siguiendo la ejecución del proceso se realizan las siguientes acciones:

- Se actualizan los objetos Aplicaciones, Tipos de Evento, Organizaciones y Reglas en ElasticSearch, añadiendo un campo de “IsUpdated = N”, el cual se usa para indicar que por defecto el registro no se utiliza y puede ser eliminado.
- Por cada tipo de objeto se hace una llamada a la API REST[14] configurada como *endpoint* con el nodo de Hyperledger Fabric/Composer configurado para la organización, la cual devuelve la lista (en *json[15]*) de todos los objetos que están activos actualmente en la Blockchain.
- Cada lista de objetos es actualizada en los índices configurados en ElasticSearch para cada objeto[2], cambiando el campo *IsUpdated* de cada objeto insertado/actualizado a “IsUpdated = Y”.
- Por último, se eliminan de ElasticSearch todos los objetos que no han sido actualizados, finalizando la sincronización de estos objetos entre ElasticSearch y la Blockchain.

Este proceso de actualización es necesario ya que ofrece la posibilidad al sistema de visualizar y consultar los datos existentes en la Blockchain desde un único lugar: la copia de esta información en ElasticSearch. Esto también ofrece más posibilidades a Kibana para hacer reporting.

2 Para una información más extendida sobre estos datos, por favor consulte la sección “Diseño de salvado de datos en ElasticSearch”

- WATCHER_GET_ALERTS

  Este proceso consulta nuevas alertas generadas (por la organización que consulta los datos o por otras organizaciones) en la Blockchain y guarda estas nuevas alertas en el repositorio privado de la organización (ElasticSearch).

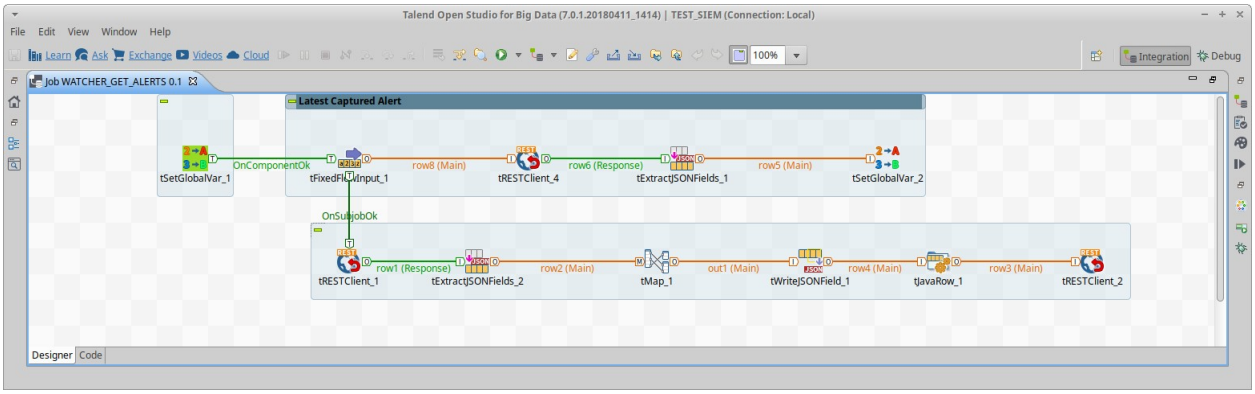



*Figura 4: Diseño de proceso de captura de alertas desde la Blockchain hacia el repositorio privado*

Siguiendo la ejecución del proceso se realizan las siguientes acciones:

- Se busca el último *timestamp* guardado en ElasticSearch
- Se usa este *timestamp* para consultar en Hyperledger Fabric las alertas con una fecha y hora de generación mayores que dicho *timestamp*.
- Una vez obtenidas las nuevas alertas desde la Blockchain, se guardan en ElasticSearch[3].

- WATCHER_RULES

Este proceso utiliza las reglas guardadas en el repositorio privado para generar nuevas alertas basándose en eventos y alertas registradas en ElasticSearch. Es un proceso multi-hilo[16] que garantiza alto rendimiento en búsqueda y generación de nuevas alertas en el sistema.

3 Para una información más extendida sobre los índices usados en ElasticSearch para guardar alertas, por favor consulte la sección “Diseño de salvado de datos en ElasticSearch”

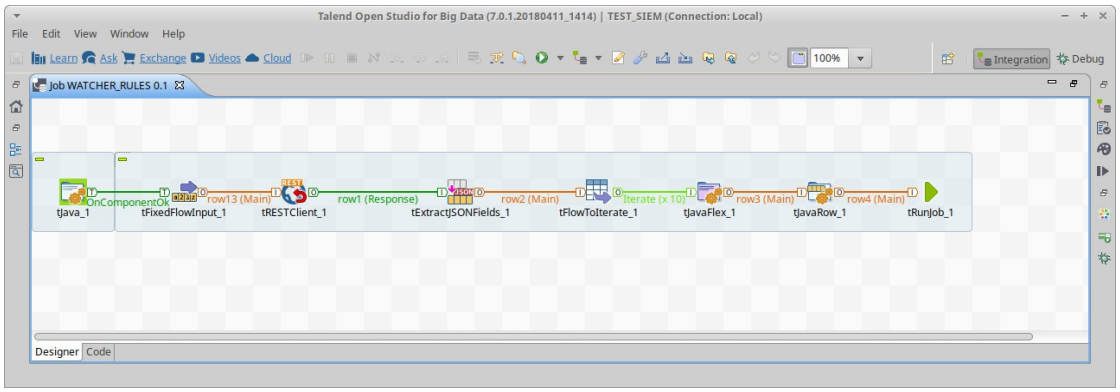



*Figura 5: Diseño de proceso de generación de nuevas alertas basándose en datos del repositorio privado*

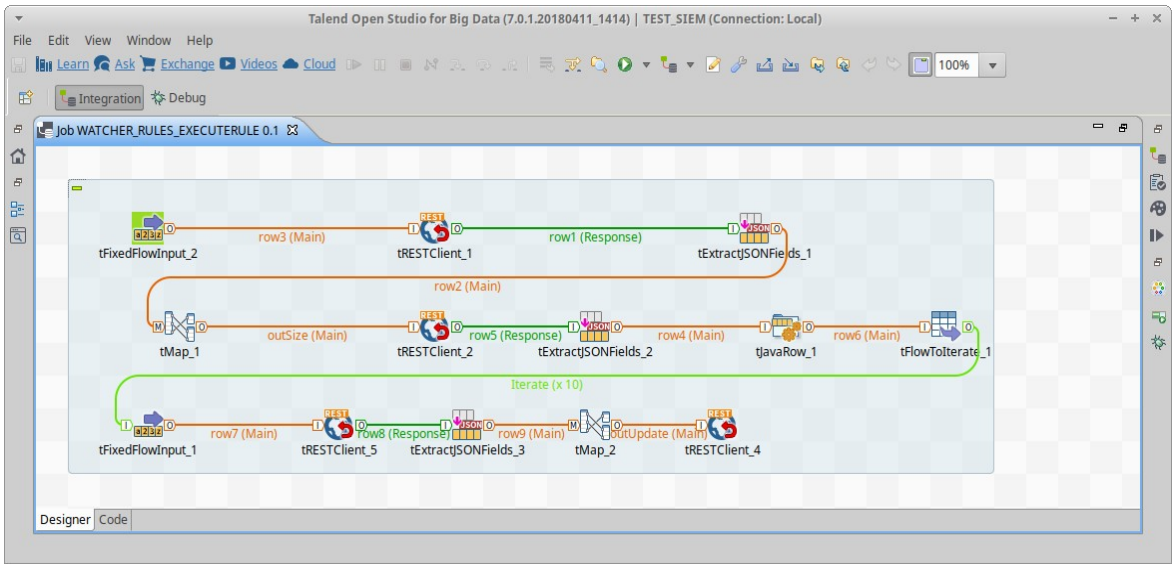



*Figura 6: Diseño del subproceso que se ejecuta para cada regla implicado en la generación de nuevas alertas basándose en datos del repositorio privado*

Siguiendo la ejecución de estos procesos se realizan las siguientes acciones:

- Se consultan todas las reglas que aplican a la organización conectada guardadas en ElasticSearch.
- Por cada regla se ejecuta un subproceso que consulta la información del repositorio privado basándose en la información de Consulta DSL[4] guardada en la regla[5].

