Trabajo de Fin de Grado

Adquisición, normalización, enriquecimiento y visualización de eventos de ciberseguridad con tecnologías Big Data.

Acquisition, standardization, enrichment and visualization of cyber security events with Big Data technologies.

Juan Martínez Hurtado de Mendoza

San Cristóbal de La Laguna,6 de julio de 2020

D. **Sergio Fernando Alonso Rodríguez**, con N.I.F. 42.093.441-Z profesor Titular de Universidad adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

*"*Adquisición, normalización, enriquecimiento y visualización de eventos de ciberseguridad con tecnologías Big Data.*"*

ha sido realizada bajo su dirección por D. Juan Martínez Hurtado de Mendoza, con N.I.F. 06020569-C.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de julio de 2020.

Agradecimientos

A mi familia, que me han dado fuerzas para seguir adelante en los momentos duros.

A mis compañeros de universidad por la ayuda prestada cuando la necesitaba.

Y a la empresa redBorder, por darme la oportunidad de trabajar con ellos e iniciar mi carrera profesional, sobre todo a mi compañero de trabajo Ian por la ayuda prestada.

Licencia

Resumen

En este TFG se pretende integrar diferentes productos relacionados con el ámbito de la ciberseguridad, que surgen de la necesidad de buscar nuevas zonas de partners dentro del marco de trabajo de la empresa redBorder. Algunos de estos son: Secret Server, Forescout, Gravity Zone, Rapid7 InsightVM…

El objetivo de este trabajo consistirá en la instalación de dichos productos, y la visualización de sus eventos en tiempo real en el manager web, donde podremos llevar un control más exhaustivo sobre lo que está pasando en nuestro sistema a través de los productos mencionados. Adicionalmente de la visualización en el manager (con su correspondiente enriquecimiento y normalización de los datos), se crearán también reglas de correlación con las que podremos extraer nueva información, la cual nos será también bastante útil, pudiendo relacionar datos de los diferentes módulos, por ejemplo, y opcionalmente, creando un formulario que facilite la creación de reglas.

En este proyecto están involucrados diferentes servicios como Apache Druid, Kafka, Logstash, Rsyslog, por mencionar los más importantes,y de los cuales hablaremos más adelante, explicando el papel de cada uno.

Palabras Clave: manager, reglas, enriquecimiento, normalización.

Abstract

In this TFG we aim to integrate different products related with cybersecurity, in order to search for new partners zones, all within the work area of redBorder. Some of those are: Secret Server, Forescout, Gravity Zone, OpenVPN…

The objective of this work will consist of the installation of the mentioned products, and the visualization of events in real-time on the web manager, where we can have a better control of the things happening in our system using those products. Additionally, we'll also create correlation engine rules in order to extract more useful information, correlating information coming from different modules, for example, and optionally, creating a form so the user can create rules in a easier way.

In this project there are multiple services involved, like Apache Druid, Kafka, Logstash,Rsyslog, to mention the most important ones, that we will explain later, in depth, in the document.

Keywords: manager, rules, enrichment, normalization

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Desde hace unos años, las palabras ciberseguridad y Big Data se han vuelto un estándar entre las empresas, ya que como bien sabemos, vivimos en la era de la información, donde cada día podemos generar millones de bytes de contenido que las empresas tienen que saber manejar y visualizar para extraer conocimiento útil. Además, ya que la informática es una herramienta común en los negocios, necesitamos de diferentes medidas de seguridad que nos ayuden a evitar no vernos expuestos a grandes riesgos.

Cuando hablamos de ciberseguridad, entendemos el concepto como la seguridad de la tecnología de la información, puesto que dentro de ella encontramos diferentes métodos y técnicas con las que podemos proteger nuestro sistema.

1.1 Motivación

Con objeto de buscar nuevas zonas de partners y de dotar de mayor control al administrador de sistemas que gestione el manager, en la empresa redBorder se plantea la necesidad de integrar diferentes productos relacionados con el ámbito de la ciberseguridad. Algunos de estos productos son:

- BitDefender Gravity Zone.
- Rapid7 Insight
- F-secure
- Forescout CounterACT

Aparte de los mencionados, también se deberán integrar otros productos ya instalados en el entorno de redBorder, pero de los cuales no estamos llevando a cabo la monitorización de logs como OpenVPN, Untangle o Kaspersky. Todo esto nos ayudará a tener un mejor control sobre nuestro sistema, pudiendo controlar las conexiones vpn o end-points específicos, por ejemplo.

1.2 Objetivos

Con el fin de cubrir estas necesidades expuestas en el punto anterior, podríamos establecer 3 objetivos a cumplir:

- Tratamiento de los datos enviados por syslog mediante la herramienta Logstash.
- Visualización de los datos mediante la creación de Dashboards.
- Creación de reglas de correlación para extraer nueva información útil.

A grandes rasgos, estas son las metas que debemos alcanzar para el correcto desarrollo de este proyecto, donde una vez conseguidas, el cliente tendrá la posibilidad de monitorizar los eventos generados dentro de cualquiera de los productos mencionados anteriormente mediante la instalación de un sencillo script.

1.3 Metodología

En primer lugar, se va a llevar a cabo una investigación sobre la infraestructura en la que se basa el manager de redBorder, investigando los servicios que lo sustentan como Apache Zookeeper, Rsyslog, Apache Kafka, Logstash o Apache Druid.

Una vez comprendido el funcionamiento de dichos servicios, entraremos en la primera fase del desarrollo donde contactaremos con las diferentes empresas para solicitar una demo de sus productos. Además, habrá que preparar las máquinas virtuales para su instalación.

Cuando tengamos dichas máquinas preparadas e instaladas, tendremos que configurarlas para que puedan exportar syslog a un topic de kafka, en donde el siguiente paso será tratar los datos mediante Logstash. Aquí, además de la normalización y enriquecimiento de los datos, también tendremos que crear el script que permita la automatización de la creación en el manager de todos los campos parseados con Logstash.

Tratados ya dichos datos, el siguiente paso será la visualización de ellos en el manager. Para ello, habrá que proponer casos de uso para poder crear widgets donde se muestre información útil al administrador, de manera que se pueda extraer conocimiento de ellos rápidamente. Además, para reforzar esta última idea, se crearán reglas de correlación en las que podremos relacionar información de tráfico de red con información proveniente del syslog, de manera que podríamos ver por ejemplo, el top de aplicaciones usadas por el top usuarios conectados a la vpn.

Al finalizar cada fase, se iniciará otra de testeo en la que comprobaremos el correcto funcionamiento de lo desarrollado, de manera que así no arrastraremos errores a las fases posteriores.

Capítulo 2.
Descripción del sistema

En este capítulo vamos a describir someramente las capacidades del producto redBorder y el papel que desempeña en el ámbito de la ciberseguridad y el análisis de datos, para más adelante, habiendo comprendido la herramienta, pasar a explicar de manera más detallada los diferentes servicios que los componen.

redBorder es una plataforma de Ciberseguridad y Análisis de Tráfico de Red en tiempo real. Está basada en tecnologías Big Data para empresas y proveedores de servicios. Tiene una gran capacidad para la ingesta y el análisis de grandes cantidades de datos ya que ha sido desarrollada y testeada en entornos muy demandantes como el análisis de tráfico de red, ciberseguridad y el análisis e informe de las conexiones a través de redes inalámbricas en determinados lugares.[1]

redBorder unifica las diferentes fuentes de datos que los módulos y sensores proporcionan. Además, incluye otra serie de funcionalidades: dashboards e informes personalizables, correlación, motores de análisis, administración de usuarios y almacenamiento por capas.

Dichos **módulos** se encuentran dentro de la plataforma y son instalables como si de plugins se tratasen. Cada módulo (o app) se centra en digerir nuevos tipos de datos.

Los **sensores** se encuentran fuera de la plataforma, pero son administrados desde ella y su función es la creación de nuevos tipos de datos y sus capacidades para la inspección.

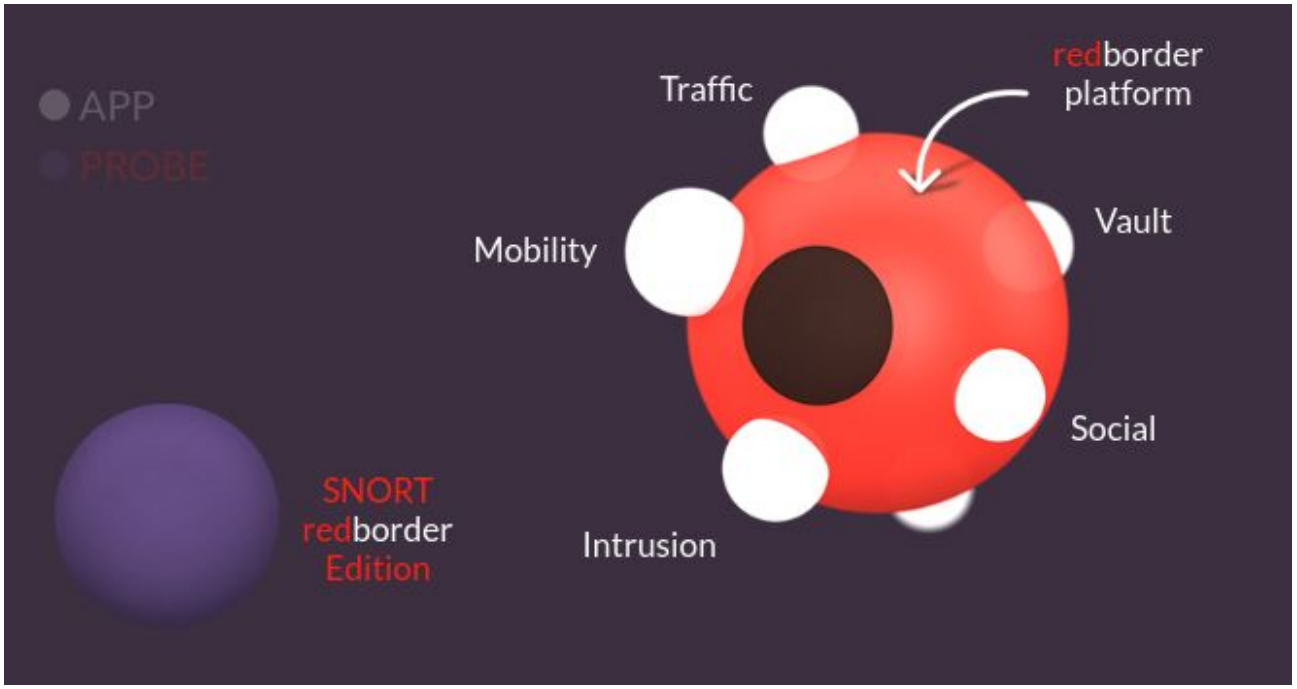


Figura 2-1. Esquema Plataforma redBorder.

Dependiendo de los módulos integrados en la plataforma, el usuario verá en la barra superior uno o algunos de las apps mostradas en la imagen de arriba.

- **Tráfico.** La recolección de información sobre el uso de la red y la localización de los dispositivos es muy importante en diferentes contextos: descubrimiento de amenazas de seguridad, detección de problemas de conectividad, aprendizaje y uso del comportamiento de multitudes, optimización de rutas y monetización de la infraestructura.
- **Movilidad.** Esta App usa Wi-fi e información de localización de elementos en nuestra red, para mostrar, entre otras cosas, información útil sobre el movimiento de los dispositivos en nuestra organización. De esta manera, con esta App podemos saber en todo momento los dispositivos que están presentes en nuestra organización, su fidelidad, la calidad de la señal,etc. Todos estos datos se presentan en tiempo real, aunque también se puede consultar el histórico de datos.
- **Intrusión.** redBorder ofrece una solución de código abierto para protección IPS/IDS. Los eventos generados por cientos de sensores IPS/IDS alcanzarán un punto central donde son recolectados, enriquecidos y almacenados en pipelines escalables a tiempo real. Se le da al usuario la capacidad de supervisar y buscar los eventos de su

interés, para visualizarlos relacionándolos con otras fuentes de datos y así poder tomar acciones.

- **Social.** Mide el éxito de tus campañas de marketing en redes sociales. Te permite recoger información de interés para ti y tus clientes a través de Twitter. Descubre cuáles son sus reacciones basadas en análisis de sentimientos y monitoriza hashtags relevantes o usuarios. Si los usuarios tienen sus tweets geolocalizados, podrás ver cómo reaccionan a un evento específico.
- **Vault.** redBorder Vault colecciona, enriquece, correlaciona y almacena logs de manera segura y escalable. Usa almacenamiento para beneficiarse de los conocimientos que estábamos ignorando hasta ahora.

Capítulo 3.
Adquisición de los datos.

Una vez explicado brevemente el funcionamiento de la plataforma redBorder, en los siguientes capítulos vamos a hablar conjuntamente del trabajo que se ha tenido que realizar para cumplir los objetivos, y a su vez se explicarán los diferentes servicios/herrramientas que se han utilizado según vayan entrando en juego.

De esta manera, los siguientes capítulos se van a corresponder con las diferentes fases que tenemos que superar para desarrollar con éxito este trabajo. Estas son adquisición, normalización, enriquecimiento y visualización de los datos. El último capítulo se corresponderá con las reglas de correlación, las cuales utilizaremos para conseguir información extra.

En este capítulo, por tanto, vamos a hablar sobre la adquisición de los datos. Cuando hablamos de datos, en realidad nos referimos a los logs que los diferentes productos que vamos a integrar están exportando. Podríamos definir un log como un registro de un evento, es decir, un mensaje que un servicio envía para dejar huella de cierta acción que se ha realizado.

En esta primera fase, y previamente a poder recibir dichos logs en nuestra fuente de datos, lo primero que haremos será, obviamente, instalar los productos de los cuales queremos hacer la integración. Como de momento no vamos a trabajar con versiones premium del producto, lo que haremos será contactar con las diferentes empresas y solicitar una demo del producto.

Una vez recibida la demo, habrá que instalar dicho producto en una máquina virtual, donde configuraremos un Adaptador Puente para que el servicio se pueda comunicar con el exterior. Este proceso es el mismo para todos los productos, y no requiere de ninguna dificultad así que no entraremos en detalle. Sí saber que una vez instalado, hay una fase de aprendizaje en la que tenemos que averiguar cuales son las posibilidades que el producto nos ofrece, para más adelante llevar un control completo de todas las acciones que ocurren en el producto. En definitiva, tenemos que aprender a manejar el servicio.

3.1 Rsyslogd

El primer servicio que entra en juego en el desarrollo de este trabajo es Rsyslogd. Este es una utilidad del sistema que provee soporte para el

registro de mensajes, el cual la mayoría de programas modernos utilizan. Ofrece un rendimiento bastante alto, un diseño modular y buenas características de seguridad.

Empezó como un syslogd normal, pero evolucionó para permitir entradas de diferentes fuentes de datos, las cuales transforma y devuelve los resultados a otros destinos.[2]

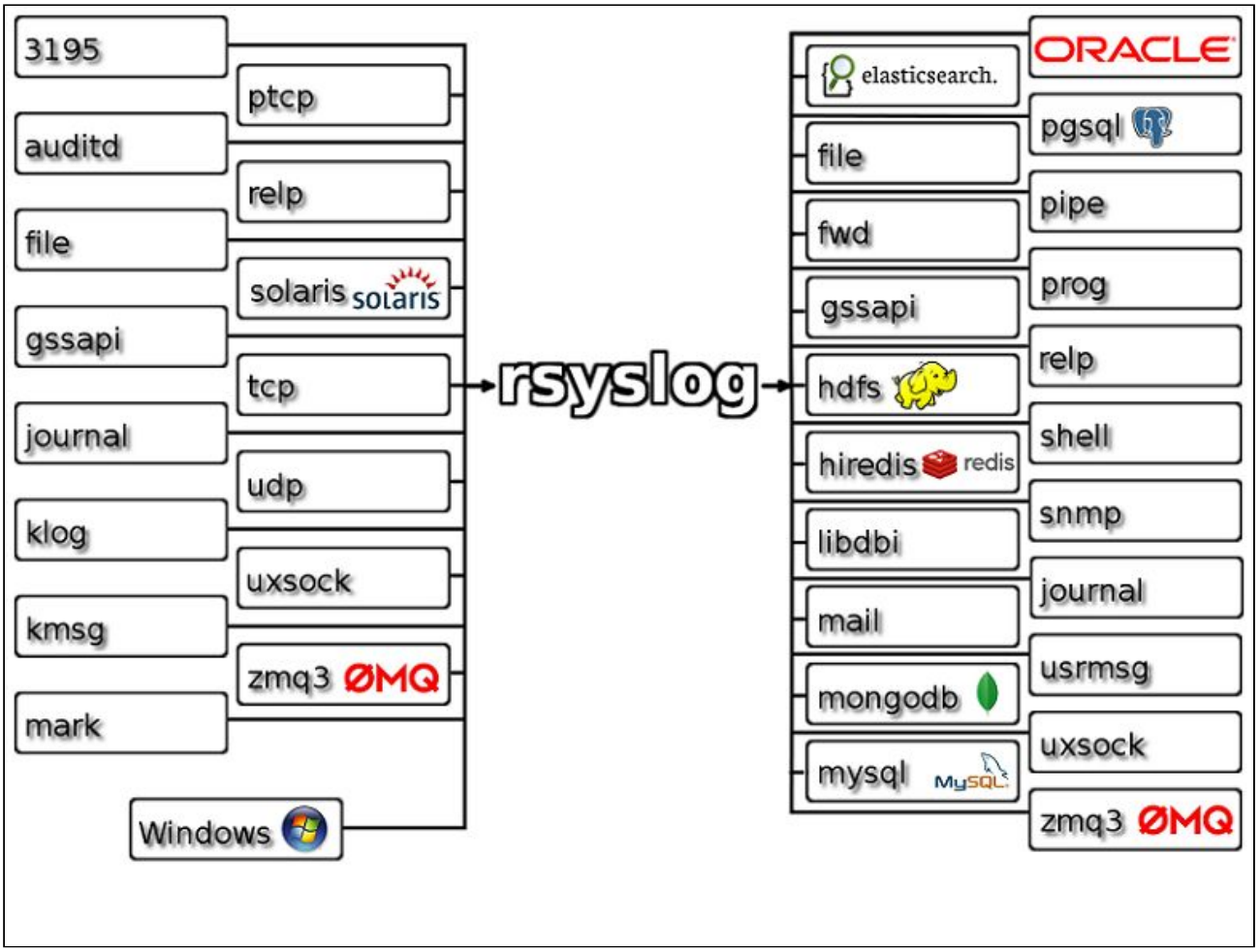


Figura 3-1. Entradas y salidas de Rsyslog.

3.2 Apache Kafka

Apache Kafka va a ser la herramienta a usar para almacenar temporalmente los mensajes provenientes de syslog. Es una aplicación multiplataforma de código abierto desarrollada por Apache Software Foundation y especializada en el procesamiento de flujos de datos.

Ha sido diseñada para optimizar la transmisión y el procesamiento de los flujos de datos que se intercambian entre la fuente y el receptor por conexión directa. Un problema típico que soluciona Kafka es la imposibilidad de almacenar datos en la caché si el destinatario no está disponible.[3]

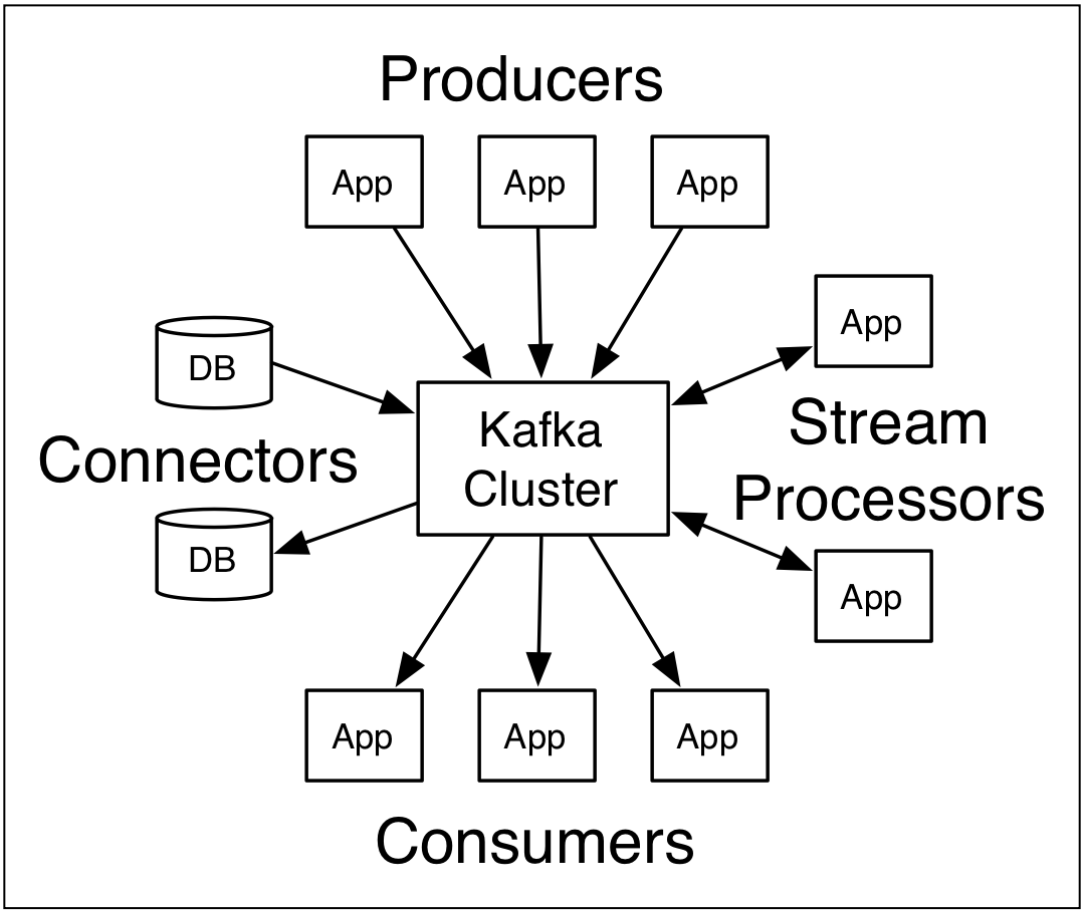


Figura 3-2. Esquema Apache Kafka

Kafka tiene 5 APIs que lo conforman:

- **Producer API** (Productor). Permite a una aplicación publicar una serie de registros/mensajes a uno o más topics de Kafka.
- **Consumer API** (Consumidor). Permite a una aplicación subscribirse a uno o más topics y procesar los registros/mensajes que son producidos por dicho topic.
- **Streams API.** Permite a una aplicación actuar como un procesador de registros, consumiendo una entrada desde uno o más topics y produciendo una salida a uno o más topics, transformando de manera efectiva una entrada en una salida.

- **Connector API**. Permite construir y ejecutar productores o consumidores reutilizables que conectan un topic de Kafka con aplicaciones existentes o sistemas de datos.
- **Admin API.** Permite administrar e inspeccionar topics, brokers y otros objetos de Kafka.

3.2.1 Topics y Logs.

Vamos a explicar una de las características centrales del Apache Kafka: los topics.

Un topic es una categoría donde ciertos mensajes son publicados. Los topics en Kafka siempre permiten múltiples suscriptores, lo cual significa que los topics pueden tener cero, uno o más consumidores que se encargan de leer los mensajes almacenados en ellos. Para cada topic, Kafka mantiene un registro particionado:

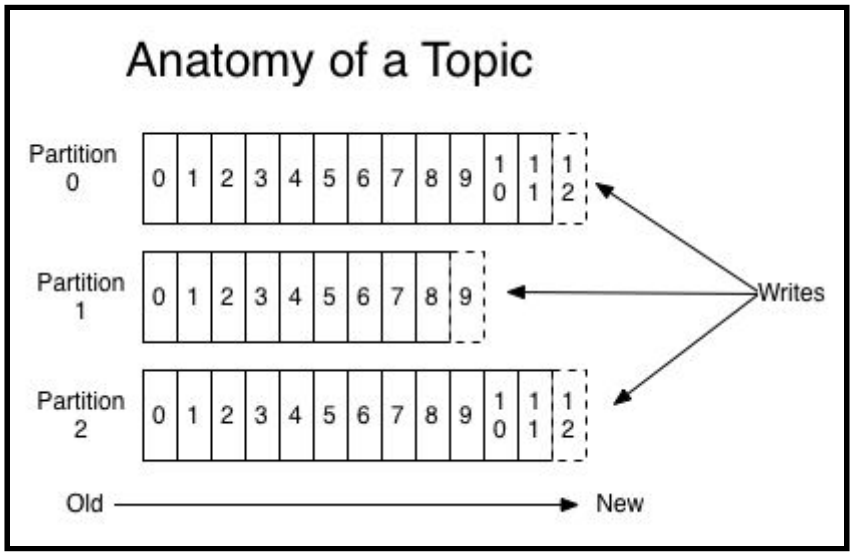


Figura 3-3. Anatomía de un Topic.

Cada partición es una secuencia ordenada e inmutable de mensajes que son continuamente añadidos a una estructura de registros commit. Los mensajes dentro de cada partición reciben un número, el cual representa el offset para ese mensaje dentro de la partición correspondiente.

El cluster Kafka almacena de manera persistente todos los mensajes publicados (tanto si han sido consumidos como si no), usando un período configurable de duración. Por ejemplo, si establecemos una política de retención de dos días, podremos consumir un mensaje aún dos días después de haber sido publicado. Pasado este tiempo, este mensaje se descarta para

liberar espacio. El rendimiento de Kafka es constante respecto al tamaño de los datos así que no hay problemas en almacenar datos durante un largo período de tiempo.

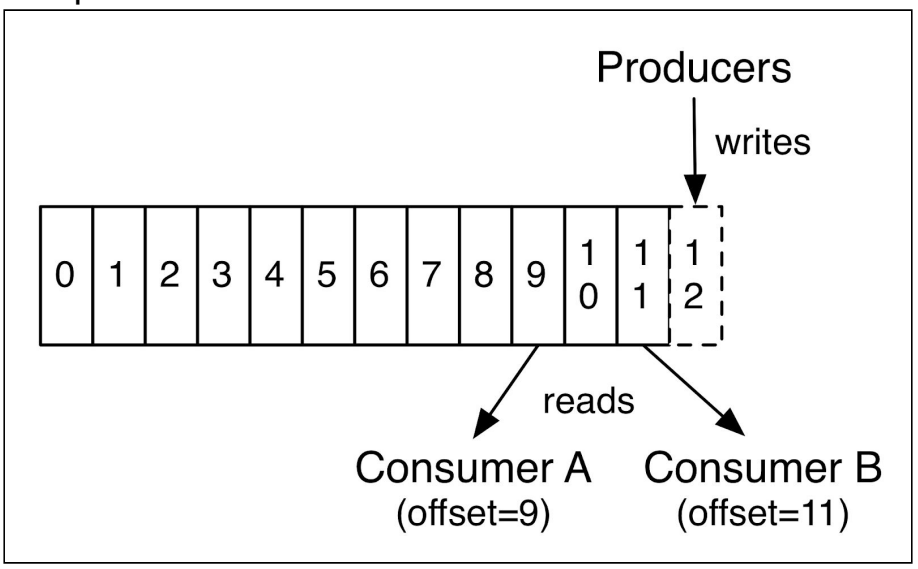


Figura 3-4. Productores y Consumidores.

3.3 Adquiriendo los datos

Una vez explicados los servicios que van a desempeñar un papel en esta primera fase, pasemos a explicar lo desarrollado para conseguir el objetivo de esta primera etapa: la adquisición de los datos.

Como se mencionó al principio del documento, tenemos una serie de productos que vamos a integrar con la plataforma redBorder, con objeto de buscar nuevas zonas de partners, y además, dotar de mayor seguridad al producto mediante la detección de eventos de ciberseguridad y el análisis de los datos. Los datos los vamos a almacenar en un topic de Kafka llamado rb_vault. Pero para que dichos logs lleguen ahí, primero habrá que configurar el servicio Rsyslog para establecer la conexión entre Kafka y dichos productos. Sin entrar mucho en detalle, una de las cosas que tendremos que hacer es definir reglas iptables, las cuales permiten la entrada de paquetes a través del puerto 514.

```
#Si queremos recibir datos mediante el protocolo TCP
-A INPUT -m state --state NEW,ESTABLISHED,RELATED -m tcp -p
tcp -- dport 514 -j ACCEPT

#Si queremos recibir datos mediante el protocolo UDP
-A INPUT -m udp -p udp --dport 514 -j ACCEPT
```

Lo siguiente, será configurar el servidor para poder recolectar logs usando UDP y TCP (TLS también pero no se va a mencionar en este trabajo). En el caso de **UDP** tendremos que añadir lo siguiente en el fichero **/etc/rsyslog.d/01-server.conf:**

```
module(load="imudp")
```

También tendríamos que añadir la siguiente sentencia para indicarle al servidor que queremos escuchar en el puerto 514:

```
input(type="imudp" port="514")
```

Para **TCP** es más de lo mismo pero cargando otro módulo diferente.

```
module(load="imtcp")
input(type="imtcp" port="514")
```

Paso previo a enviar un mensaje Syslog a un topic de Kafka, debemos definir una plantilla de salida el cual establecerá la estructura que el mensaje tendrá en Kafka. Estas plantillas las definiremos en el fichero **/etc/rsyslog.d/99-parse_rfc5424.conf:**

```
template(name="openvpn-vault" type="list") {
           constant(value="{")
           constant(value="\"raw_message\":\"")
           property(name="rawmsg" format="json")
           constant(value="\",\"message\":\"")
           property(name="msg" format="json")
           constant(value="\",\"pri\":\"")
           property(name="pri" format="json")
           constant(value="\",\"pri_text\":\"")
           property(name="pri-text" format="json")
           constant(value="\",\"syslogfacility\":\"")
           property(name="syslogfacility" format="json")
           constant(value="\",\"syslogfacility_text\":\"")
           property(name="syslogfacility-text" format="json")
           constant(value="\",\"syslogseverity\":\"")
           property(name="syslogseverity" format="json")
           constant(value="\",\"syslogseverity_text\":\"")
           property(name="syslogseverity-text" format="json")
           constant(value="\",\"protocol_version\":\"")
           property(name="protocol-version" format="json")
```

```
            constant(value="\",\"timestamp\":\"")
            property(name="timegenerated" dateFormat="unixtimestamp")
            constant(value="\",\"hostname\":\"")
            property(name="hostname" format="json")
            constant(value="\",\"fromhost_ip\":\"")
            property(name="fromhost-ip" format="json")
            constant(value="\",\"app_name\":\"")
            property(name="app-name" format="json")
            constant(value="\",\"procid\":\"")
            property(name="procid" format="json")
            constant(value="\",\"msgid\":\"")
            property(name="msgid" format="json")
            constant(value="\",\"structured_data\":\"")
            property(name="structured-data" format="json")
            constant(value="\",\"tag\":\"")
            property(name="$!rfc5424-sd!rbvault@39483!tag" format="json")
            constant(value="\",\"hash\":\"")
            property(name="$!rfc5424-sd!hash@39483!hash" format="json")
            constant(value="\",\"inputname\":\"")
            property(name="inputname" format="json")
            constant(value="\",\"sensor_ip\":\"")
            constant(value="192.168.10.230")
            constant(value="\",\"sensor_name\":\"")
            constant(value="openvpn-vault")
            constant(value="\",\"sensor_uuid\":\"")
            constant(value="f54e7128-e797-4620-abc4-f99a978cd851")
            constant(value="\"}")
        }

if $fromhost-ip == '192.168.10.240' then {
  #Action to send logs to Apache Kafka
  action(type="omkafka" topic="rb_vault" broker="kafka.redborder.cluster:9092"
template="openvpn-vault")
  stop
}
```

Primero definimos los diferentes campos que contendrá el mensaje, y después le indicamos a donde enviar dichos logs, en este caso, al topic rb_vault, usando la plantilla definida anteriormente llamada openvpn-vault.
Una vez configurado, podemos comprobar que efectivamente podemos recibir syslog, haciendo uso de la herramienta tcpdump y netcat. Con netcat enviaremos un mensaje de prueba, y con tcpdump escucharemos en el puerto 514 para ver si llega algún mensaje a nuestro servidor:

```
    tcpdump port 514 -i bond0
```

y desde cualquier otra máquina, ejecutamos el comando netcat para enviar un mensaje de prueba:

```
echo "message" | nc 172.18.10.29 514
```

Efectivamente, podemos ver que nos llegan mensajes de la dirección 192.168.10.197, la cual es la dirección IP desde la que ejecuté el comando netcat.