4 Consulta DSL (Query DSL) es el mecanismo para consultar información en ElasticSearch[17] , similar a SQL en bases de datos relacionales.

5 A estas consultas se le aplican transformaciones para que se adapten al contexto de ejecución de la consulta por ejemplo: se reemplaza la palabra clave {TODAY} por la fecha actual (GMT), {TODAY-X} con la fecha actual menos X días, {NOW} con la fecha y hora exactas actuales (GMT), {NOW-X} con la fecha y horas menos X milisegundos. Esto nos permite hacer consultas basadas en lapsos de tiempo de forma óptima.

- Una vez obtenido el resultado de la consulta se generan nuevas alertas (si existieran) basándose en los datos recibidos de ElasticSearch. Estas nuevas alertas son guardadas en la Blockchain para su consulta por otras organizaciones del consorcio.

• WATCHER_ORCHESTATOR

Este es un proceso encargado de orquestar otros procesos ETL[9] generados en Talend.

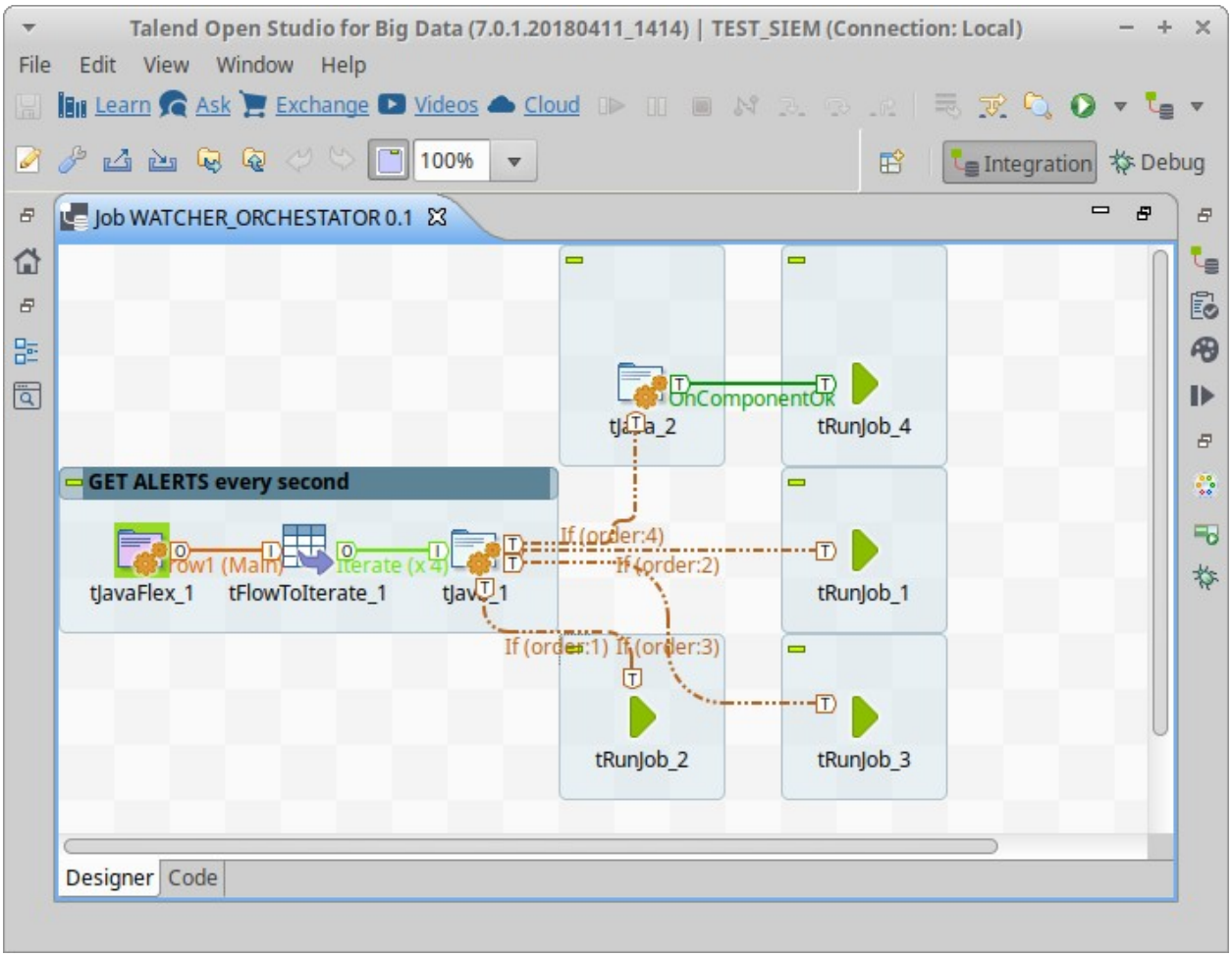



*Figura 7: Diseño de proceso encargado de orquestar otros procesos ETL*

Este proceso gestiona las siguientes tareas, en un entorno multi-hilo [16] :

- Al inicio del proceso se lanza la tarea de generación de nuevos eventos en el sistema[6], que se mantiene en funcionamiento de forma constante.

6 Para más información sobre esta tarea, por favor consulte la sección “Diseño de motor de eventos en Talend”.

- La tarea de obtención de nuevas alertas desde la Blockchain “WATCHER_GET_ALERTS” es ejecutada cada segundo (para mantener una actualización constante de las nuevas alertas que se lanzan en todo el consorcio de organizaciones).
- La tarea de generación de nuevas alertas usando reglas “WATCHER_RULES” es ejecutada cada 10 segundos (buscando un equilibro para generar nuevas alertas de forma constante sin sobrecargar el sistema).
- La tarea de actualización de datos auxiliares y reglas “WATCHER_FILL_COMMON” es ejecutada cada 60 segundos. Los datos que gestiona esta tarea no son actualizados de forma muy frecuente, por lo que no es necesario ejecutar la tarea en un menor período de tiempo.

La temporalización de las tareas son configuraciones por defecto que pueden adaptarse según las capacidades y compromisos de cada organización. Así, si una organización necesita generar alertas cada menos de 10 segundos, este parámetro puede reducirse a un menor tiempo (hasta 1 segundo), aunque esto repercutirá en una mayor carga de trabajo para el sistema en su organización (esto no tiene impacto para otras organizaciones del consorcio).

# Diseño de datos en ElasticSearch

Como se ha comentado en secciones anteriores ElasticSearch es usado como repositorio privado de cada organización en el consorcio de este SIEM distribuido. Este repositorio es usado para guardar la siguiente información:

- Información relativa a Eventos: Acciones simples (atómicas), que pueden suponer (o no) una amenaza. Con la agregación, correlación y filtrado de los Eventos se generan alertas en el sistema.
    - El diseño y configuración de los Eventos en ElasticSearch se contempla en el documento de Sergio Muñoz Gamarra.
- Información relativa a Reglas: Cada regla es un conjunto de consultas a ElasticSearch para generar Alertas. Cada Organización guarda en su repositorio privado las reglas que dicha organización aplica (cada regla puede ser aplicada por una o varias organizaciones del consorcio). Cada regla se compone de la siguiente información:
    - ID: El identificador único de la regla. Debe ser una cadena de letras (en minúscula) y/ o números.
    - Name: El nombre de la regla, el cual da una descripción significativa de la regla.
    - TypesInvoled: Tipos de Evento involucrados en la alerta (útil para realizar agregaciones según tipos de evento).
    - DSLs[7]: Lista de Consultas DSL a aplicar en ElasticSearch para generar alertas en base a esta regla.