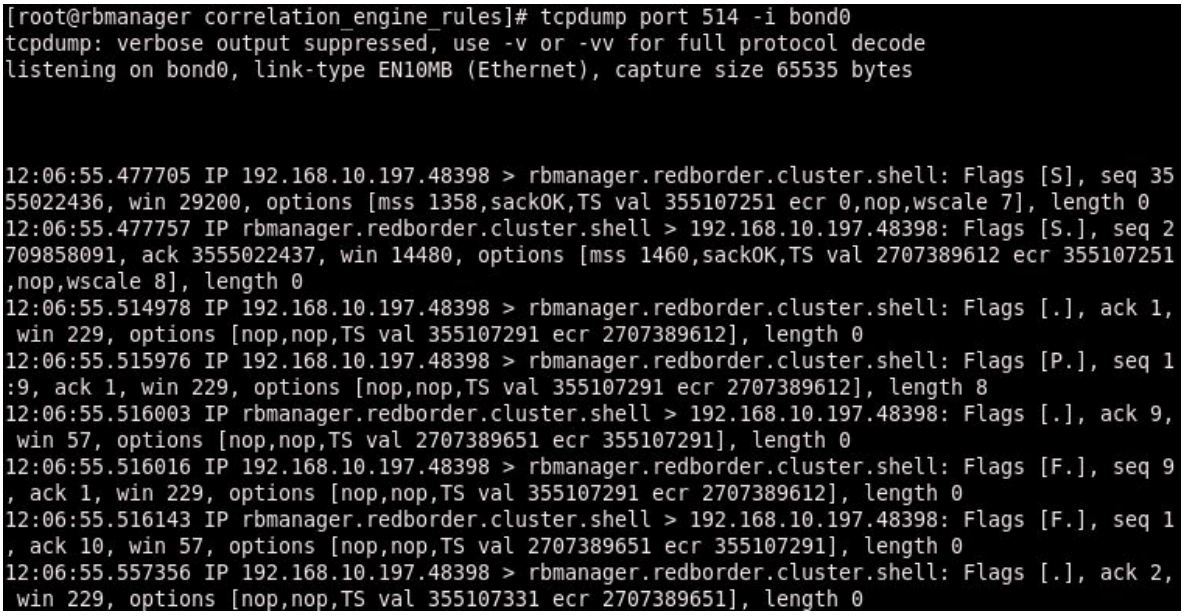


Figura 3-5. Escuchando el puerto 514.

Pero todo esto lo hemos hecho con el objetivo de recibir logs de los productos en nuestro topic de kafka, luego si hemos configurado todo bien, al conectarnos a la openvpn por ejemplo, podemos ver que recibimos mensajes:

Figura 3-6. Consumiendo de rb_vault.

Capítulo 4. Normalización, enriquecimiento y visualización de los datos.

Una vez dispongamos de nuestro servicio syslog bien configurado y funcionando junto a los topics de apache Kafka, pasamos a la siguiente fase, en la cual vamos a tratar los datos recibidos anteriormente. De momento, los logs que tenemos en el topic rb_vault, son mensajes en crudo parseados directamente de los mensajes syslog que pudimos ver anteriormente con tcpdump. Estos mensajes no nos son útiles, ya que a veces no tienen información que nos interesa, o si bien la que nos interesa no se puede acceder a ella de manera sencilla. Es aquí donde entra en juego el servicio clave para esta fase: Logstash.

4.1 Tratamiento de datos con logstash

Logstash es una herramienta para la recolección, el procesamiento y el envío de mensajes de registros y eventos. La recolección se lleva a cabo con una serie de plugins de entrada configurables. Una vez que los datos se han recogido con dichos plugins, ahora se pueden procesar con un gran número de filtros que modifican los datos del evento. Finalmente, logstash redirecciona estos datos a plugins de salida, los cuales se encargan de mandar estos eventos ya modificados a otra fuente de datos, en nuestro caso será a otro topic de Kafka.[4]

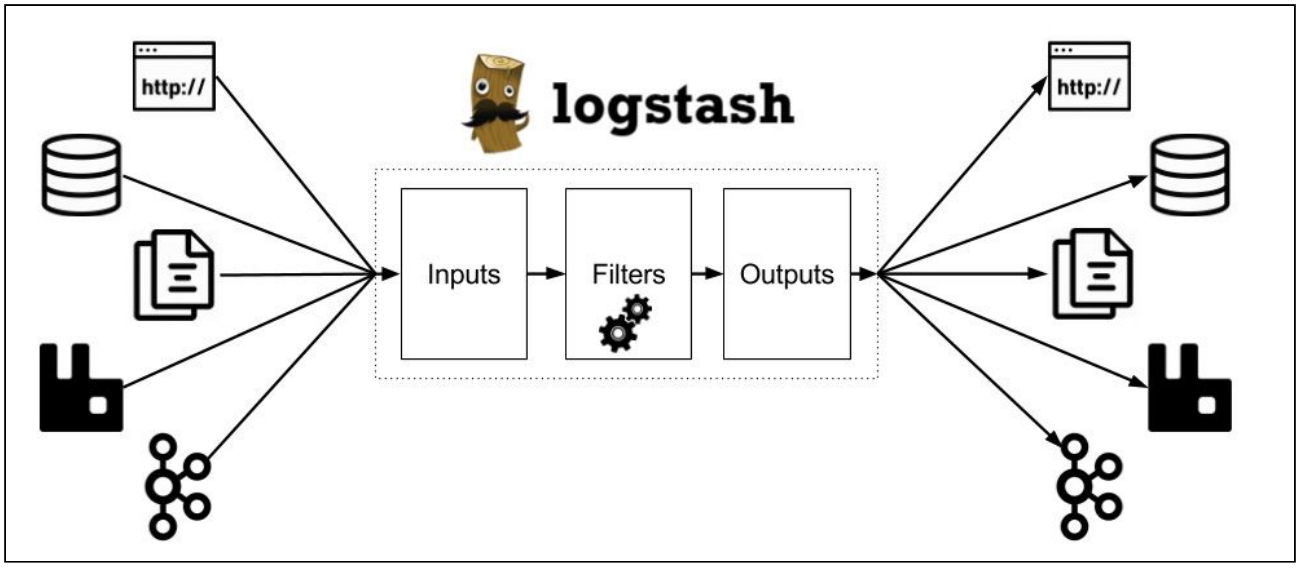


Figura 4-1. Esquema funcionamiento de Logstash.

4.1.1 Entradas

Logstash usa las entradas para recibir los datos que deseamos de diferentes fuentes de datos. en nuestro caso, vamos a leer los mensajes de un topic de Kafka, pero podríamos leer la salida de una API HTTP, de Twitter, de un fichero de Google Clouds...etc. Logstash tiene una gran cantidad de plugins, luego las posibilidades son bastante grandes. Como hemos dicho, nosotros vamos a leer datos de un topic de Kafka, luego usaremos el plugin "**kafka**". Para ello, como estamos trabajando con Syslog, del cual se encarga el módulo Vault, vamos a crear los ficheros necesarios dentro de **/etc/logstash/pipelines/vault/**. Para este primer paso, crearemos el fichero de entrada, **00_input.conf**.

```
input {
  kafka {
    codec => json
    bootstrap_servers => "manager.redborder.cluster:9092"
    topics => ["rb_vault"]
  }
}
```

- **Codec.** El codificador usado para los datos de entrada. Este se encarga de decodificar los datos antes de que llegue a la entrada, de manera que no hace falta que creemos un filtro aparte en el pipeline.

- **Bootstrap Servers.** Con esta opción, establecemos una lista de instancias de Kafka para la conexión inicial con el cluster. Estas urls se utilizan para descubrir más adelante todos los miembros del cluster, luego no es necesario que contenga el conjunto entero de servidores (aunque es recomendable, en caso de que un servidor esté caído).
- **Topics.** Topics de Kafka de dónde queremos leer nuestros datos. En nuestro caso, es rb_vault, donde tenemos los mensajes obtenidos mediante Rsyslog.

4.1.2 Filtros

La potencia de Logstash reside en sus filtros. Con ellos, podemos modificar los datos a nuestro gusto, usando una gran cantidad de diferentes plugins y los cuales a su vez nos ofrecen diferentes opciones. Nosotros vamos a usar en mayor proporción el filtro **grok** y el **mutate**.

- **Grok.** Este plugin se encarga de parsear y estructurar texto arbitrario. Grok es actualmente una de las mejores formas en dar estructura a registros de datos que no disponen de una. Grok dispone de 120 patrones ya construidos dentro del plugin que nos harán la vida mucho más sencilla a la hora de parsear los datos.
- **Mutate.** Realiza transformaciones generales con los campos de los eventos. Podemos renombrar, borrar, reemplazar y modificar campos a nuestro gusto.

Al igual que con la entrada, vamos a crear un fichero donde vamos a realizar operaciones con estos filtros. En este caso, el fichero se llamará **43_openvpn.conf**. El primer número podría ser cualquier otro, solo hay que asegurarse de que no sea ni 0 ni 99 ya que estos pertenecen a los ficheros de entrada y salida respectivamente.

Para ilustrar el funcionamiento de estos filtros, voy a coger uno de los mensajes que recibimos en rb_vault al conectarnos a través de openvpn y vamos a aplicar los filtros a dicho log:

```
{"raw_message":"<14>Jun  5 08:53:21 rbovpn openvpnas: [-] OVPN 0 OUT:
'Fri Jun  5 08:53:21 2020 openvpn\/31.4.210.217:39005 MULTI: Learn:
172.33.33.58 -> openvpn\/31.4.210.217:39005'","message":" [-] OVPN 0
OUT: 'Fri Jun  5 08:53:21 2020 openvpn\/31.4.210.217:39005 MULTI:
Learn: 172.33.33.58 ->
```

```
openvpn\/31.4.210.217:39005'","pri":"14","pri_text":"user.info","syslog
facility":"1","syslogfacility_text":"user","syslogseverity":"6","syslog
severity_text":"info","protocol_version":"0","timestamp":"1591347290","
hostname":"rbovpn","fromhost_ip":"192.168.10.230","app_name":"openvpnas
","procid":"-","msgid":"-","structured_data":"-","tag":"","hash":"","in
putname":"imtcp","sensor_ip":"192.168.10.230","sensor_name":"openvpn-va
ult","sensor_uuid":"f54e7128-e797-4620-abc4-f99a978cd851"}
```


El campo que nos interesa es “message”. Ahí es donde está la información que openvpn está enviando a modo de registro de un evento, en este caso conexión al cliente. Como dijimos, vamos a crear nuestro fichero, el cual va a tener la siguiente estructura:

```
#Version: 1.1
filter {
  if "openvpnas" in [app_name] {
      ….
      Aquí irán los diferentes plugins (grok y mutate)
      ….
  }
}
```

Lo primero será comprobar el campo app_name, en donde normalmente podemos diferenciar un producto de otro, de tal manera que sólo procesamos los eventos que provengan del openvpn.
Como segundo paso, habría que eliminar caracteres raros que no nos interesen, y para ello utilizaremos el plugin mencionado anteriormente, mutate.

```
mutate {
      gsub => [
        "message", "\"", "",
        "message", "/", " ",
        "message", "[\\]", " ",
        "message", "u'", "",
        "message", "'", "",
        "message", "  ", " "
      ]
}
```

Usamos la opción gsub para hacer un reemplazo global dentro del campo message. En el mensaje que hemos puesto como ejemplo, solo nos interesa borrar las barras invertidas escapadas, pero en otros mensajes nos interesa borrar unas comillas simples, o espacios de más.

Una vez normalizado el mensaje, vamos a enriquecerlo. Para ello, vamos a hacer uso de los patrones del filtro grok. Habrá que crear un fichero llamado openvpn dentro de la carpeta **/etc/logstash/pipelines/vault/patterns/.** En este fichero vamos a crear diferentes patrones para detectar los diferentes tipos de mensajes que podemos recibir, y así poder sacar información útil que después vamos a utilizar para la visualización de datos en la web.

La sintaxis de un patrón de grok es %{SYNTAX:SEMANTIC}. **SYNTAX** es el nombre del patrón que casará con el texto. La mayoría de los patrones que vamos a utilizar ya están definidos, aunque podemos crear patrones propios, como veremos ahora. Por ejemplo, NUMBER casaría con 3.44, y 192.168.10.1 casaría con el patrón IP. **SEMANTIC** es el nombre que le vamos a dar al texto que hemos casado anteriormente con el patrón. Por ejemplo, 3.44 podría ser **duración_evento**, y 192.168.10.1 podría ser **ip_cliente.**

Siguiendo el ejemplo del mensaje de conexión de openvpn, tenemos el siguiente mensaje:

```
 [-] OVPN 0 OUT: 'Fri Jun  5 08:53:21 2020 openvpn\/31.4.210.217:39005
MULTI: Learn: 172.33.33.58 -> openvpn\/31.4.210.217:39005'
```

Al cual le correspondería el siguiente patrón:

```
OPENVPN_P11             OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{GREEDYDATA:openvpn_msg} %{IP:openvpn_lan_ip} -> %{WORD:openvpn_user}
%{IP:remove}:%{NUMBER:remove}
```

Una vez definamos los patrones para todos los tipos de mensajes que podemos recibir de openvpn, ya podemos enriquecer los logs usando grok:

```
grok {
      patterns_dir =>
["/etc/logstash/pipelines/vault/patterns/openvpn"]
      match => [
```

```
        "message","%{OPENVPN_P1}",
        "message","%{OPENVPN_P2}",
        "message","%{OPENVPN_P3}",
        "message","%{OPENVPN_P4}",
                ..............
       "message","%{OPENVPN_P38}"
}
```

Como podemos ver hemos tenido que crear un gran número de patrones, 38 exactamente, ya que son el número de logs diferentes que podemos recibir. El orden de los patrones es importante, ya que si un mensaje casa con el primer patrón, ya no seguirá comparando con el resto, por lo que hay que asegurarse que no haya patrones muy generales en donde varios mensajes diferentes puedan hacer match. Cuanto más específico, mejor.

4.1.3 Salidas

Las salidas son la fase final de logstash. Una vez hemos transformado los datos a nuestro gusto, es hora de mandarlos de una fuente de datos, la cual es en nuestro caso, de nuevo, un topic de Kafka, rb_vault_post. Luego, en el fichero 99_output.conf:

```
output {
  kafka {
    codec => json
    topic_id => "rb_vault_post"
    bootstrap_servers => "rbmanager.redborder.cluster:9092"
  }
}
```

Si todo ha salido bien, al consumir del topic rb_vault_post, deberíamos ver el mensaje original de nuestro ejemplo con nuevos campos creados, que son los que definimos en los patrones.

Mensaje original en rb_vault:

```
{"raw_message":"<14>Jun  5 08:53:21 rbovpn openvpnas: [-] OVPN 0 OUT:
'Fri Jun  5 08:53:21 2020 openvpn\/31.4.210.217:39005 MULTI: Learn:
172.33.33.58 -> openvpn\/31.4.210.217:39005'","message":" [-] OVPN 0
OUT: 'Fri Jun  5 08:53:21 2020 openvpn\/31.4.210.217:39005 MULTI:
```

```
Learn: 172.33.33.58 ->
openvpn\/31.4.210.217:39005'","pri":"14","pri_text":"user.info","syslog
facility":"1","syslogfacility_text":"user","syslogseverity":"6","syslog
severity_text":"info","protocol_version":"0","timestamp":"1591347290","
hostname":"rbovpn","fromhost_ip":"192.168.10.230","app_name":"openvpnas
","procid":"-","msgid":"-","structured_data":"-","tag":"","hash":"","in
putname":"imtcp","sensor_ip":"192.168.10.230","sensor_name":"openvpn-va
ult","sensor_uuid":"f54e7128-e797-4620-abc4-f99a978cd851"}
```


Mensaje en rb_vault_post después de pasar por el pipeline de logstash:


```
{"source":"","structured_data":"-","app_name":"openvpnas","pri_text":"u
ser.info","openvpn_port":"38971","procid":"-","openvpn_date":"Fri Jun 5
09:56:37
2020","@timestamp":"2020-06-05T09:58:08.180Z","raw-message":"<14>Jun  5
09:56:37 rbovpn openvpnas: [-] OVPN 0 OUT: 'Fri Jun  5 09:56:37 2020
openvpn/31.4.210.217:38971 MULTI: Learn: 172.33.33.59 ->
openvpn/31.4.210.217:38971'","msgid":"-","syslogseverity":"6","openvpn_
wan_ip":"31.4.210.217","tag":"","openvpn_msg":"MULTI:
Learn:","fingerprint":"4064a221272f27382254bdbabc5267fba714797d8017bca0
7a90a82e4f2bf7f8","message":" [-] OVPN 0 OUT: Fri Jun 5 09:56:37 2020
openvpn 31.4.210.217:38971 MULTI: Learn: 172.33.33.59 -> openvpn
31.4.210.217:38971","action":"","target":"","openvpn_lan_ip":"172.33.33
.59","openvpn_user":["openvpn","openvpn"],"syslogfacility_text":"user",
"pri":"14","timestamp":"1591351087","syslogseverity_text":"info","syslo
gfacility":"1","fromhost_ip":"192.168.10.230","sensor_ip":"192.168.10.2
30","hostname":"rbovpn","sensor_uuid":"f54e7128-e797-4620-abc4-f99a978c
d851","status":"","inputname":"imtcp","remove":["31.4.210.217","38971"]
,"sensor_name":"openvpn-vault","protocol_version":"0","@version":"1"}
```


4.2 Visualización de los datos

El siguiente paso en nuestro pipeline del tratamiento de los datos, consistirá en la visualización de la información recogida hasta el momento. Los datos de los que disponemos no nos son útiles ya que no podemos establecer relaciones entre ellos que nos muestren información útil de la que un administrador de sistemas pueda extraer conocimiento sobre lo que está pasando en el sistema. Es por ello, que necesitamos visualizar los mensajes en nuestra plataforma web de redBorder.

Para conseguir este objetivo, primero tendremos que estudiar el papel del servicio de conexión entre Ruby on Rails y nuestros mensajes en el topic de Kafka rb_vault_post. Este servicio es Apache Druid.

4.2.1 Apache Druid

Apache Druid es una base de datos de alto rendimiento en tiempo real, está orientada a columnas y se trata de un sistema distribuido que se compone de varios servicios. Proporciona consultas analíticas rápidas sobre datos de eventos. Druid indexa los datos para crear vistas y almacena los datos en un formato columnar optimizado para agregaciones y filtros. Sus principales características son:

- **Almacenamiento orientado por columnas.** Almacena y comprime cada columna individualmente, de manera que sólo necesita leer las que se necesiten para una query en particular, produciendo escaneos rápidos, rankings y agrupamientos.
- **Streaming e ingesta por lotes.** Druid permite a sus usuarios utilizar los principios de la Arquitectura Lambda. Esta es una arquitectura de procesamiento de datos creada para el manejo masivo de datos mediante métodos de procesamiento por lotes y streaming.
- **Particionamiento optimizado por tiempo.** Apache Druid particiona de manera inteligente los datos según el tiempo, permitiendo tiempos de respuesta rápidas. Además, Druid permite lo que se conoce como Roll-Up. Esto es, la pre-agregación de datos según son procesados, con la cual podemos ahorrar bastante espacio. Roll-up es una agregación de primer nivel sobre una serie de columnas que reduce el tamaño de los datos. Durante el Roll-Up, las columnas seleccionadas se agrupan según el timestamp y las dimensiones. Cuando se activa el roll-up, las filas que tengan dimensiones y timestamp idénticos se colapsarán y se convertirán en una sola fila.
- **Columnas Flexibles y soporte SQL.** Druid puede manejar esquemas cambiantes y datos anidados. Y además de su lenguaje nativo JSON, Druid también puede comunicarse mediante SQL sobre HTTP o JDBC(Java Database Connectivity).

Druid tiene una arquitectura distribuida, multi-procesos que fue diseñada para ser compatible con la nube y fácil de manejar. Cada proceso de Druid puede ser configurado y escalado independientemente, dándote una gran

flexibilidad sobre tu cluster. Esto también proporciona tolerancia a fallos, ya que si un componente cae, no afectaría inmediatamente al resto.

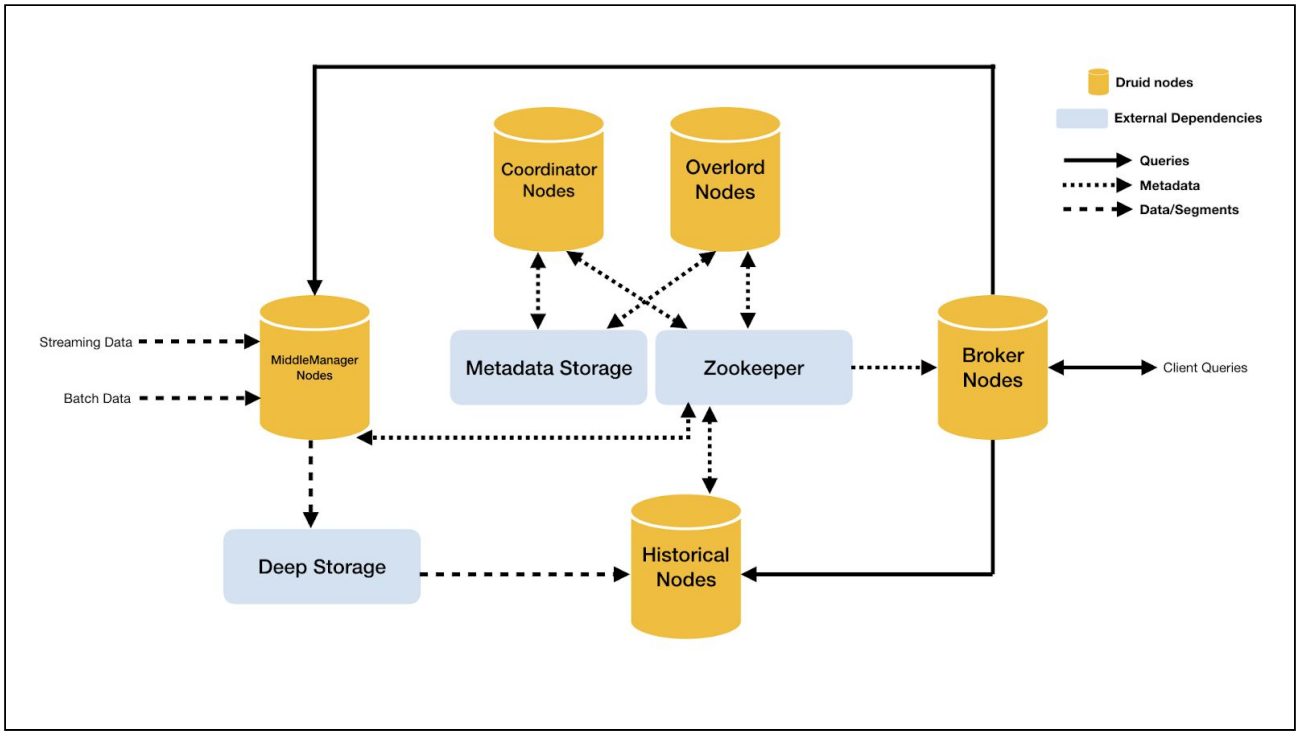


Figura 4-2. Esquema Druid.

- **Coordinator.** Es el responsable de manejar la disponibilidad de datos en el cluster. Este proceso se comunica con el proceso Historical para cargar o desestimar segmentos basados en una configuración. Mantiene una conexión con el cluster Zookeeper para obtener información actual del cluster. También mantiene una conexión a una base de datos que contiene información sobre segmentos disponibles y reglas.
- **Overlord.** Se encarga de asignar la carga de trabajo de la ingesta de datos. Es responsable de aceptar tareas, coordinarlas mediante su distribución, crear bloqueos sobre ciertas tareas y devolver estados a los que hicieron las llamadas.
- **MiddleManager.** Este proceso es un worker que se encarga de ejecutar las tareas que han sido enviadas. Este a su vez manda estas tareas a los Peones, los cuales están corriendo en JVM (Java Virtual Machine) separadas, con motivo de aislar los recursos y los logs. Un peón sólo puede ejecutar una tarea a la vez, sin embargo un MiddleManager puede disponer de varios peones. [5]

4.2.1.1 Druid Ingestion

En Druid, hay varios métodos de ingesta de datos. En nuestro caso, vamos a utilizar el de Kafka, el cual se encarga de leer directamente de un topic de Kafka. Pero independientemente del método que elijamos, los datos se cargan en Druid haciendo uso de una tarea que se ejecuta solo una vez, o de supervisores. En cualquier caso, parte de la definición de esa tarea o supervisor se encuentra en un fichero spec. Estos ficheros constan de 3 partes:

- **dataSchema**, en donde se configura el nombre de la fuente de datos, el timestamp, dimensiones, métricas y las transformaciones y filtros necesarios.
- **ioConfig,** el cual le dice a Druid cómo conectarse al sistema fuente y como parsear los datos.
- **tuningConfig**, el cual controla varios parámetros específicos a cada método de ingesta de datos.

En nuestro caso, para la ingesta de datos de rb_vault_post, este sería nuestro supervisor, el cual se llama rb_realtime.spec.

```
{
    "dataSchema" : {
      "dataSource" : "rb_vault",
      "parser" : {
        "type" : "string",
        "parseSpec" : {
          "type" : "jsonLowercase",
          "format" : "json",
          "timestampSpec" : {
            "column" : "timestamp",
            "format" : "ruby"
          },
          "dimensionsSpec" : {
            "dimensions": ["pri", "pri_text",...etc]
          }
        }
      },
      "metricsSpec" : [
                                    {"type":"count"   , "name":"events"}
```

```
                    ],
    "granularitySpec" : {
      "type" : "uniform",
      "segmentGranularity" : "HOUR",
      "queryGranularity" : "MINUTE"
    }
  },
  "ioConfig" : {
    "type" : "realtime",
    "firehose": {
      "type": "kafka-0.8",
      "consumerProps": {
        "zookeeper.connect": "127.0.0.1:2181",
        "zookeeper.connection.timeout.ms" : "15000",
        "zookeeper.session.timeout.ms" : "15000",
        "zookeeper.sync.time.ms" : "5000",
        "rebalance.max.retries": "4",
        "group.id": "rb-group",
        "fetch.message.max.bytes" : "1048576",
        "auto.offset.reset": "largest",
        "auto.commit.enable": "true"
      },
      "feed" : "rb_vault_post"
    },
    "plumber": {
      "type": "realtime"
    }
  },
  "tuningConfig": {
    "type" : "realtime",
    "maxRowsInMemory": 120000,
    "intermediatePersistPeriod": "PT20m",
    "windowPeriod": "PT30m",
    "basePersistDirectory": "/tmp/realtime",
    "shardSpec": {"type": "linear", "partitionNum": 0 },
    "rejectionPolicy": {
      "type": "serverTime"
    }
  }
```

En las primeras líneas definimos el nombre, y el tipo de los datos que vamos a recibir. En dimensionsSpec, estamos definiendo las columnas que vamos a tener en nuestra base de datos. Normalmente se define el nombre de la columna y su tipo, pero si no se especifica coge String por defecto. Después definimos las métricas, que son agrupaciones que Druid hará si Roll-Up está activado. En este caso no tenemos ninguna métrica excepto contar el número de eventos.

Lo siguiente es definir la configuración de la fuente de entrada. Primero definimos el modo a utilizar, el cual es realtime. Después establecemos los parámetros de configuración para conectarse a Apache Kafka mediante el parámetro firehose. También habrá que indicarle el topic de donde queremos leer los mensajes mediante “feed”.

Por último, establecemos las propiedades que deseamos para nuestro método de ingestión, como máximo número de mensajes a almacenar, el tamaño máximo de cada uno, el período de tiempo para incluir mensajes… etc.[6]

4.2.1.2 Druid Broker

El broker de Druid es el responsable de manejar las queries externas provenientes del cliente. Es el proceso que redirige las queries cuando ejecutamos Druid en un cluster distribuido. Entiende los metadatos publicados en el Zookeeper sobre que procesos donde se encuentran los segmentos y envía las queries a los procesos que contienen dicha información. También se encarga de unir todos los conjuntos de resultados de cada uno de los procesos.

4.2.1.3 Ejecutando una query

Para ilustrar mejor el funcionamiento de Druid, vamos a seguir el pipeline que este sigue cuando recibe una query desde la web. La query contendrá información sobre el intervalo (período de tiempo), las dimensiones y las métricas requeridas.

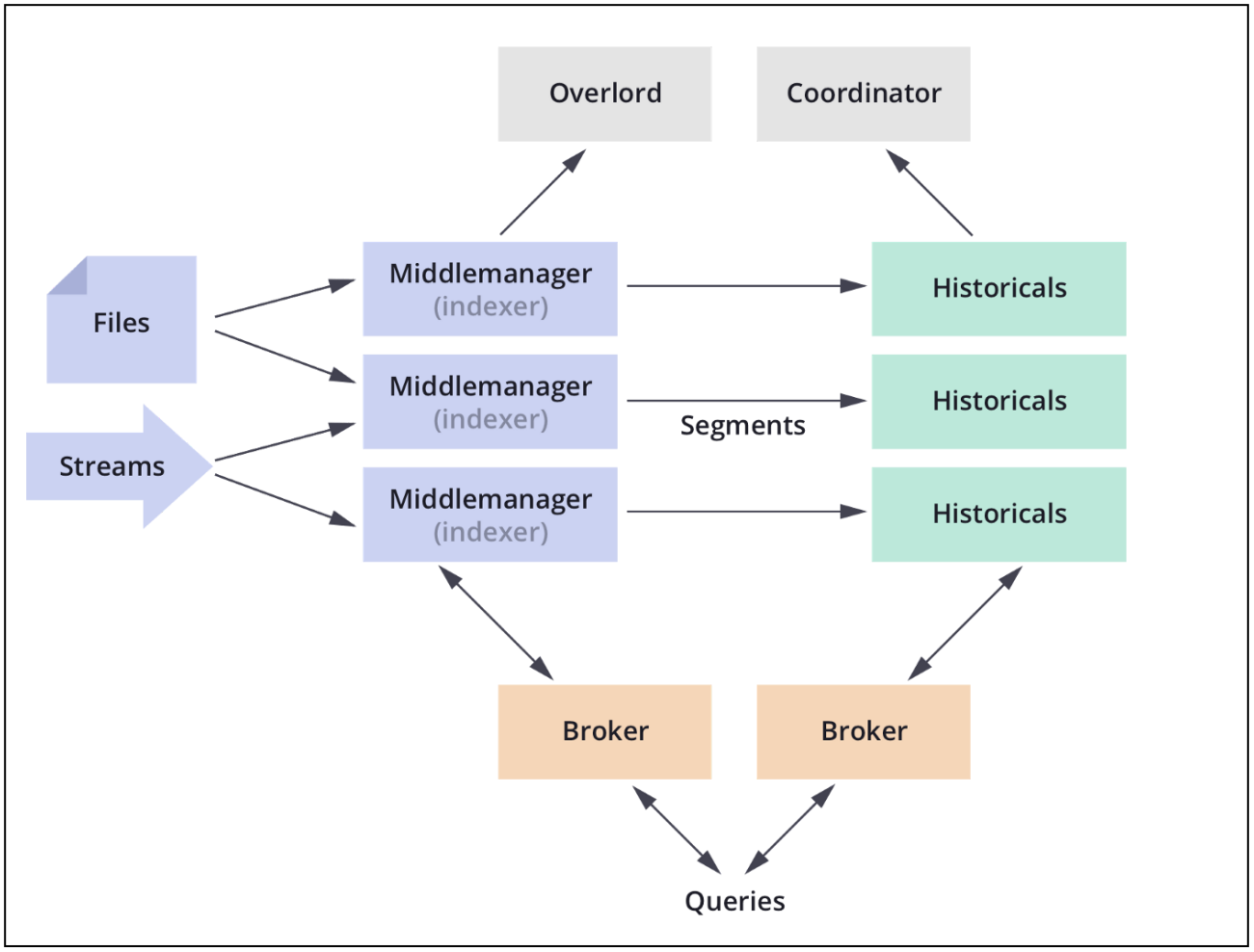


Figura 4-3. Ejecución de una query.

1. La query llega al broker desde nuestro proyecto web desarrollado en Ruby on Rails. El broker ya conoce dónde se encuentran los segmentos para ese período de tiempo. (Por ejemplo, 2 segmentos se necesitan del nodo historical, otro del nodo B, y además también necesita información del actual segmento indexado publicado en MiddleManager A.
2. La query se envía a todos los nodos necesarios (Historical A, B y MiddleManager A).
3. Cada uno de esos nodos realizará la agregación necesaria, unirá los datos de acuerdo a la query y enviará los resultados de vuelta al broker.
4. Los datos se unen en el broker, dependiendo de la query, y se devuelven al cliente.

4.2.2 Creación de dimensiones con scripts.

Como vimos anteriormente, con Logstash nos encargamos de tratar los datos que recibimos en rb_vault, para enriquecerlos y mandarlos al topic rb_vault_post, donde el mensaje original añade nuevos campos, como openvpn_lan_ip por ejemplo. Todos estos campos tienen que ser accesibles desde nuestra Web, de tal manera que podamos seleccionarlos y visualizarlos. A veces, el número de nuevas dimensiones puede ser bastante grande, por lo que vamos a crear unos scripts, los cuales al ser ejecutados se encarguen de crear las tabs en la web, y además de crear los ficheros necesarios para el parseo de mensajes (filtros y patrones).