7 Consulta DSL (Query DSL) es el mecanismo para consultar información en ElasticSearch[17] , similar a SQL en bases de datos relacionales.

- URLs: Lista de URLs que hacen referencia a los índices[8] que se consultan para generar alertas en base a esta regla.
- FieldsOutputPath: Lista de campos que son añadidos a la alerta (en formato de JsonPath[9]), obtenidos de los resultados de la ejecución de los DSL.
- Hits: Ruta JsonPath donde encontrar los resultados de una consulta DSL (dependiendo de la consulta realizada, estos datos pueden estar diferentes secciones de la respuesta que ElasticSearch genera).

```
"id": "ransom13",
"name": "Ransomware encrypted",
"typesInvolved": [
  "resource:watcher.model.EventType#oscr"
],
"DSLs": [
  """{"query":{"bool":{"must":[{"regexp":{"file_created":".*encrypted"}}],"must_not":{"term" :{"internal_ruleId":"ransom13"}}}}}"""
],
"URLs": [
  "event-os-{TODAY}-oscr,event-os-{TODAY-1}-oscr/events"
],
"fieldsOutputPath": [
  "_source.source_host",
  "_source.monitorization_date",
  "_source.file_created",
  "_source.event_date"
],
"hits": "$.hits.hits[*]",
```

*Figura 8: Estructura de Regla en ElasticSearch*

Todas las reglas relativas a la organización se guardan en un mismo índice del repositorio privado de la organización: *rules/myrules* . Estas reglas son actualizadas mediante consultas a la Blockchain usando el proceso de Talend “WATCHER_FILL_COMMON”.

- Información relativa a Alertas: Las alertas son el producto de la aplicación de reglas a los eventos (u otras alertas) del repositorio privado de cada organización. Para un mayor enriquecimiento de la información en los repositorios privados de cada organización,

8 Índice (o Index) es el elemento básico de consulta de información en ElasticSearch (como una “Tabla en SQL)[18].

9 JSONPath es una forma estandarizada para consultar elementos de un objeto JSON. JSONPath utiliza expresiones de ruta para desplazarse por elementos, elementos anidados y matrices en un documento JSON[19].

todas las organizaciones guardan una réplica de todas las alertas generadas en el consorcio. Esto permite consultas y análisis más exhaustivos de cada organización usando las herramientas disponibles en ElasticSearch/Kibana. Cada alerta se compone de la siguiente información:

- TransactionID: El ID único de la alerta. Este ID es idéntico al ID de la transacción generada en la Blockchain para crear la alerta, manteniendo una sincronía entre la Blockchain y el repositorio privado.
- Timestamp: La fecha y hora exactas (ajustado a micro segundos) de generación de la alerta, en zona horaria GMT.
- Organization: El ID de la organización que ha generado la alerta.
- Field0...X: conjunto de campos (Field0, Field1…) que guardan la información devuelta la regla asociada a la alerta (estos campos se generan usando el campo FieldsOutputPath de la regla).
- Rule: ID de la regla aplicada para generar esta alerta.


```
"transactionId": "f6966951a881cdc8abd6cf66db570f40357d5b0f7edc1b15d066bad9ce407d6c",
"timestamp": "2018-10-27T17:56:51.566Z",
"organization": "org2",
"field0": "127.31.42.24",
"field1": "2018-10-27T17:56:45.583Z",
"field2": "/etc/login.defs",
"field3": "2018-10-16 10:19:56",
"rule": "accesslogin2"
```


*Figura 9: Estructura de Alerta en ElasticSearch*

Todas las alertas se guardan en índices del repositorio privado de la organización, siguiendo el siguiente patrón: alert-{org}-YYYY-MM-DD-{rule}*/alerts* , siendo:

- {org} el ID de la organización que genera la alerta.
- YYYY-MM-DD la fecha en la que se genera la alerta, en formato Año-Mes-Día

- {rule} el ID de la regla aplicada para generar esta alerta.

Esta forma de guardar los índices los posibilita hacer consultas eficientes sobre los datos, por ejemplo:

- Para consultar todas alertas de todas las reglas aplicadas el 28 de Octubre de 2018 a la organización Org2 podemos consultar: *alert-org2-2018-10-28-** .
- Para consultar las alertas de la regla r1 y r2 aplicadas a todas las organizaciones en Octubre de 2018, podemos consultar: *alert-*-2018-10-*-r1*, *alert-*-2018-10-*-r2* .
- Para consultar todas las alertas de todas las organizaciones de la regla r1, podemos aplicar: *alert-*-*-*-*-r1* .

Y así sucesivamente.

Las alertas son actualizadas en el repositorio privado de la organización mediante consultas a la Blockchain usando el proceso de Talend “WATCHER_GET_ALERTS”.

- Información relativa a datos auxiliares: También se guarda en el repositorio privado de cada organización una copia actualizada de los datos enriquecidos de los siguientes objetos:
  - Organizaciones: índice *organizations*
  - Aplicaciones: índice a*pplications*
  - Tipos de Evento: índice e*venttypes*

    Estos registros son idénticos a la información guardada en la Blockchain, por lo que la definición de la información de estos objetos está descrita en el apartado “Diseño de datos en Blockchain con Hyperledger”. Esta información es relevante que se encuentre en ElasticSearch para facilitar la creación de reportes enriquecidos con Kibana. [10]

10 Esta información es actuaizada el proceso de Talend “WATCHER_FILL_COMMON”.

# Diseño de datos en Blockchain con Hyperledger

Uno de los mayores avances que presenta este proyecto en relación a otros sistemas SIEM en el mercado es la incorporación de una capa Blockchain que posibilita sinergias entre distintas organizaciones que utilicen el sistema.

Para la integración de la tecnología Blockchain en el SIEM distribuido se han usado 2 herramientas:

- Hyperledger Fabric[13]: Hyperledger Fabric es una implementación de Blockchain y uno de los proyectos de Hyperledger alojados por la Fundación Linux. Concebido como base para el desarrollo de aplicaciones o soluciones con una arquitectura modular, Hyperledger Fabric permite que componentes como los servicios de consenso y de membresía, sean plug-and-play. Hyperledger Fabric aprovecha la tecnología de contenedores para alojar contratos inteligentes llamados "chaincode" que comprenden la lógica de aplicación del sistema.
- Hyperledger Composer[20]: Hyperledger Composer es un conjunto de herramientas y un Framework de desarrollo amplio y abierto para facilitar el desarrollo de aplicaciones Blockchain sobre Hyperledger Fabric. El objetivo principal del Framework es acelerar el tiempo de desarrollo y facilitar la integración de aplicaciones de Blockchain con los sistemas empresariales existentes.

Hyperledger Fabric es el sistema Blockchain usado como base. Permite crear una red de actores (en nuestro caso, cada organización de un consorcio) permisionada (es decir, permite que cada organización tenga una serie de reglas definidas sobre qué puede y no puede hacer con los datos del sistema Blockchain) que mantiene trazabilidad y autoría sobre los datos compartidos en el sistema (es decir, se almacena en todo momento quién ha creado/editado/borrado cualquier

registro, de forma histórica para todos los objetos del sistema Blockchain; la autoría de cada acción en el sistema está garantizada gracias al uso de sistemas criptográficos de clave pública / clave privada para identificar a cada actor en el sistema[21]). Además usa un sistema de consenso entre actores del sistema basado en Practical Byzantine Fault Tolerance [22][23] [11], ofreciendo una latencia de más de 3500 transacciones por segundo en el sistema Blockchain [24].