Mientras que para los filtros y patrones el script simplemente se encargará de crear los ficheros vistos en el capítulo anterior de Logstash (43_openvpn.conf y el fichero openvpn donde se encontraban los patrones), lo importante del script es que las diferentes tabs se creen correctamente en el fichero dimensions.yml, para su posterior visualización en el manager. Por motivos de espacio, se colocará el código desarrollado para OpenVPN en el apéndice A.1 de esta memoria. El resto de scripts para los diferentes productos sería exactamente igual solo que cambiando las columnas y los filtros.

4.2.3 Creación de Dashboards y Widgets.

Una vez dispongamos en el manager de las tabs creadas a partir del script anterior, es el momento de visualizar todos estos datos mediante la creación de dashboards y widgets. El dashboard es simplemente un tablero donde vamos a crear los widgets, los cuales son los elementos que nos van a mostrar de diferentes formas (gráficas, tablas, mapas de calor...etc) los datos que hemos recogido hasta ahora.

Lo primero que haremos será crear un dashboard para cada producto, en donde vamos a visualizar los datos de cada uno de ellos. Para ello, hacemos click en “Add Dashboard”.

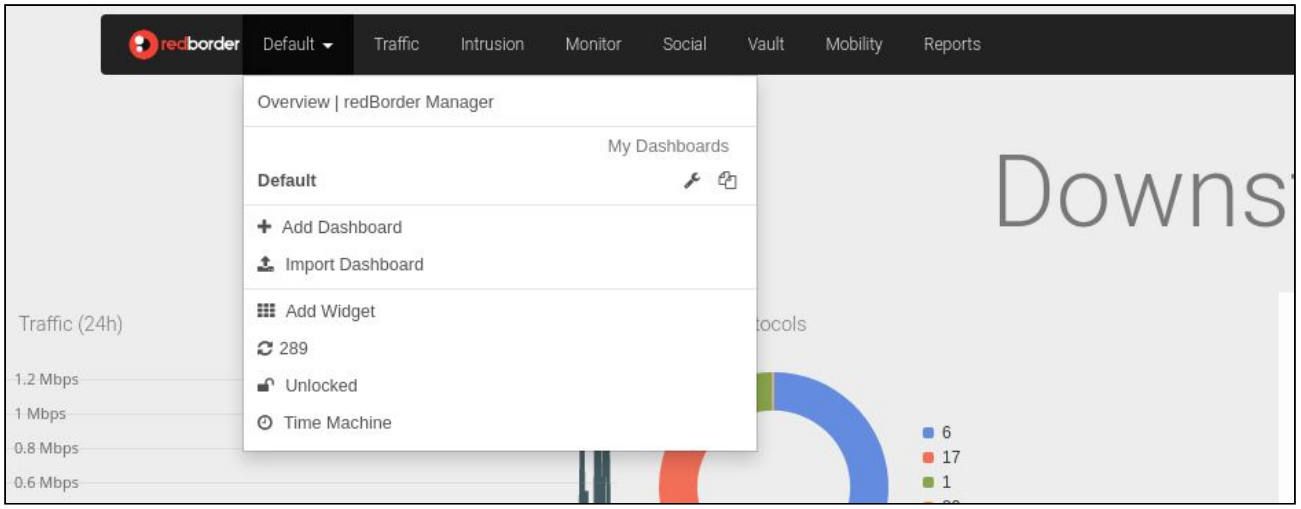


Figura 4-4. Creando un Dashboard.

Una vez creado el Dashboard, vamos a añadir los Widgets, que son nuestra herramienta para visualizar la información de diferentes formas gráficas. Si pulsamos en Add Widget, veremos los diferentes módulos presentes en nuestro proyecto (Traffic, Vault, Intrusion...etc) en donde cada uno de ellos dispone de sus propias formas de representación gráficas.

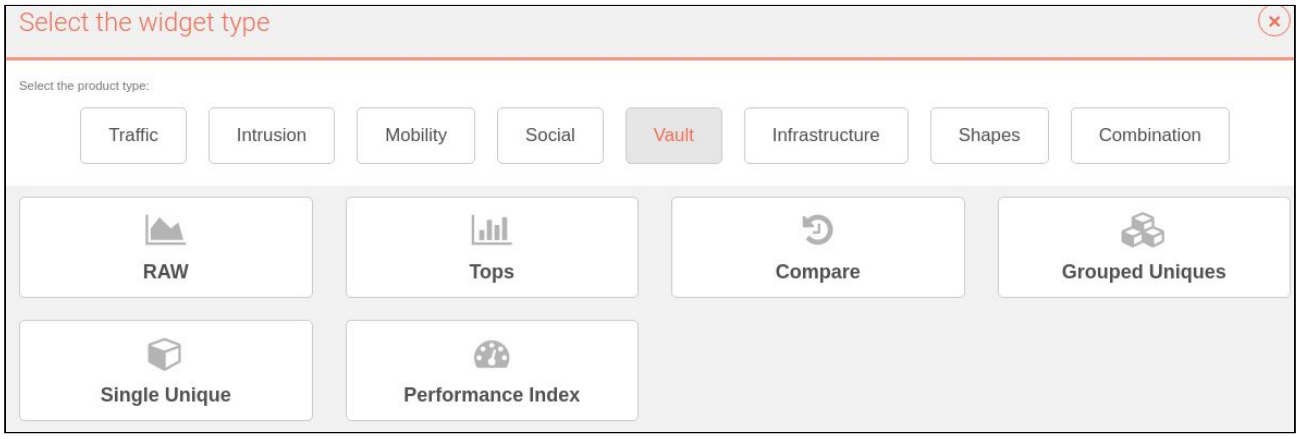


Figura 4-5. Tipos de widgets.

Dentro de cada tipo de widget, tenemos diferentes formas de visualización. Por ejemplo, si pulsamos en RAW, podemos ver:

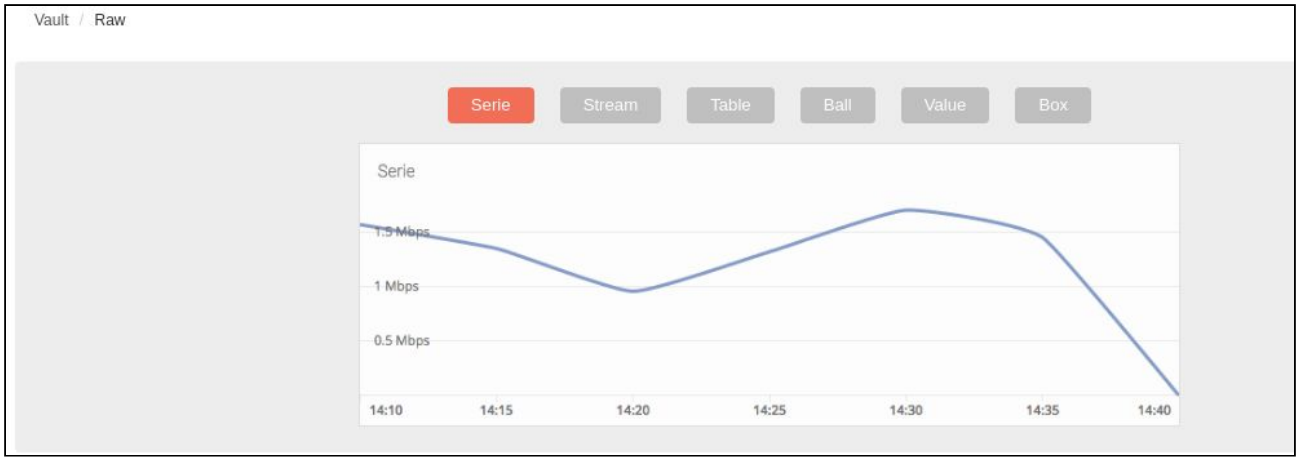


Figura 4-6. Más tipos de widgets

Puesto que todos los widgets ya están implementados, este apartado se centra en el estudio de los diferentes tipos y como se encargan de representar la información, para así poder elegir el widget adecuado a la hora de crear los dashboards. Aparte de esto, también habrá que plantear los diferentes casos de uso para la representación de la información, ya que no cualquier widget vale para los datos. Por ejemplo, para representar el top de usuarios que se han conectado a la vpn, el widget adecuado sería una tabla, mientras que para representar el número de conexiones efectuadas, el widget correcto sería la bola. A continuación, se presentan los dashboards creado para OpenVPN, BitDefender Gravity Zone, Untangle e InsightVM. Por motivos de espacio, se colocarán las capturas de dichos dashboards en el anexo.

4.2.3.1 Juniper

Como Juniper es una VPN, los widgets que hemos creado están relacionados con las conexiones que se realizan a dicha vpn, para controlar quién accede y desde donde. Los Dashboards creados se encuentran en el **Apéndice B.1.**

4.2.3.2 OpenVPN

De nuevo, vamos a crear widgets que muestran información útil sobre las conexiones llevadas a cabo. Más adelante veremos como con la creación de reglas de correlación, podremos crear más widgets para mostrar información extra. **Apéndice B.2**

4.2.3.3. Untangle

Nos encontramos otra vez con otro producto el cual ofrece conexiones vpn y del cual mostraremos información relacionada sobre usuarios y conexiones. **Apéndice B.3.**

4.2.3.4. BitDefender GravityZone

GravityZone proporciona una visión completa de la seguridad de una organización, amenazas globales y control sobre la servicios de seguridad que se encargan de proteger equipos virtuales o físicos, servidores y dispositivos móviles. Por lo tanto, los dashboard que vamos a crear mostrarán información sobre la seguridad de end-points en nuestra organización, como dispositivos comprometidos o ficheros sospechosos. **Apéndice B.4.**

4.2.3.5. Rapid7 InsightVM

InsightVM se encarga de recolectar datos de vulnerabilidad , convertirlos en respuestas y minimizar el riesgo. Por lo tanto, de nuevo, los dashboard estarán relacionados con vulnerabilidades. **Apéndice B.5.**

Capítulo 5. Correlation Engine Rules.

Ya disponemos de los datos exportados por los diferentes productos en nuestros dashboards, en los cuales hemos creado diferentes widgets para visualizar los datos de diferentes formas de tal manera que el administrador de sistemas pueda extraer información útil sobre lo que está sucediendo en el sistema de manera rápida y eficaz. Pero aún podemos ir un paso más allá y, utilizando dichos datos, extraer aún más información relacionándolos con otros tipos de datos provenientes de otros módulos (correlación).

Para ello, vamos a hacer uso de CEP (Complex Event Processing), que nos permitirá crear diferentes reglas haciendo uso del lenguaje Siddhi Streaming SQL. Una vez hayamos creado dichas reglas, las cuales se crearán para el producto OpenVPN, implementaremos un formulario en la web para que el usuario pueda crear reglas de manera más sencilla sin necesidad de conocer el lenguaje Siddhi. Dicho esto, vamos a explicar primero la herramienta que nos va a permitir conseguir todo esto: WSO2 CEP.

5.1 WSO2 CEP.

Cada transacción y actividad de una empresa consiste en una serie de eventos constantes. Aquellas empresas que monitorean dichos eventos en tiempo real y que responden de manera rápida antes ellos tienen más ventajas de negocio que sus competidores. El procesamiento de eventos complejos (Complex event processing) trata sobre la escucha de eventos y la detección de patrones en tiempo real, sin tener que almacenar dicha información.

WSO2 Complex Event Processor (CEP) es un servidor ligero, sencillo de utilizar y de código abierto bajo la licencia de apache Software License. CEP se encarga de identificar los eventos más relevantes, analizarlos y actuar sobre ellos en tiempo real. Se desarrolló para ser altamente escalable y con un rendimiento muy alto.

5.2 Arquitectura y funcionamiento

La arquitectura de WSO2 CEP consta de los siguientes componentes:

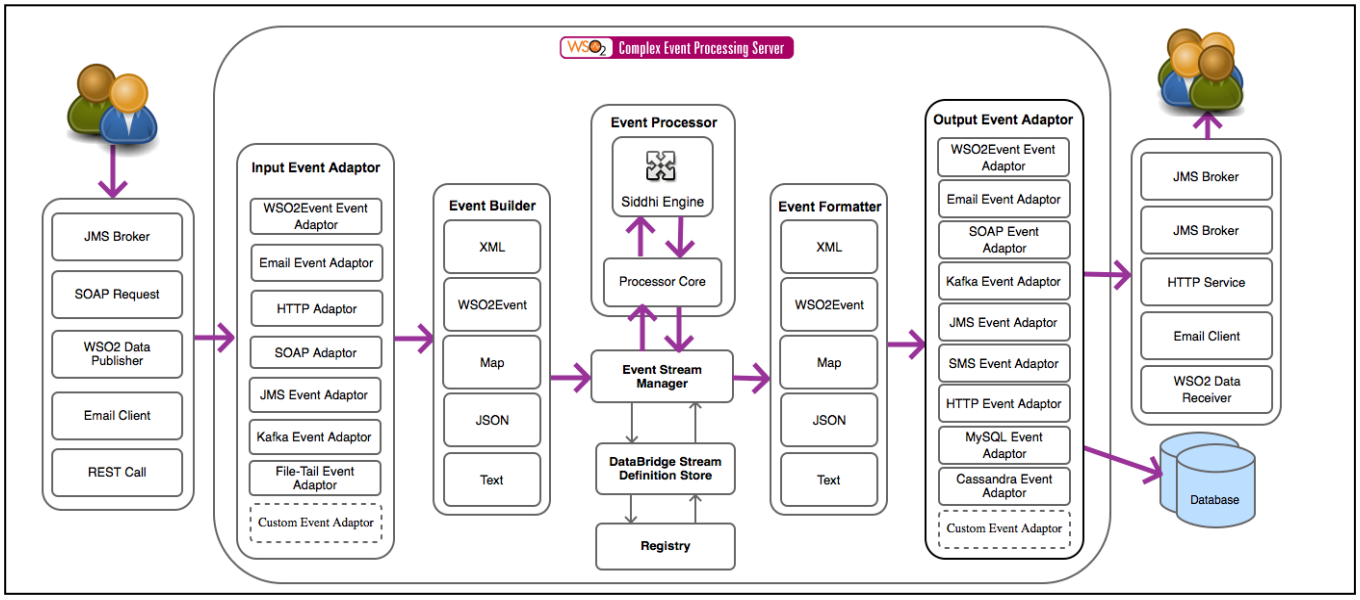


Figura 5-1. Esquema WSO2 CEP

.

- **Input Event Adapter.** Recibe eventos que van dirigidos al CEP. WSO2 CEP soporta de manera predeterminada los adaptadores más comunes (SOAP, email, HTTP, Kafka, File).

- **Event Builder.** Convierte los eventos recibidos por el Input Event Adapter (en formato XML, JSON, texto, mapa y WSO2Event) en eventos streams (stream es el formato con el que funciona el procesador de eventos).

- **Event Stream Manager.** Se encarga de administrar todos los streams de eventos en el sistema. Es el eje central para todos los eventos. Los componentes de Event Builder, Processor y Formatter interactúan con el Event Stream Manager para conseguir información acerca de sus streams.

- **Event Processor.** Soporta el procesamiento de eventos. Es la unidad de procesamiento de eventos central de CEP. Administra diferentes planes de ejecución y procesa eventos basándose en la lógica, con la ayuda de diferentes motores Siddhi. Recibe un conjunto de streams de eventos desde el Stream Manager, los procesa usando el motor Siddhi, y devuelve nuevos eventos en diferentes streams de vuelta al Stream Manager.

- **Event Formatter.** Los nuevos eventos generador en el Event Processor, se convierten a diferentes formatos como XML, Map, JSON, WSO2 Event y Texto. Al igual que el Event Builder, se puede configurar el formato de salida, de manera que podemos añadir cualquier otro formato.

- **Output Event Adapter.** Publica eventos a sistemas externos y vuelca los datos en bases de datos para análisis futuros. Al igual que el Input Adapter, este componente tiene implementaciones diferentes según el adaptador. Los más comunes están activados por defecto en el CEP, pero también puedes implementar adaptadores personalizados para casos de uso específicos. [7]

Atendiendo a esto, en nuestro caso, como Input Event Adapter utilizaremos el adaptador de Kafka, ya que vamos a correlacionar información que se encuentra en diferentes topics de kafka, como Event Builder y Formatter vamos a utilizar JSON. De todas maneras, vamos a centrarnos solo en el Event Processor, que es donde se encuentra el motor de Siddhi, y donde vamos a definir las diferentes reglas. Pero antes de crear dichas reglas, vamos a explicar por encima el lenguaje en las que están basadas: Siddhi Streaming SQL, el cual facilita al aprendizaje al ser muy similar a SQL.