Una vez se han configurado los nodos de Blockchain y las claves públicas y privadas para cada organización del consorcio[12] se debe instalar el Framework de Hyperledger Composer, con el cual se configuran los Smart Contracts y modelos utilizados en el sistema SIEM distribuido.

A continuación se presenta una descripción de los modelos y transacciones usadas en Hyperledger Composer:

- Modelos:
  - Organization: Representa una organización en el sistema (participante). Se compone de su Nombre y un ID (debe ser una cadena de letras en minúscula y/o números).
  - Application: Representa una aplicación en el sistema. Se compone de su Nombre y un ID (debe ser una cadena de letras en minúscula y/o números).
  - EventType: representa un tipo de evento en el sistema. Se compone de su Nombre, ID (debe ser una cadena de letras en minúscula y/o números) y un link de referencia a qué Aplicación pertenece (relación 1 Aplicación tiene N EventType).
  - Rule: representa una regla en el sistema, con las mismas propiedades descritas en “Diseño de datos en ElasticSearch” (Información relativa a Reglas), pero añade 2 propiedades más: “Endorsers”, que identifica una lista de organizaciones que aplican

11 Hyperledger Fabric no tiene sistema de recompensa, por lo que no es un sistema orientado a criptodivisas.
12 La configuración de Hyperledger Fabric para múltiples nodos y organizaciones excede el propósito de este documento. Para mayor información sobre este ámbito, puede consultar [25]

esta regla y “hasEvents”, que identifica si la regla ha generado alguna alerta en el sistema en algún momento.

    - Alert: representa una alerta en el sistema. Se compone de una referencia al ID de la regla que ha generado la alerta y una lista de cadenas de caracteres que representan el contenido de los campos recibidos por la ejecución de la regla para la creación de esta alerta.

- Transacciones:
    - AddAlert: Permite añadir una alerta al sistema. Para ejecutar esta transacción se comprueba que la organización que ejecuta esta transacción aplica esta regla y que los campos de a alerta coinciden con los campos que debería de generar según la alerta.
    - AddRule: Permite añadir una regla nueva al sistema. Para ejecutar esta transacción se comprueba que la regla incluye URLs, DSLs y tipos de evento válidos. La organización que hace la petición de la regla será añadida como organización que aplica a esta regla directamente.
    - EndorseRule: Permite que una organización aplique a una regla guardada en el sistema que no aplica anteriormente.
    - DisendorseRule: Permite que una organización deje de aplicar a una regla. Si la regla tiene 0 organizaciones aplicando a ella y nunca ha generado eventos, la regla es marcada como eliminada en la Blockchain (si la regla ha generado alertas no es marcada como borrada por razones de integridad).
- Consultas predefinidas en la Blockchain:
    - NextAlerts: Obtiene las alertas que se han generado a partir de un tiempo X.
    - RulesOfOrg: Obtiene las reglas que aplican a una organización.