5.3 Siddhi Streaming SQL.

Siddhi Streaming SQL fue diseñado para procesar streams de eventos, detectar comportamientos de eventos complejos y notificar sobre ellos en tiempo real. Cada query de Siddhi puede consumir de uno o más streams, procesar los eventos, y entonces generar un evento a otro stream o realizar alguna operación CRUD (Create, Read, Update, Destroy) sobre una tabla.

Todas las queries constan de una entrada y una salida. Algunas incluso contienen una sección de proyección. Una query simple con estas 3 secciones es de la siguiente manera:

```
from <input stream>
select <attribute name>, <attribute name>, ...
insert into <output stream/table>
```

5.3.1 Proyecciones

Siddhi soporta las siguientes proyecciones:

Acción	Descripción	Uso

Seleccionar Atributos Requeridos	Selección de un conjunto de atributos para ser insertados en el stream de salida.	`from rb_vault_post`
		`select app_name, message`
		`insert into cep_alert;`

Seleccionar todos los Atributos	Seleccionamos todos los atributos haciendo uso de (*) o ignorando el select.	`from rb_vault`
		`select *`
		`insert into cep_alert;`
		
		o
		
		`from rb_vault`
		`insert into cep_alert;`

Renombrando Atributos	Seleccionamos atributos y los introducimos en el stream de salida con otro nombre.	`from rb_vault_post`
		`select message as new_message`
		`insert into cep_alert;`

Valores Constantes	Añade valores constantes mediante la asignación con “as”	`from rb_vault_post`
		`select message, ‘C’ as new_value`
		`insert into cep_alert;`

Operaciones lógicas y matemáticas.	Se puede usar operaciones lógicas y matemáticas con atributos.	`from rb_flow_post`
		`select bytes * 10/5 +4, lan_ip in Table`
		`insert into cep_alert;`

Tabla 5-1. Proyecciones.

5.3.2 Filtros

Los filtros se incluyen en las queries para filtrar información de los streams de entrada basándose en una condición específica.

```
from <input stream>[<filter condition>]
select <attribute name>, <attribute name>, …
insert into <output stream>
```

Las operaciones que podemos realizar en los filtros son las siguientes, y además su funcionamiento es el mismo que en otros lenguajes:

- **< , <=, >, >=, ==, !=, and, or, not**
- **contains**
- **instanceof**

```
from TempStream[(roomNo >= 100 and roomNo < 210) and temp > 40]
select roomNo, temp
insert into HighTempStream;
```

5.3.3 Window

Las ventanas te permiten capturar un subconjunto de eventos basándose en un criterio específico. Cada stream solo puede tener una ventana. Las ventanas trabajan sobre diferentes tipos de eventos:

Tipos de Eventos	Descripción
current-events	La query solo tiene efecto a la llegada de un evento a la ventana
expired-events	La query mostrará resultados cuando los eventos expiren
all-events	Se muestran resultados tanto a la llegada como a la salida de un evento de la ventana.

Tabla 5-2. Tipos de Eventos.

Además, podemos utilizar una serie de funciones de agregación como las que encontramos en el SQL (avg, sum, count, max, min). Si se especifica una ventana, la agregación se realiza dentro de dicha ventana. En caso de que no se haya proveído una ventana, se realiza desde el inicio de la aplicación Siddhi. Dicho esto, los tipos de ventanas de los cuales podemos hacer uso a la hora de definir la query son los siguientes:

Tipo	Descripción	Uso
Length Window	Ventana que mantiene los últimos N eventos.	`from rb_vault_post[filter]#window.length(N)`
Length Batch Window	Procesa los últimos N eventos como un lote.	`from rb_vault_post[filter]#window.lengthBatch(N)`
Time Window	Mantiene los eventos recibidos dentro del intervalo T.	`from rb_vault_post[filter]#window.time(T min)`
Time Batch Window	Agrupa los eventos recibidos en el los últimos T minutos como un lote.	`from rb_vault_post[filter]#window.timeBatch(T min)`

Tabla 5-3. Tipos de ventana.

5.3.4 Join

Los Joins te permiten obtener un resultado combinado de dos stream en tiempo real basándose en una condición especificada. Este elemento es muy importante para lo que queremos conseguir, ya que con esto podemos relacionar información de dos topics diferentes, seleccionar cierta información y mandarla a otro topic donde podemos trabajar con esos datos nuevos.

Para combinar dos streams, cada uno de ellos tienen que estar asociados a una ventana para que haya un conjunto de eventos que se puedan usar para hacer el join. Los joins aceptan también diferentes condiciones para unir los eventos apropiados.

```
from <input stream>#window.<window name>(<parameter>, ... )
{unidirectional} {as <reference>}
    join
from <input stream>#window.<window name>(<parameter>, ... )
{unidirectional} {as <reference>}
    on <join condition>
select <attribute name>, <attribute name>, ...
insert into <output stream>
```

Por defecto, los eventos que lleguen a cualquier streaming dispararán el proceso de join, pero podemos controlar eso con la palabra clave

unidirectional, la cual provocará el proceso de join cuando llegue un evento a ese stream.

5.3.5 Patrones

Existe la posibilidad de detectar patrones en los eventos que nos van llegando a medida que avanza el tiempo. Los patrones podemos utilizarlos tanto con uno como entre varios streams. La sintaxis es la siguiente:

```
from (every)? <event reference>=<input stream>[<filter condition>]
->
     (every)? <event reference>=<input stream [<filter condition>]
->
    ...
    (within <time gap>)?
```

La flecha (->) se utiliza para indicar que un evento debería ir seguido de otro evento. No tiene porqué ocurrir inmediatamente, ya que podemos establecer una condición antes del signo flecha. Con **event-reference** podemos añadir una referencia a la tabla para poder acceder a ella posteriormente. La cláusula **within** es opcional, pero básicamente define el tiempo en el que los eventos tendrían que ocurrir. **Every** es otra cláusula opcional que especifica si el casado de eventos se debería disparar por cada evento que llegué al stream especificado por la condición.

```
from every( e1=TempStream ) -> e2=TempStream[ e1.roomNo == roomNo
and (e1.temp + 5) <= temp ]
    within 10 min
select e1.roomNo, e1.temp as initialTemp, e2.temp as finalTemp
insert into AlertStream;
```

Esta query manda una alerta cada vez que la temperatura de la habitación asciende 5 grados en un intervalo de 10 minutos.[8]

5.4 Casos de Uso

En este apartado vamos a definir unas reglas, para ilustrar cómo funciona el proceso de correlación. Vamos a crear las reglas para el producto openvpn, el cual tiene más posibilidades y además es más sencillo de manejar, ya que tiene una aplicación para smartphones donde puedes conectarte/desconectarte con bastante rapidez, con lo que podemos generar logs más eficazmente.

5.4.1 Openvpn Correlation

Uno de los propósitos de la creación de estas reglas, es relacionar información entre diferentes topics, con el fin de obtener nuevos datos. En este primer caso de uso, vamos a relacionar el Tráfico con el Vault.

Sabemos, ya que hemos trabajado en el tratamiento de los datos, que disponemos de un campo proveniente de OpenVPN llamado “openvpn lan ip”. Este campo representa la ip asignada al usuario por OpenVPN. También sabemos que, en el módulo Traffic, podemos ver el análisis del tráfico de red, donde tenemos un campo llamado “Lan IP”. Por lo tanto, al navegar por internet conectados a OpenVPN, tendremos en el campo “Lan IP” de Traffic, nuestra ip asignada por la vpn. Con tal de obtener los datos del Traffic asociados a esa IP, tenemos que crear una regla la cual lea de ambos topics, y filtre los mensajes a solo aquellos donde “OpenVPN Lan IP” y “Lan IP” sean iguales.

```
from rb_flow_post[ http_user_agent is null ]#window.length(40) as F
join
rb_vault_post[ not (openvpn_lan_ip is null)]#window.length(40) as V
on F.lan_ip == V.openvpn_lan_ip
```

Leemos de rb_flow_post con la condición de que el campo http_user_agent no esté presente, ya que como veremos más adelante, con una de las reglas estamos leyendo de rb_flow_post y mandando el resultado a rb_flow otra vez, por lo que si no comprobamos este campo, podemos generar un loop y provocar que la máquina se bloquee.
Una vez hayamos leído de los dos topics los mensajes que nos interesan, lo siguiente sería hacer la selección de atributos. En este caso, nos interesa saber la media de bytes consumidos por el usuario, el nombre de la aplicación que consumió dichos bytes, el nombre del usuario openvpn y la ip con la que navegó.

```
select avg(F.bytes) as cep_avgBytes, F.application_id_name as
cep_application, V.openvpn_user as cep_user, V.openvpn_lan_ip as
cep_lan_ip insert into cep_alert;
```

Al aplicar la regla y hacer un consumer de cep_alert, si navegamos por unas cuantas páginas conectados a la vpn, podremos ver los siguientes mensajes:

```
[root@rbmanager ~]# rb_consumer.sh -t cep_alert | grep -v DNS
Waiting cep_alert data (zookeeper: 127.0.0.1:2181) ...



{"cep_lan_ip":"172.33.33.14","cep_application":"Facebook","cep_user":"openvpn","cep_avgBytes":107.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Facebook","cep_user":"openvpn","cep_avgBytes":97.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Facebook","cep_user":"openvpn","cep_avgBytes":122.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Twitch","cep_user":"openvpn","cep_avgBytes":117.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Amazon","cep_user":"openvpn","cep_avgBytes":139.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Amazon","cep_user":"openvpn","cep_avgBytes":176.5}
{"cep_lan_ip":"172.33.33.14","cep_application":"Twitch","cep_user":"openvpn","cep_avgBytes":226.0}
{"cep_lan_ip":"172.33.33.14","cep_application":"Amazon","cep_user":"openvpn","cep_avgBytes":190.25}
{"cep_lan_ip":"172.33.33.14","cep_application":"Twitch","cep_user":"openvpn","cep_avgBytes":174.83333333333334}
{"cep_lan_ip":"172.33.33.14","cep_application":"PlayStore","cep_user":"openvpn","cep_avgBytes":165.33333333333334}
{"cep_lan_ip":"172.33.33.14","cep_application":"Amazon","cep_user":"openvpn","cep_avgBytes":159.5}
{"cep_lan_ip":"172.33.33.14","cep_application":"Twitch","cep_user":"openvpn","cep_avgBytes":94.0}
```

Figura 5-2. Regla CEP 1. Aplicaciones Usadas por openVPN.

Con esta regla en funcionamiento, podemos añadir un widget a nuestro dashboard mostrando las aplicaciones más usadas por openvpn.

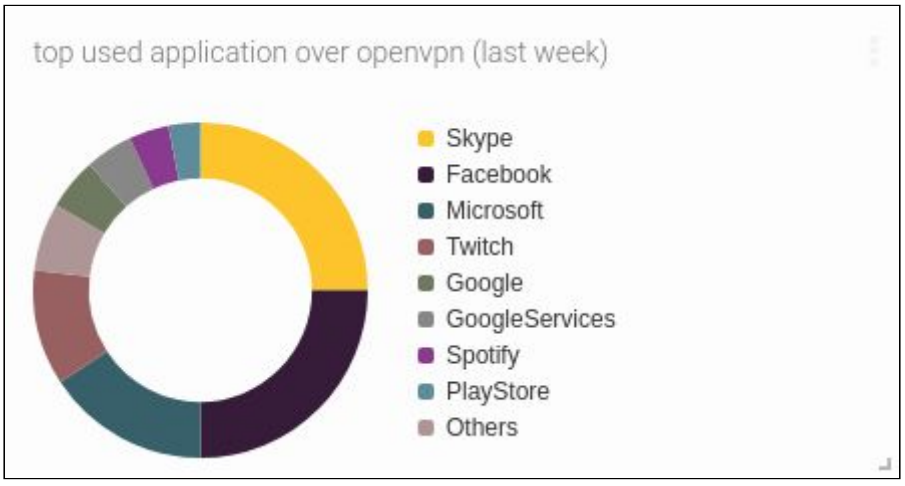


Figura 5-3. Regla CEP 1. Dashboard Aplicaciones Usadas.

5.4.2 Bytes recibidos y enviados

Otro caso de uso interesante para el administrador de sistemas, sería poder comprobar cuántos datos están consumiendo los usuarios conectados a la vpn. Para ello, habrá que crear una regla similar a la anterior, pero además de comprobar en rb_flow_post que el campo http_user_agent es null, también tendremos que comprobar que la dirección es “downstream”. Para los bytes enviados será lo mismo pero “upstream”.

```
from rb_flow_post[ http_user_agent is null and direction ==
```

```
'downstream' ]#window.time(2 min) as F
join
rb_vault_post[not (openvpn_lan_ip is null)]#window.length(20) as V
on F.lan_ip == V.openvpn_lan_ip
```

En esta ocasión estamos leyendo los mensajes que lleguen a rb_flow_post en los 2 últimos minutos y los últimos 20 mensajes de rb_vault_post. Hay que tener en cuenta que, al conectarse, openvpn solo manda un mensaje donde aparece el campo openvpn_lan_ip, osea que en el caso de que se conectasen 21 personas, se perdería la información de esa primera persona que se conectó, ya que pasaría a ser el mensaje 21. En principio esto no es problema ya que no disponemos de tantas personas en la empresa como para que 20 de ellas estén utilizando la vpn a la vez. Podríamos haber utilizado window.unique, pero esa ventana no está incorporada en la versión de la que disponemos.
Similar que antes, ya disponemos de los mensajes de ambos topics donde la ip sea la misma, es el momento de hacer la selección de atributos.

```
select F.direction,F.timestamp, F.sensor_name, F.wan_ip, F.lan_ip,
F.bytes, V.openvpn_user as http_user_agent
insert into rb_flow;
```

Aquí introducimos el posible loop del que hablábamos antes. Como podemos comprobar, estamos leyendo del rb_flow_post, y mandando de nuevo los resultados a rb_flow, el cual a su vez se encarga de mandar al rb_flow_post. Si no le decimos a la regla que lea de rb_flow_post donde http_user_agent no exista, podríamos generar un loop y bloquear la máquina.

Una vez aplicada la regla, podemos crear un nuevo dashboard donde se refleje los datos consumidos globalmente por openvpn. Pero también podríamos filtrar solo por cada usuario, ya que definimos el campo http_user_agent.

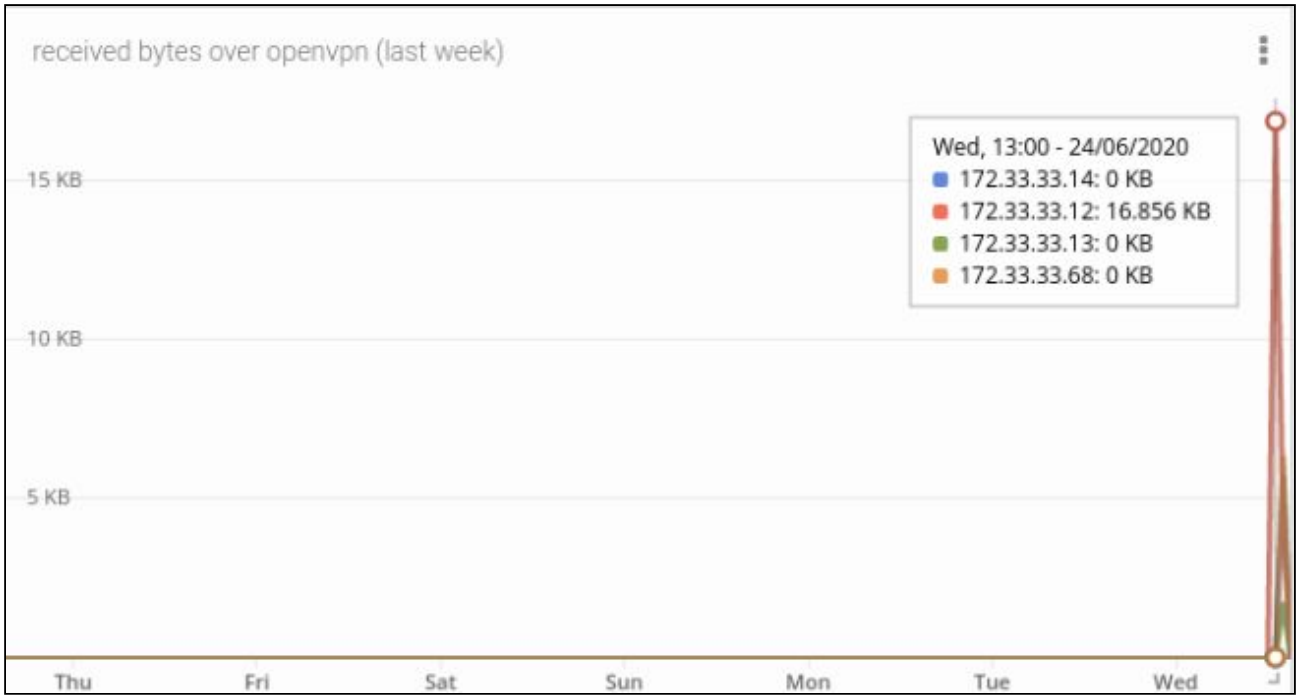


Figura 5-4. Regla CEP 2. Bytes Recibidos.