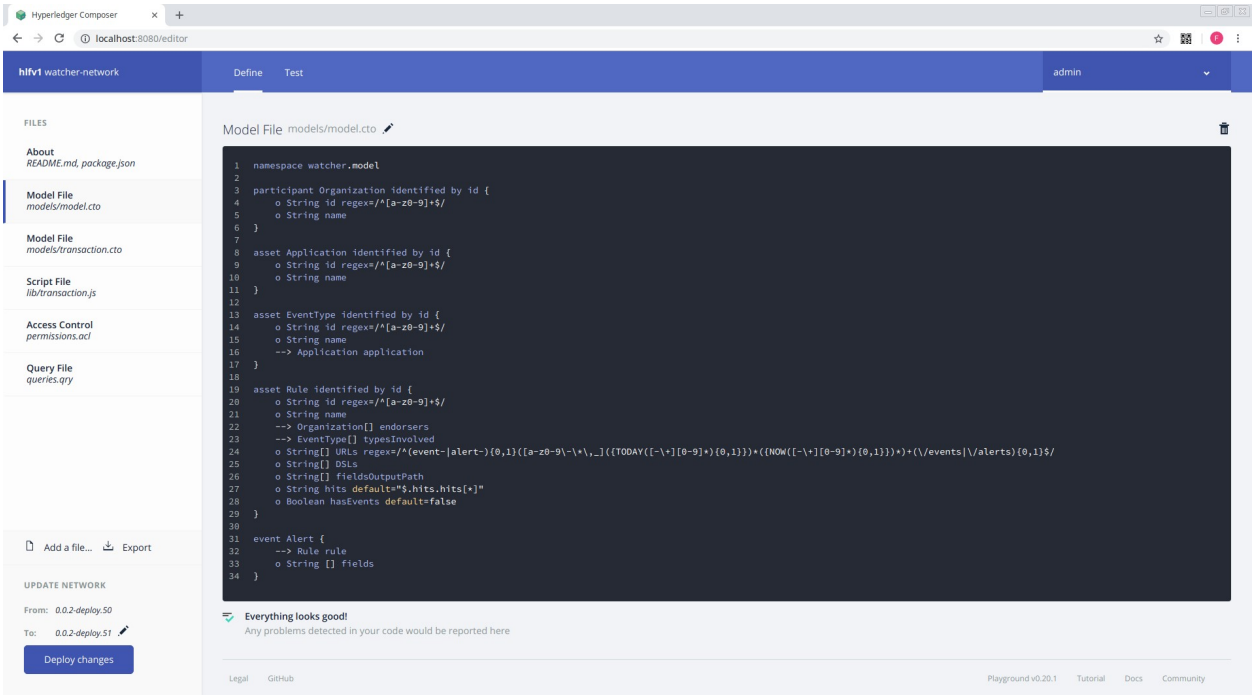



*Figura 10: Definición de Modelos del sistema Blockchain*

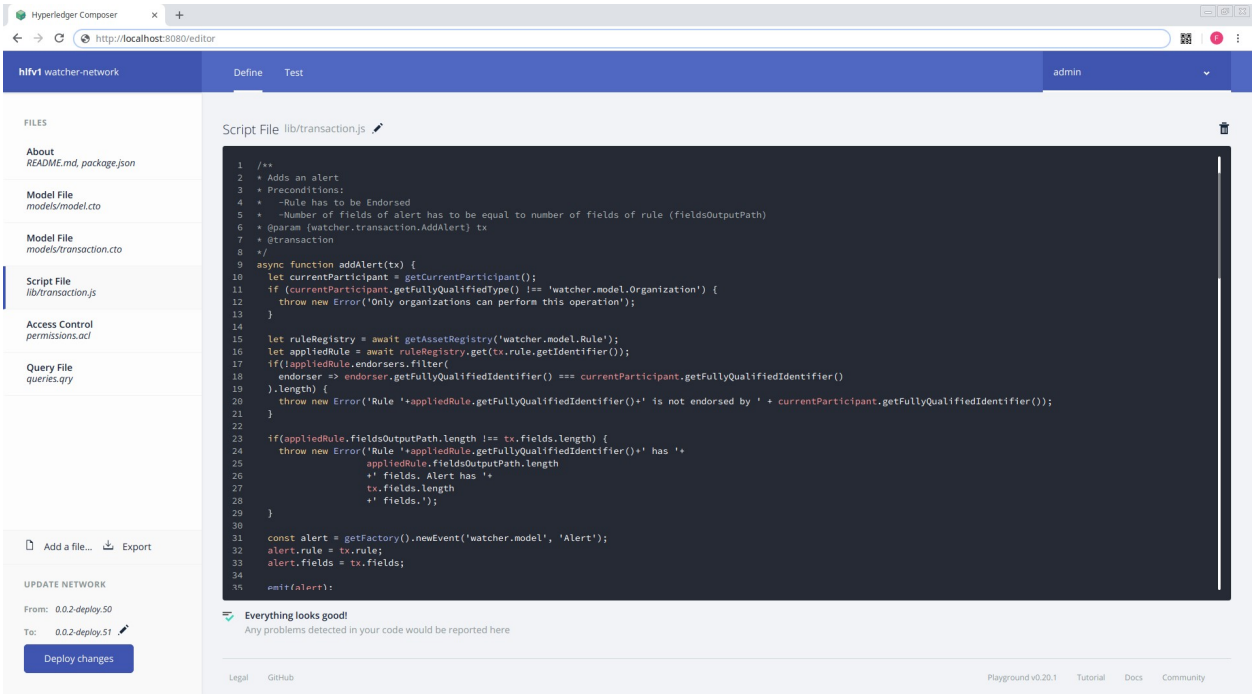



*Figura 11: Definición de Transacciones del sistema Blockchain*

Hyperledger Composer ofrece una API REST para poder integrarse y comunicarse con otros sistemas de forma sencilla, segura y eficiente. Este sistema es el que utiliza el SIEM Distribuido para comunicarse con la Blockchain en cada organización del consorcio y, a través de procesos de ETL descritos con Talend, sincroniza la información de la Blockchain y del repositorio privado (ElasticSearch).

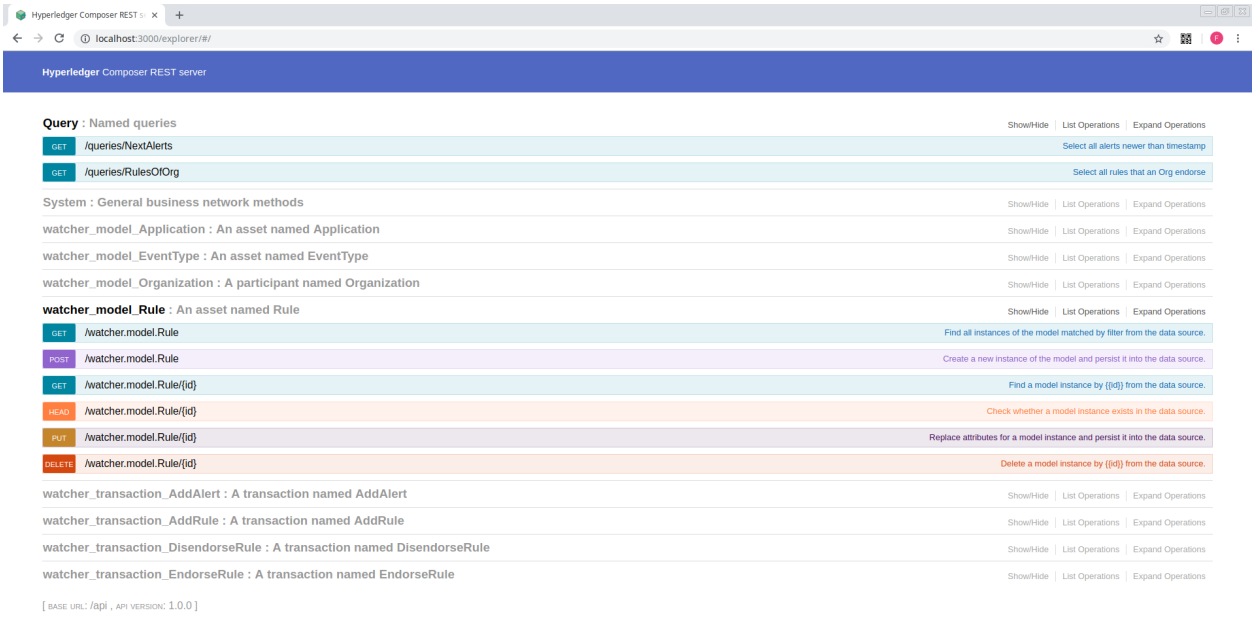



*Figura 12: API de Hyperledger Composer con la definición de las transacciones y modelos del SIEM Distribuido*

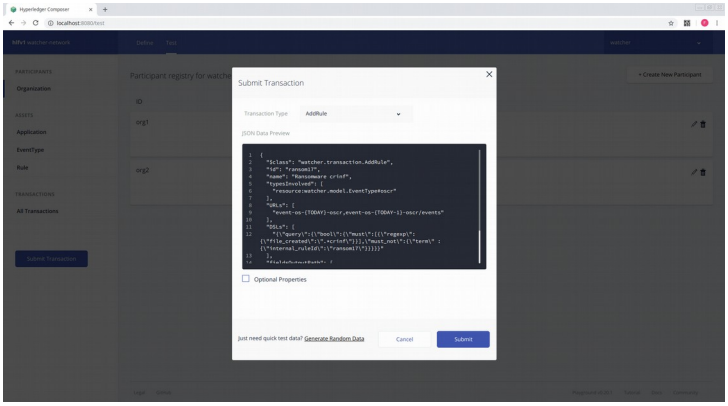



*Figura 13: Ejemplo de transacción “AddRule” para añadir una nueva regla al Sistema Blockchain*

# Diseño de reporting y cuadros de mando en Kibana

Para mostrar el estado de las organizaciones se usan las herramientas ofrecidas por Kibana.

Estos bashborads pueden editarse tal y como el usuario desee. El dashboard por defecto generado para mostrar el estado del consorcio en materia de ciberseguridad (mostrando todas las alertas gestionadas) es el siguiente[13]:

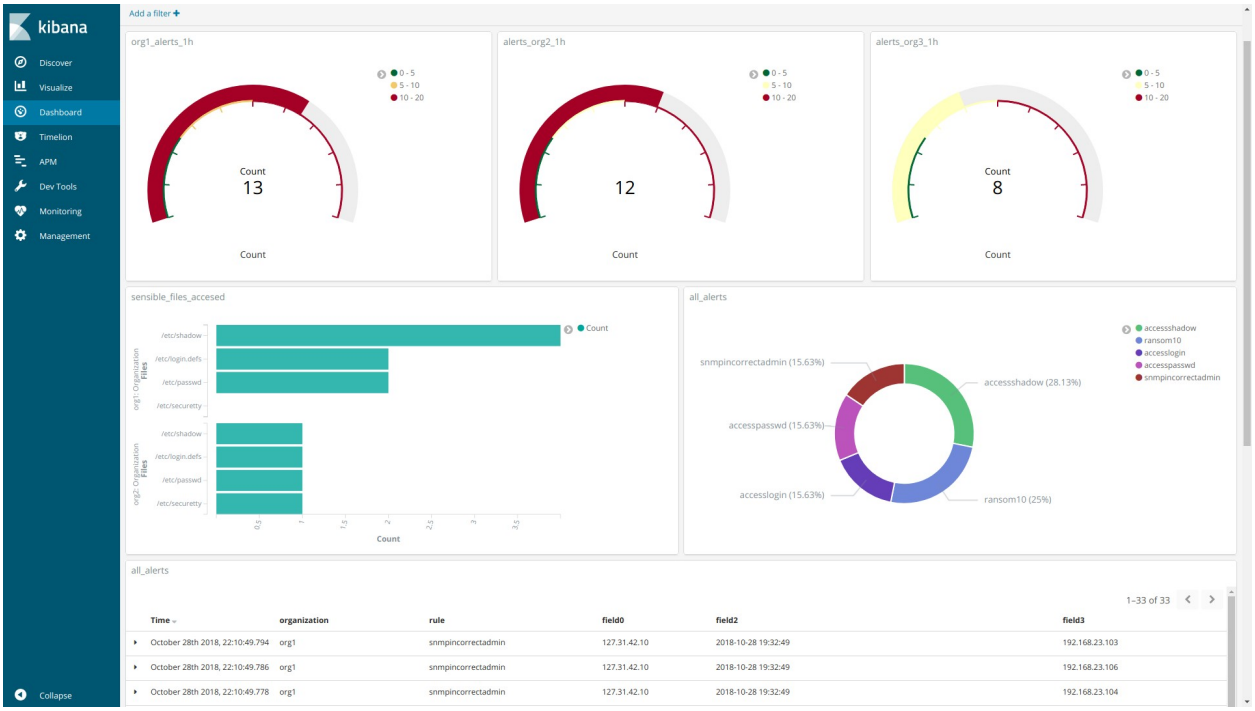



*Figura 14: Dashboard del consorcio en Kibana.*

En él se muestran los siguientes objetos (de izquierda a derecha y de arriba a abajo):

- Número de alertas generadas para la Organización 1 en el tiempo seleccionado (1 hora).
- Número de alertas generadas para la Organización 2 en el tiempo seleccionado (1 hora).
- Número de alertas generadas para la Organización 3 en el tiempo seleccionado (1 hora).
- Número de accesos no controlados a ficheros sensibles en el tiempo seleccionado (1 hora), para cada organización.
- Tipos de alertas recibidas y su porcentaje en el tiempo seleccionado (1 hora), para cada organización.

Por último, se muestra una lista de todas las alertas recibidas para el análisis detallado de cada alerta.

13 El diseño del Dashboard de Eventos se contempla en el documento de Sergio Muñoz Gamarra.

# Tests del Sistema

Con el fin de demostrar la funcionalidad de este SIEM Distribuido hemos desarrollado un entorno de pruebas donde se ha simulado el comportamiento real de varias máquinas de un consorcio de organizaciones para la generación de eventos y alertas que se puedan ajustar a la realidad. En esta simulación se han instalado y configurado todas las tecnologías anteriormente citadas pudiendo analizar el proceso end-to-end (desde que ocurre el evento en una máquina hasta que es reportado en el consorcio).

## Creación de entorno de prueba – Organizaciones y Servidores

La arquitectura física de la simulación es similar a la siguiente:

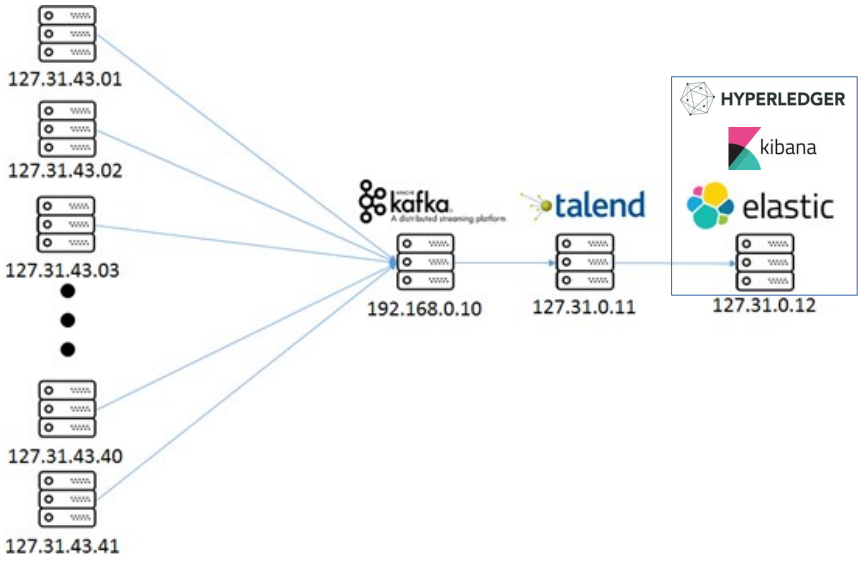



*Figura 15: Arquitectura de la simulación*

El entorno simulado se realiza con un consorcio de organizaciones para demostrar la capacidad de la Blockchain en un sistema multi-organización. En la siguiente figura puede estudiarse la composición con 3 organizaciones: Watcher, Umbrella y Cyberdyne:

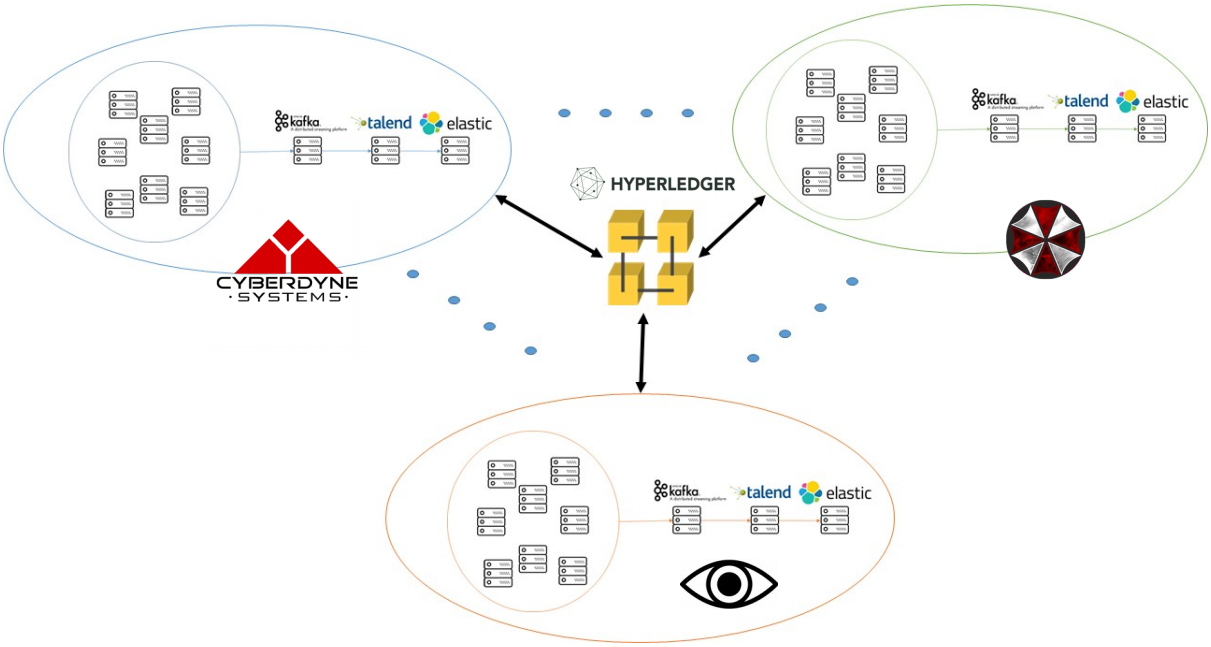



*Figura 16: Composición de entorno de pruebas con 3 organizaciones*

# Creación de reglas

En el entorno de pruebas se han generado las siguientes reglas, aplicadas a cada organización, que generan diversas alertas en base a los eventos generados en cada organización:

| 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 |

| ID | Nombre | URLs | DSL | Org. |
|---|---|---|---|---|
| | | | | |
| 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 |

| ID | Nombre | URLs | DSL | Org. |
|---|---|---|---|---|
| accesssecuretty | 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":"accesssecuretty"}}}}} | 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/ | {"query": {"bool": {"must": [{"regexp": | Watcher, Cyberdyne, Umbrella |

| ID | Nombre | URLs | DSL | Org. |
|---|---|---|---|---|
| | | events | {"file_created":".*zzz"}}],"must_not":{"term":{"internal_ruleId":"ransom4"}}}}} | |
| 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_rule | Watcher, Cyberdyne, Umbrella |

| ID | Nombre | URLs | DSL | Org. |
|---|---|---|---|---|
| | | | Id":"ransom8"}} }}} | |
| 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":"rootnonwor | Watcher, Cyberdyne |

| ID | Nombre | URLs | DSL | Org. |
|---|---|---|---|---|
| | | | kinghours"}}}}} | |
| snmpincorrectadmin | Log-in incorrecto de usuario “admin” | event-snmp-{TODAY}-snla/events | {"query": {"bool": {"must": [{"term": {"user_name.keyword":"admin"}}],"must_not": {"term" : {"internal_ruleId":"snmpincorrectadmin"}}}}} | Watcher, Cyberdyne |