5.4.3 Tiempos de Conexión

Otro aspecto interesante para el administrador podría ser los tiempos de conexión de los usuarios, cuánto tiempo estuvieron conectados y si hubo alguna conexión fuera de los horarios normales de trabajo, lo cual podría resultar en un acceso no autorizado.

a) *Tiempo conectado.* En este caso, vamos a leer solo de un topic, pero vamos a utilizar también tablas intermedias. Primero, obtenemos los mensajes sobre los que queremos trabajar. Uno es el mensaje de cierre de sesión, el cual dispara la regla, y otro es el mensaje de inicio de sesión. Capturamos estos mensajes y los relacionamos para que tengan el mismo usuario y que el timestamp del mensaje de cierre de sesión sea mayor que el de inicio de sesión.

```
from rb_vault_post[ openvpn_status == 'restarting' ]#window.length(1)
as R
join
rb_vault_post[ openvpn_authentication == 'PAM auth succeeded'
]#window.length(40) as S
on R.openvpn_user == S.openvpn_user and
convert(convert(R.timestamp,'string'),'int')>convert(convert(S.timestam
p, 'string'),'int')
```

A continuación, vamos a calcular el tiempo conectado. Para ello, habrá que restar los timestamps. Hacemos uso de la función "convert" para la conversión de valores. Los resultados los incluimos en la tabla intermedia "#tabla1", de donde leeremos en la última parte de nuestra query.

```
convert(convert(R.timestamp,'string'),'int')>convert(convert(S.timestam
p, 'string'),'int')
select convert(convert(R.timestamp,
'string'),'int')-convert(convert(S.timestamp, 'string'),'int') as
my_time_worked, R.openvpn_user as user,
max(convert(convert(S.timestamp, 'string'),'int')) as time insert into
#table1;
```

Y finalmente, mandamos los campos que nos interesan a rb_vault_post.

```
from #table1
select user as openvpn_user, my_time_worked as
openvpn_time_connected
insert into outputStream;
```

Una vez aplicada la regla, y dejando una conexión abierta durante un tiempo, podemos ver que al cerrar la conexión, recibimos este mensaje en nuestro topic cep_alert con el tiempo en segundos que hemos estado conectados:

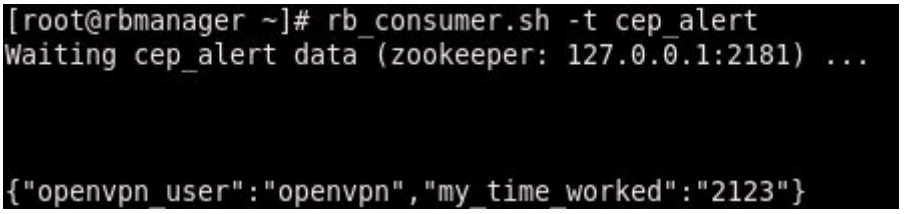


Figura 5-5. Regla CEP 3. Tiempo Conectado.

b) *Conexión fuera del horario laboral.* Puede que los accesos que se realicen fuera del horario de trabajo sean accesos no autorizados, por lo que se creará una regla que notifique de que usuario se conectó a esas horas. Leemos de rb_vault_post donde el campo openvpn_connection_hour exista, el cual marca la hora a la que se

inició la conexión. Convertimos el valor del campo a un entero para poder operar con él y mandamos todo a la tabla intermedia #table1.

```
from rb_vault_post[ not (openvpn_connection_hour is
null)]#window.length(20)
select convert(convert(openvpn_connection_hour, 'string'),
'int') as connection_hour, openvpn_user
insert into #table1;
```

Ahora leemos de la tabla intermedia, y capturamos solo los mensajes que tengan la hora de conexión encima de las 19 o por debajo de las 8, que se situaría fuera del horario laboral.

```
from #table1[ connection_hour > 19 and connection_hour < 8]
select connection_hour, openvpn_user
insert into cep_alert;
```

Aplicamos la regla, y realizamos una conexión fuera del horario laboral, y podemos ver en nuestro topic la hora y el usuario que se conectó:

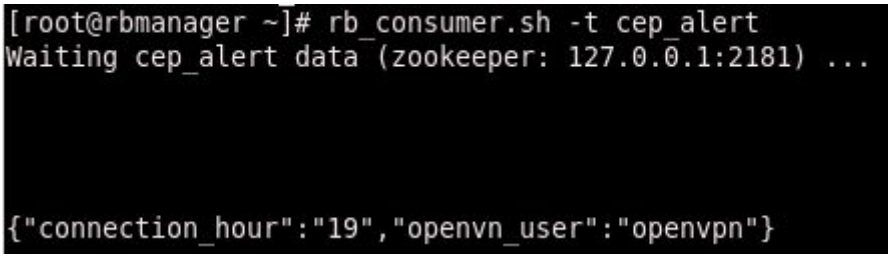


Figura 5-6. Regla CEP 3. Tiempo Conectado.

A partir de aquí, podemos decidir qué hacer con estos datos. Si crear una alarma que nos notifique cada vez que alguien se conecta fuera del trabajo o crear un widget que nos muestre el top de usuarios que se han conectado fuera del horario laboral y a que horas.

5.5 Ayuda al usuario para la creación de reglas.

En un principio se planteó la idea de crear un nuevo módulo para visualizar los datos correlacionados por el motor, pero más adelante vimos que esto no era necesario, ya que cuando una regla se ejecuta y genera información, por

defecto manda un mensaje al syslog con dichos datos, por lo que para visualizar dichos datos sólo tendríamos que tratarlos como hicimos anteriormente con Logstash, para poder quedarnos con los campos que nos interesan y mandarlos al manager para su representación gráfica.
Por lo que a raíz de esto, surgió la idea de crear unos formularios que permitan al usuario la creación de reglas de manera sencilla y sin tener que conocer ni el lenguaje Siddhi, ni tampoco el nombre de las dimensiones que se quieren seleccionar dentro de un topic, ya que para ello el usuario tendría que tener más conocimiento de cómo funciona el producto por dentro. La idea es tener un botón en la vista de las reglas donde podamos crear una regla de correlación donde haya 2 topics implicados de manera sencilla y sin tener que conocer el nombre de las dimensiones de cada módulo de manera interna (por ejemplo, el campo Application de Traffic internamente se llama por :application_id_name, con esta aproximación no hace falta saber de este segundo nombre).

El código desarrollado para este objetivo se colocará en el apéndice C.

Vamos a coger como ejemplo a seguir una de las reglas que ya hemos creado, para mostrar cómo podemos crear exactamente la misma regla pero sin conocer SiddhiSQL. La regla que vamos a construir es la que nos permite ver los bytes que están consumiendo los usuarios conectados a la vpn:

```
from rb_flow_post[ http_user_agent is null and direction ==
'downstream' ]#window.time(2 min) as F
join
rb_vault_post[not (openvpn_lan_ip is null)]#window.length(20)
as V
on F.lan_ip == V.openvpn_lan_ip
select F.direction,F.timestamp, F.sensor_name, F.wan_ip,
F.lan_ip, F.bytes, V.openvpn_user as http_user_agent
insert into outputStream;
```

Lo primero es especificar el nombre y el topic al que deseamos mandar la información correlacionada. Lo segundo sería especificar los topics de donde queremos leer la información, y los filtros asociados a dichas lecturas. Para el caso del primer topic:

```
from rb_flow_post[ http_user_agent is null and direction ==
'downstream' ]#window.time(2 min) as F
```

Tendríamos el siguiente equivalente en nuestro formulario:

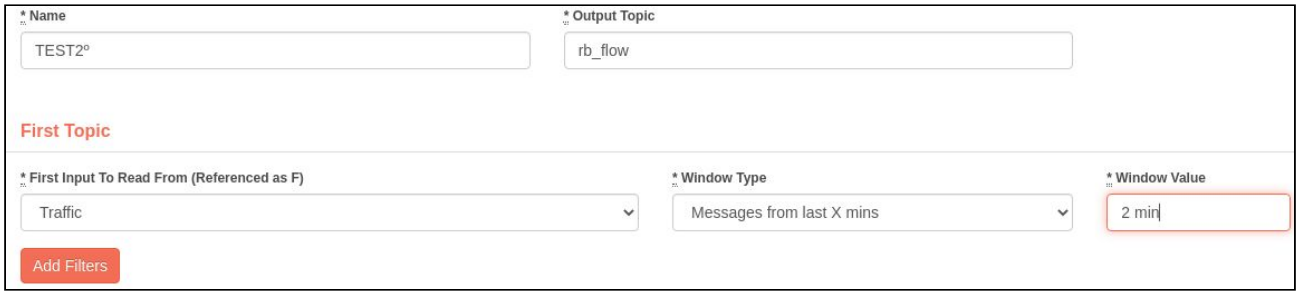


Figura 5-7. Formulario. Primer Topic.

Pero como además tenemos la posibilidad de añadir filtros a nuestras lecturas de mensajes, disponemos de un botón para añadir más filtros. En este caso, siguiendo el ejemplo anterior, tendríamos que añadir dos filtros, uno para el http_user_agent y otro para la direction. Con nuestro formulario esto es fácil, ya que no hace falta conocer la sintaxis de los filtros ni como se referencian las variables. Luego tendríamos:

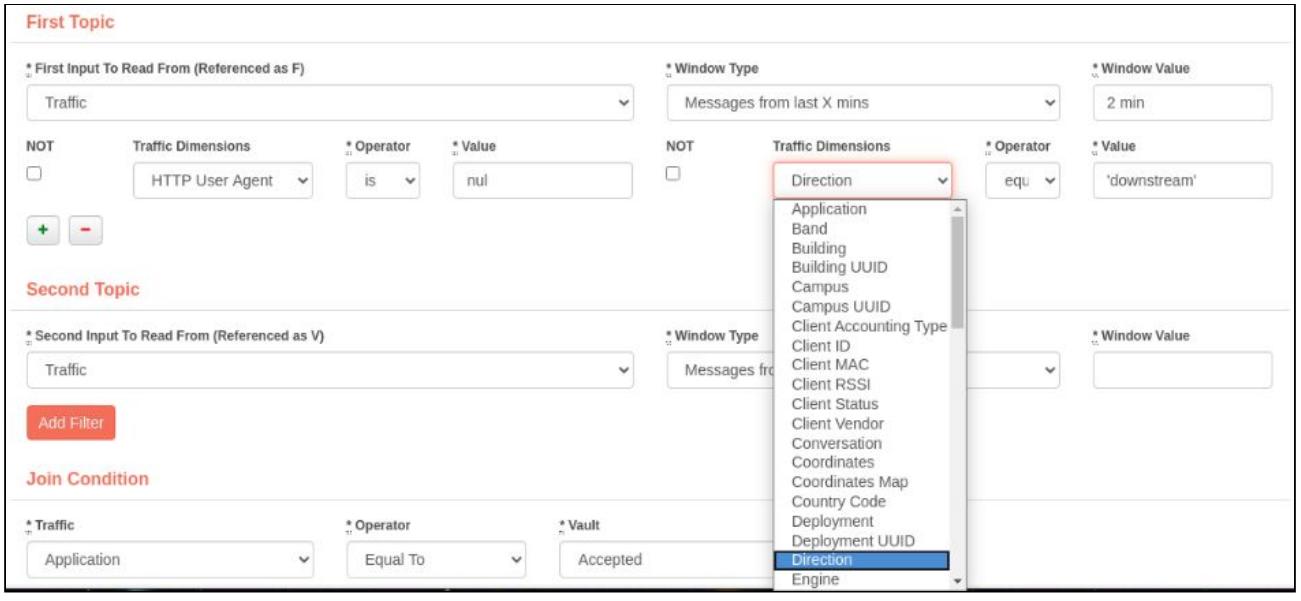


Figura 5-8. Formulario. Añadiendo filtros.

Para el segundo topic sería lo mismo, pero estableciendo que vamos a leer de Vault, de donde vamos a coger los últimos 40 mensajes, donde openvpn_lan_ip no es null.

Lo siguiente es establecer la condición de unión.

```
on F.lan_ip == V.openvpn_lan_ip
```

Cuyo equivalente en nuestro formulario sería:

Figura 5-9. Formulario. Join.

Por último, vamos a hacer la selección de las dimensiones de Vault y Traffic que nos interesa poner en nuestro log de cep.

```
select F.direction,F.timestamp, F.sensor_name, F.wan_ip,
F.lan_ip, F.bytes, V.openvpn_user as http_user_agent insert
into outputStream;
```

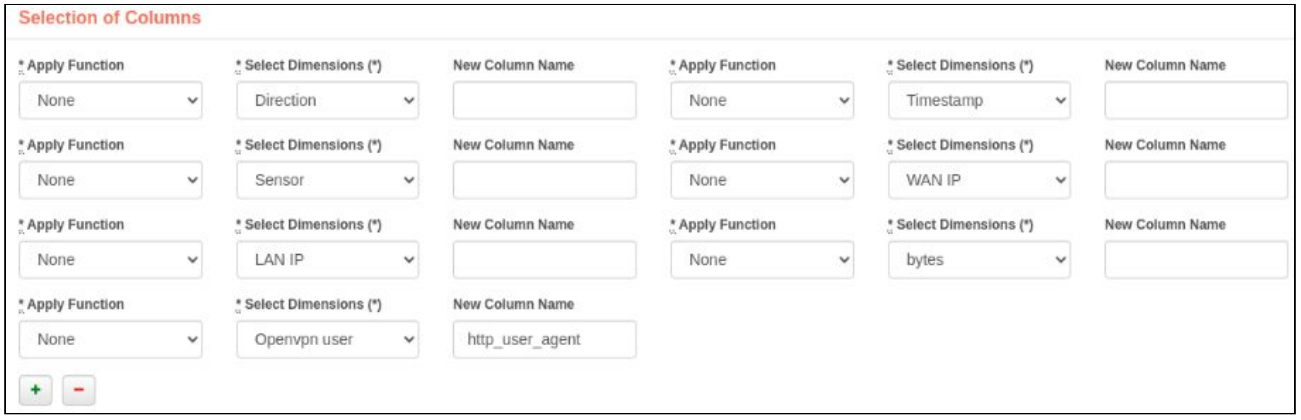


Figura 5-10. Formulario. Seleccionando Atributos.

Pulsamos en crear, y al editar la regla podemos comprobar que efectivamente, la query se ha construido de manera correcta, por lo que podemos aplicar y empezar a correlacionar los datos sin necesidad de haber aprendido Siddhi, lo único que necesitábamos era conocer los datos que queríamos relacionar y en que topics se encontraban.