*Tabla 1: Relación de alertas en entorno de pruebas*

Se han generado este conjunto de reglas como ejemplos prácticos de explotación del sistema:

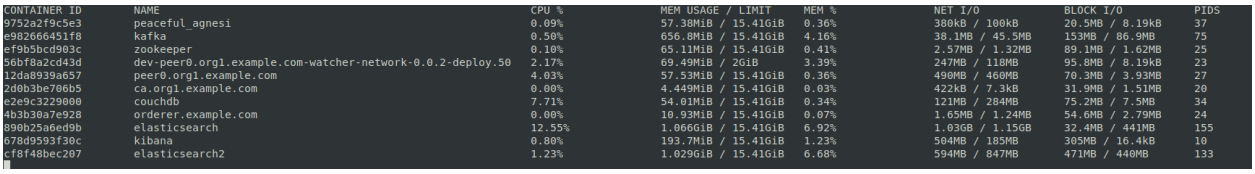
- Con las reglas de detección de cifrado de Ransomware, las organizaciones pueden estar prevenidas si alguna de las organizaciones comienza a recibir este tipo de ataque, detectando su origen y método de infección ágilmente y compartiendo esta información con el resto de organizaciones del consorcio para que puedan protegerse correspondientemente.
- Con las reglas de acceso indebido a ficheros y accesos con usuario ROOT fuera de horas de trabajo se pueden detectar patrones de amenazas: scripts maliciosos que intentan modificar estos ficheros de forma sistemática o indicios de APT[14]
- Con intentos de acceso incorrecto a usuarios en sistemas SNMP pueden detectarse ataques de fuerza bruta por parte de BotNets[27]. Esta información es importante que pueda ser distribuída rápidamente entre las organizaciones del consorcio para de esta manera detectar las IPs de estas redes y mitigar posibles ataques.

14 Amenaza persistente avanzada [26]

# Explotación y gestión de alertas

Con todo lo anteriormente configurado se inician los servicios en el entorno de pruebas.

Para iniciar estos servicios se ha usado el gestor de contenedores Docker. De esta manera es posible ejecutar todos los servicios en la misma máquina de forma eficiente.



| CONTAINER ID | NAME | CPU % | MEM USAGE / LIMIT | MEM % | NET I/O | BLOCK I/O | PIDS |
|---|---|---|---|---|---|---|---|
| 9752a2f9c5e3 | peaceful_agnesi | 0.09% | 57.38MiB / 15.41GiB | 0.36% | 380kB / 100kB | 20.5MB / 8.19kB | 37 |
| e902666451f8 | kafka | 0.50% | 656.8MiB / 15.41GiB | 4.16% | 38.1MB / 45.5MB | 153MB / 86.9MB | 75 |
| ef9b5bcd903c | zookeeper | 0.10% | 65.11MiB / 15.41GiB | 0.41% | 2.57MB / 1.32MB | 89.1MB / 1.62MB | 25 |
| 56bf8a2cd43d | dev-peer0.org1.example.com-watcher-network-0.0.2-deploy.50 | 2.17% | 69.49MiB / 2GiB | 3.39% | 247MB / 118MB | 95.8MB / 8.19kB | 23 |
| 12da8939a657 | peer0.org1.example.com | 4.03% | 57.53MiB / 15.41GiB | 0.36% | 490MB / 460MB | 70.3MB / 3.93MB | 27 |
| 2d0b3be706b5 | ca.org1.example.com | 0.00% | 4.449MiB / 15.41GiB | 0.03% | 422kB / 7.3kB | 31.9MB / 1.51MB | 20 |
| e2e9c3229000 | couchdb | 7.71% | 54.01MiB / 15.41GiB | 0.34% | 121MB / 284MB | 75.2MB / 7.5MB | 34 |
| 4b3b30a7e928 | orderer.example.com | 0.00% | 10.93MiB / 15.41GiB | 0.07% | 1.65MB / 1.24MB | 54.6MB / 2.79MB | 24 |
| 890b25a6ed9b | elasticsearch | 12.55% | 1.066GiB / 15.41GiB | 6.92% | 1.03GB / 1.15GB | 32.4MB / 441MB | 155 |
| 678d9593f30c | kibana | 0.80% | 193.7MiB / 15.41GiB | 1.23% | 504MB / 185MB | 305MB / 16.4kB | 10 |
| cf8f48bec207 | elasticsearch2 | 1.23% | 1.029GiB / 15.41GiB | 6.68% | 594MB / 847MB | 471MB / 440MB | 133 |

*Figura 17: Estadísticas de funcionamiento de los contenedores Docker con todo el sistema SIEM Distribuido para una organización del consorcio*

Se van a lanzar los siguientes ataques al sistema de prueba:

- Ataque de Ransomware “xxx” a la organización Cyberdyne
- Acceso y edición de archivos críticos a las organizaciones Umbrella y Watcher
- Ataques de fuerza bruta en login incorrecto por SNMP al usuario “admin” en la organización Watcher.

Tras la ejecución de los ataques se han obtenido los reportes y resultados del sistema mostrados en la figura que aparece en la sección “Diseño de reporting y cuadros de mando en Kibana”.

Todas las alertas y eventos generados han cumplido las expectativas de tiempos en el test (generadas en menos de 10 segundos desde que se crea la alerta en el sistema de origen hasta que se muestra en los sistemas de reporting).

# Conclusiones

Tras demostrar el diseño, desarrollo y explotación del sistema a continuación se muestra una comparativa de este sistema con otras herramientas en el mercado y una propuesta de mejoras futuras para el proyecto.

## Comparativa con otras herramientas en el mercado

Existen otras soluciones SIEM en el mercado y otras herramientas que pueden ser usadas para sistemas SIEM. Este proyecto de SIEM Distribuido se diferencia con otros sistemas SIEM principalmente por la adición de un sistema Blockchain para poder realizar sinergias con otras organizaciones. Esto hace que “Watcher” tenga mejoras con respecto a otros proyectos Open Source SIEM (como ELK Stack[3]):

- Integridad de los datos: Las alertas y reglas se almacenan en un registro inmutable, formado por bloques de transacciones que, gracias a sus características criptográficas, son casi imposibles de modificar o falsificar.
- Consenso: Mediante el algoritmo de consenso de la Blockchain, las organizaciones definen unívocamente en conjunto de de operaciones que se grabará en Blockchain.
- Tolerancia a fallos gracias al sistema Blockchain, los datos se replican en todos los servidores que participan en las transacciones. Por lo tanto, toda la información es accesible (a nivel de alertas y reglas) si al menos uno de los servidores de cualquier organización funciona correctamente.

Si diseccionamos el sistema por tecnologías usadas, también se ha realizado una comparativa de herramientas y una explicación de por qué se han elegido estas y no otras

(siempre se ha partido del requisito de que estas herramientas sean o tengan versiones Open Source para su aplicación sin restricciones en el proyecto):

- 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:

  - Posee un contector 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.
  - 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[28].
  - 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, la IP de origen, etc.
  - Es posible instalar el agente en cualquier sistema operativo.

- 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[29]:

- 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 amortiguardor 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 [30]:

  - 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 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 Oct 2018 | Rank Sep 2018 | Rank Oct 2017 | DBMS | Database Model | Score Oct 2018 | Score Sep 2018 | Score 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 |

*Figura 18: Comparativa de motores de búsqueda [31]*

- 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 esperas innecesarias entre ambos.
  - Creación de dashboards 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.

- 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:


*Figura 19: Cuadrante Mágico de herramientas de integración de datos [32]*

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:

  - Debe de permitir gran número de transacciones por segundo que deben de ser validadas en el menor tiempo posible.
  - Debe de ser versátil y permitir el desarrollo de procesos y transacciones propias (Smart Contracts).
  - Debe de permitir gestionar y añadir nuevos nodos y participantes de forma dinámica.
  - Debe de ser estable y maduro para ser usado de forma productiva.
  - 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<br>Can be made using chaincode | None |

*Figura 20: Comparativa entre Ethereum, Hyperledger Fabric y R3 Corda[33]*

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:

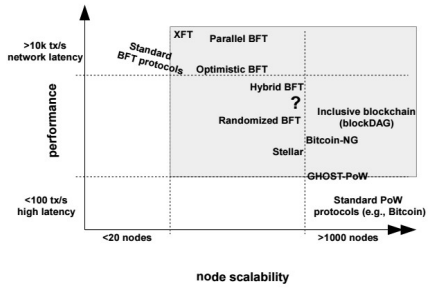



*Figura 21: Comparativa entre diversos mecanismos de consenso[34]*

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

# Mejoras Futuras

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:
  - Basar la entrada de nuevas alertas de otras organizaciones en WebSockets[35].
  - 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).
- Añadir integración con sistema de gestión de tickets/incidencias (como Jira).

*Todo el código y configuraciones para este proyecto se pueden encontrar en el siguiente repositorio de GitHub (con las colaboraciones de Francisco Serrano Carmona y Sergio Muñoz Gamarra en forma de commits al repositorio): https://github.com/frandai/watcher*

# Referencias

## Bibliografía

[1] Dr.Dobb's Journal (Octubre 2018). SIEM: A Market Snapshot. Recuperado de: http://www.drdobbs.com/siem-a-market-snapshot/197002909
[2] Elli Androulaki, Artem Barger (Enero 2018). Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. Recuperado de: https://arxiv.org/abs/1801.10228
[3] Urfeena Elahi (Julio 2018). Elastic Stack — A Brief Introduction. Recuperado de: https://hackernoon.com/elastic-stack-a-brief-introduction-794bc7ff7d4f
4: Leon, Sixto O, ecurity: Defense Against Crime. Manila: National Book Store., 1976
[5] CCN-CERT (2015). CERT - Equipo de reacción rápida ante incidentes informáticos. Recuperado de: https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html?n=173.html
[6] Balazs Scheidler (Octubre 2018). Syslog-ng Github. Recuperado de: https://github.com/balabit/syslog-ng
[7] Ismael Juma (Octubre 2018). Github Kafka. Recuperado de: https://github.com/apache/kafka/
[8] Pierre Bailly-Ferry (Octubre 2018). Talend Open Studio Github. Recuperado de: https://github.com/Talend/tbd-studio-se
[9] SAS (Octubre 2018). What is ETL and why it matters. Recuperado de: https://www.sas.com/en_us/insights/data-management/what-is-etl.html
[10] Oracle (Octubre 2018). ¿Qué es la tecnología Java y para qué la necesito?. Recuperado de: https://www.java.com/es/download/faq/whatis_java.xml
[11] Shay Banon (Octubre 2018). Elastic Search Github. Recuperado de: https://github.com/elastic/elasticsearch
[12] Spencer Alger (Octubre 2018). Kibana Github. Recuperado de: https://github.com/elastic/kibana
[13] Jason Yellick (Octubre 2018). Github Hyperledger Fabric. Recuperado de: https://github.com/hyperledger/fabric
[14] TechTarget (Octubre 2018). Guide to SOA and the cloud. Recuperado de: https://searchcloudapplications.techtarget.com/essentialguide/Guide-to-SOA-and-the-cloud
[15] w3schools (Octubre 2018). JSON - Introduction. Recuperado de: https://www.w3schools.com/js/js_json_intro.asp
[16] Tecnopedia (Octubre 2018). Multithreading. Recuperado de: https://www.techopedia.com/definition/24297/multithreading-computer-architecture
[17] ElasticSearch Reference (Octubre 2018). Query DSL. Recuperado de: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html
[18] Elastic (Octubre 2018). ElasticSearch Index. Recuperado de: https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html#_index
[19] AWS (Octubre 2018). Trabajar con JSONPath. Recuperado de: https://docs.aws.amazon.com/es_es/kinesisanalytics/latest/dev/about-json-path.html
[20] Linux foundation (Octubre 2018). Hyperledger Composer documentation. Recuperado de: https://hyperledger.github.io/composer/latest/introduction/introduction.html

[21] Anil Ambati (Octubre 2018). Fabric CA User’s Guide. Recuperado de: https://hyperledger-fabric-ca.readthedocs.io/en/latest/users-guide.html#overview
[22] Dileban Karunamoorthy (Octubre 2018). Hyperledger Fabric - Consensus. Recuperado de: https://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html
[23] Miguel Castro and Barbara Liskov (Febrero 1999). Practical Byzantine Fault Tolerance. Recuperado de: http://pmg.csail.mit.edu/papers/osdi99.pdf
[24] Marko Vukolić (Febrero 2018). Behind the Architecture of Hyperledger Fabric. Recuperado de: https://www.ibm.com/blogs/research/2018/02/architecture-hyperledger-fabric/
[25] Priyansh Jain (Junio 2015). Hyperledger Fabric: Transitioning from Development to Production. Recuperado de: https://dev.to/presto412/hyperledger-fabric-transitioning-from-development-to-production-4dch
[26] Gartner (Enero 2012). Best Practices for Mitigating AdvancedPersistent Threats. Recuperado de: http://apac.trendmicro.com/cloud-content/apac/pdfs/solutions/enterprise/best_practices_for_mitigating_apts_224682.pdf
[27] Dennis Fisher (Abril 2013). ¿Qué es un botnet?. Recuperado de: https://www.kaspersky.es/blog/que-es-un-botnet/755/
[28] Syslog-ng (Octubre 2018). Syslog-ng vs. rsyslog comparison. Recuperado de: https://www.syslog-ng.com/products/open-source-log-management/syslog-ng-rsyslog-comparison.aspx
[29] Syslog (Octubre 2018). Syslog-ng vs. rsyslog comparison. Recuperado de: https://www.syslog-ng.com/products/open-source-log-management/syslog-ng-rsyslog-comparison.aspx
[30] Greenice (Febero 2018). Best Open Source Search Platform Comparison. Recuperado de: https://greenice.net/elasticsearch-vs-solr-vs-sphinx-best-open-source-search-platform-comparison/
[31] DB-Engines (Octubre 2018). DB-Engines Ranking of Search Engines. Recuperado de: https://db-engines.com/en/ranking/search+engine
[32] Gartner (Julio 2018). Magic Quadrant for Data Integration Tools. Recuperado de: https://www.gartner.com/doc/3883264/magic-quadrant-data-integration-tools
[33] Micobo (Marzo 2016). 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
[34] Tjark Friebe (octubre 2017). Bitcoin, Ethereum, and Hyperledger Fabric—which one wins?. Recuperado de: https://medium.com/blockchainspace/3-comparison-of-bitcoin-ethereum-and-hyperledger-fabric-cd48810e590c
[35] Hyperledger Composer (Octubre 2018). Publishing events from the REST server. Recuperado de: https://hyperledger.github.io/composer/latest/integrating/publishing-events