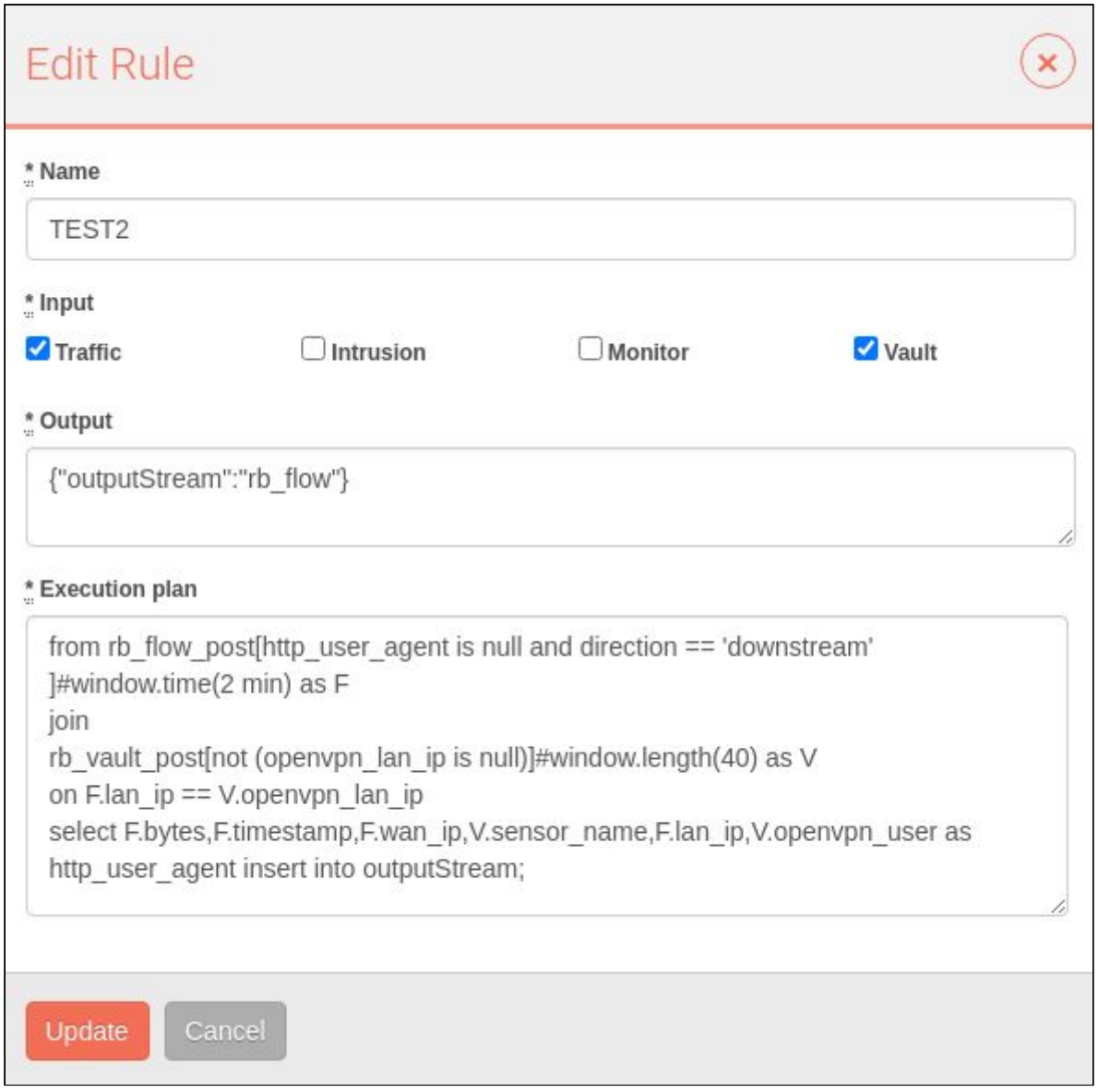


Figura 5-11. Formulario. Regla Creada.

Capítulo 6. Conclusiones y líneas futuras

Con la finalización de este trabajo termina una etapa de investigación y aprendizaje no solo sobre el producto de la empresa redBorder sino también de las diferentes metodologías y servicios que se utilizan a día de hoy con bastante frecuencia en el mundo del Big Data y la ciberseguridad.

Hemos adquirido unas nociones fundamentales sobre como funciona todo el procesado de datos en una situación real de empresa donde se necesitaba de la integración de nuevos productos relacionados con la seguridad. Hemos recorrido y desarrollado las diferentes fases en el tratamiento de los datos (adquisición, normalización, enriquecimiento y la visualización) obteniendo una visión global tanto del producto como del flujo de datos a través de los diferentes servicios que lo componen, los cuales, entre los que se encuentran Apache Kafka, Druid o Logstash, son herramientas fundamentales en empresas con presencia en el mundo del Big Data, como pueden ser Netflix, Spotify o Twitter.

Además de todo esto, hemos aprendido a utilizar el motor de correlación y las reglas Siddhi para extraer información extra que de otra manera no seríamos capaces de obtener. Por último, nos hemos familiarizado con el desarrollo web mediante Ruby on Rails, una parte bastante importante del proyecto ya que es en la web donde se procede a visualizar toda la información relevante a la organización y donde al final todos los servicios y herramientas involucrados van a entrar en juego.

En cuanto a unas líneas futuras de mejora para el proyecto, sería conveniente tener la opción de un nuevo apartado en el manager, donde podamos instalar los diferentes plugins de manera automática, sin tener que acceder a la web vía ssh y ejecutar los scripts manualmente. Además, sería bueno ofrecer al usuario la posibilidad de crear más tipos de reglas de correlación de manera sencilla, ya que actualmente solo facilitamos la creación de reglas similares a las creadas anteriormente, donde intervienen 2 topics de Kafka. También hay bastante margen de optimización en el formulario de creación de reglas (utilizar menos código, impedir entradas incorrectas, recordar de utilizar el timestamp en caso de añadir una agregación...etc).

Capítulo 7.
Summary and Conclusions

With the completion of this project, an investigation and learning phase ends. Not just about the product of redBorder, but also the different methodologies and services used on a daily basis in the world of Big Data and Cyber Security.

We have acquired fundamental knowledge about how data processing in different situations of a real company functions, where there was a need to integrate new products related to security. We have gone over and developed the different phases on data management (acquisition, normalization, enrichment and visualization) obtaining a global vision not only from the product, but also from its data flow, of which we have Apache Kafka, Druid or Logstash, these are fundamental tools used in big companies in the world of Big Data, as Netflix, Spotify or Twitter.

Furthermore, we have learnt about the correlation engine and the siddhi rules it uses to extract information that we would not have been able to obtain in a different way. Lastly, we have familiarized ourselves with Ruby on Rails development, a very important part of the project as the whole front-end, which has the most important part to the users as this will visualize all the processed data.

Considering future improvements for the project, it would be convenient to have the option where the different plugins can be installed and managed in a user friendly manner, without the need to access the back-end using ssh and manually running the plugin installation scripts. As well as offering the user the possibility to create more types of rules for the correlation engine in a more user friendly way, as for now we are only offering the creation of similar rules as they have been created previously (using 2 Kafka topics). There is also room for improvement in the user friendly form (optimize code, don’t allowe forbidden inputs, remember the user to use timestamp along with aggregations)

Capítulo 8. Presupuesto

Para el desarrollo de este trabajo, han sido necesarias alrededor de 320 horas de trabajo. Cómo se ha desarrollado conjuntamente con la empresa redBorder, en la que me encuentro actualmente trabajando, y teniendo en cuenta una media de 7 euros por hora, partimos de 2240 €. A eso, habrá que sumarle el precio de la máquina utilizada para el desarrollo del proyecto, la cual tiene que contar con 64GB de RAM, puesto que el motor de correlación consume muchos recursos. Con una media de 400 euros por 64 GB de ram, y contando el resto de componentes (torre, discos duros, 3 monitores), tendríamos un ordenador de unos 900 euros. Sumado al tiempo de trabajo, tenemos que el presupuesto total para el desarrollo de este TFG se situaría alrededor de los 3000 euros. Podemos ver dichos costes desglosados en la siguiente tabla:

Número de horas trabajadas	320
Coste por Hora	7€
Precio PC	900€
Total	~3000€

Tabla 8-1. Costes del proyecto.

Apéndice A.
Scripts para la automatización de la creación de dimensiones.

A.1. OpenVPN

```
#!/bin/bash

##########  Variable Declarations - Start  ##########
#name of the current script
export SCRIPT_NAME=configure-openvpn.sh
#version of the current script
export SCRIPT_VERSION=1.6.4

export LOGSTASH_ETCDIR_CONF="/etc/logstash/pipelines/vault"

#name of the service, in this case OpenVPN
export SERVICE="openvpn"

export RB_EXTENSIONS_DIR="/opt/rb/var/rb-extensions"

#name and location of openvpn logstash file
export LOGSTASH_CONFFILE=$LOGSTASH_ETCDIR_CONF/43_openvpn.conf

export MANUAL_CONFIG_INSTRUCTION="Manual instructions to configure
openvpn is available at
https://suport.redborder.com/docs/openvpn-server-logs#manual. Logstash
troubleshooting instructions are available at
https://support.redborder.com/docs/troubleshooting-logstash/"

export INSTALLED_FILE=$EXTENSION_DIR/extension-installed.txt

##########  Variable Declarations - End  ##########


##########  OpenVPN Functions  - Start   ###########
```

```
#function to write the contents of  logstash config file

writeFileContents()
{
      echo "INFO" "INFO: Creating file $LOGSTASH_CONFFILE"
      sudo touch $LOGSTASH_CONFFILE
      sudo chmod o+w $LOGSTASH_CONFFILE

        if [ ! -f $DIMENSIONS_CONFFILE ]; then
          echo "cannot find the dimension file to setup the logstash
pipeline. exiting..."
          exit 1
        fi

        # build the array of original dimensions
ORIG_DIM=$(
ruby - <<END
  require 'yaml'
  original_dimensions = []
  dimensions = {}
  dimensions = dimensions.merge(YAML.load_file("$DIMENSIONS_CONFFILE"))
  dimensions.each { | key, dimension |
original_dimensions.push(dimension['original_dimension']) }
  puts original_dimensions.to_s
END
)

imfileStr='#Version: '$SCRIPT_VERSION'
filter {
  if "openvpnas" in [app_name] {
    mutate {
      gsub => [
        "message", "\"", "",
        "message", "/", " ",
        "message", "[\\]", " ",
        "message", "'u\''", "",
        "message", "'\''", "",
        "message", "  ", " "
      ]
    }
```

```
    grok {
      patterns_dir =>
["/etc/logstash/pipelines/vault/patterns/openvpn"]
      match => [
        "message","%{OPENVPN_P1}",
        "message","%{OPENVPN_P2}",
        "message","%{OPENVPN_P3}",
        "message","%{OPENVPN_P4}",
        "message","%{OPENVPN_P5}",
        "message","%{OPENVPN_P6}",
        "message","%{OPENVPN_P7}",
        "message","%{OPENVPN_P8}",
        "message","%{OPENVPN_P9}",
        "message","%{OPENVPN_P10}",
        "message","%{OPENVPN_P11}",
        "message","%{OPENVPN_P12}",
        "message","%{OPENVPN_P13}",
        "message","%{OPENVPN_P14}",
        "message","%{OPENVPN_P15}",
        "message","%{OPENVPN_P16}",
        "message","%{OPENVPN_P17}",
        "message","%{OPENVPN_P18}",
        "message","%{OPENVPN_P19}",
        "message","%{OPENVPN_P20}",
        "message","%{OPENVPN_P21}",
        "message","%{OPENVPN_P22}",
        "message","%{OPENVPN_P23}",
        "message","%{OPENVPN_P24}",
        "message","%{OPENVPN_P25}",
        "message","%{OPENVPN_P26}",
        "message","%{OPENVPN_P27}",
        "message","%{OPENVPN_P28}",
        "message","%{OPENVPN_P29}",
        "message","%{OPENVPN_P30}",
        "message","%{OPENVPN_P31}",
        "message","%{OPENVPN_P32}",
        "message","%{OPENVPN_P33}",
        "message","%{OPENVPN_P34}",
        "message","%{OPENVPN_P35}",
        "message","%{OPENVPN_P36}",
        "message","%{OPENVPN_P37}",
```

```
        "message","%{OPENVPN_P38}",
        "message","%{OPENVPN_P39}",
        "message","%{OPENVPN_P40}",
        "message","%{OPENVPN_P41}",
        "message","%{OPENVPN_UNPROCESSED}"
      ]
    }
  }
}'

#change the file to variable from above and also take the directory of
the openvpn log file.

sudo cat << EOIPFW > $LOGSTASH_CONFFILE
$imfileStr
EOIPFW

     restartLogstash
}

#function to write the contents of openvpn logstash config file
writeDimensionsFile()
{
     echo "INFO: Creating dimensions file $DIMENSIONS_CONFFILE"
     sudo touch $DIMENSIONS_CONFFILE
     sudo chmod o+w $DIMENSIONS_CONFFILE

imfileStr=$SERVICE'_lan_ip:
  name: "OpenVPN LAN IP"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_lan_ip"
'$SERVICE'_wan_ip:
  name: "OpenVPN WAN IP"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_wan_ip"
'$SERVICE'_port:
  name: "OpenVPN Port"
  category: "openvpn"
  type: string
```

```
  original_dimension: "openvpn_port"
'$SERVICE'_peer_info:
  name: "OpenVPN Peer Info"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_peer_info"
'$SERVICE'_msg:
  name: "OpenVPN Message"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_msg"
'$SERVICE'_protocol:
  name: "OpenVPN Protocol"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_protocol"
'$SERVICE'_status:
  name: "OpenVPN Status"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_status"
'$SERVICE'_authentication:
  name: "OpenVPN Authentication"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_authentication"
'$SERVICE'_user:
  name: "OpenVPN User"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_user"
'$SERVICE'_client:
  name: "OpenVPN Client"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_client"
'$SERVICE'_control_channel:
  name: "OpenVPN Control Channel"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_control_channel"
```

```
'$SERVICE'_session:
  name: "OpenVPN Session ID"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_session"
'$SERVICE'_concurrent_connections:
  name: "OpenVPN Concurrent Connections"
  category: "openvpn"
  type: string
  original_dimension: "openvpn_concurrent_connections"
'$SERVICE'_connection_hour:
  name: "OpenVPN Connection Hour"
  category: "openvpn"
  type: int
  original_dimension: "openvpn_connection_hour"'

sudo cat << EOIPFW > $DIMENSIONS_CONFFILE
$imfileStr
EOIPFW

}

##########  OpenVPN Functions - End   ###########


##########  Start of the common part  ###########

#downloads configure-extensions.sh
setupDependencies() {
  ERROR="true"
  # Check if there is internet or/and can access to the web of the
script
  response=$(curl --write-out %{http_code} --silent --output /dev/null
http://repo.redborder.com/install/common/configure-extensions.sh)
  if [[ $response -eq 200 ]]; then
    echo "INFO: Downloading dependencies.."
    curl -s -o configure-extensions.sh
http://repo.redborder.com/install/common/configure-extensions.sh
  else
    echo "configure-extensions script could not be retrieved, local
copy is needed"
```

```
  fi

  if [ -f configure-extensions.sh ]; then
    if [ `head -1 configure-extensions.sh | grep bash | wc -l` -gt 0 ];
then
      source configure-extensions.sh "being-invoked"

      #definitions of variables relying on the
configure-extensions-script
      EXTENSION_DIR=$RB_EXTENSIONS_DIR/$SERVICE
      DIMENSIONS_CONFFILE=$EXTENSION_DIR/dimensions.yml

      ERROR="false"
    else
      echo "local configure-extensions script is not valid, we need to
exit"
    fi
  fi

  if [ "$ERROR" != "false" ]; then
    echo "ERROR: Missing dependencies!"
    exit 1
  fi
}

#Setup OpenVPN Patterns
setupPatterns(){
     echo "INFO: Creating OpenVPN Patterns"
     sudo touch /opt/rb/share/logstash-rules/$SERVICE

imfileStr='# Start/Stop
DATESTAMP_OTHER_CUSTOM %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}

### Pattern 1 for Generic peer info ###
OPENVPN_P1      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} peer info:
%{GREEDYDATA:openvpn_peer_info}
OPENVPN_P2      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} peer
info: %{GREEDYDATA:openvpn_peer_info}
OPENVPN_P3      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date} SENT CONTROL
```

```
\[%{WORD:openvpn_user}]: %{GREEDYDATA:openvpn_msg} \(

### Pattern 4 for Generic verify status ###
OPENVPN_P4      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} VERIFY
%{GREEDYDATA:openvpn_status}: %{GREEDYDATA:openvpn_msg}

### Patterns for Connections ###
OPENVPN_P5      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_protocol} %{GREEDYDATA:openvpn_msg}
\[%{GREEDYDATA:openvpn_type}]%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port
}
OPENVPN_P6      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
Connection reset, %{WORD:openvpn_status}
OPENVPN_P7      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{WORD:remove}\[soft,connection-reset] received, client-instance
%{WORD:openvpn_status}

### Pattern 8 for Client-instance status ###
OPENVPN_P8      : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} %{GREEDYDATA:openvpn_msg},
client-instance %{WORD:openvpn_status}

### Patterns for Authentfication error Message ###
OPENVPN_P9      \(openvpnas:auth\): authentication
%{GREEDYDATA:openvpn_authentication}; logname= uid=0 euid=0 tty=
ruser=%{WORD:openvpn_user} rhost=%{IP:openvpn_lan_ip}
OPENVPN_P10          \(openvpnas:auth\): %{GREEDYDATA:openvpn_msg}
OPENVPN_P11          OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{GREEDYDATA:openvpn_msg}%{IP:remove}:%{NUMBER:remove}
OPENVPN_P12          OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} %{GREEDYDATA:openvpn_msg}
\[%{WORD:openvpn_user}]: %{GREEDYDATA:openvpn_authentication}
OPENVPN_P13          OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{GREEDYDATA:openvpn_msg} %{IP:openvpn_lan_ip} -> %{WORD:openvpn_user2}
%{IP:remove}:%{NUMBER:remove}
OPENVPN_P14          OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
```

```
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} MULTI:
Learn: %{IP:openvpn_lan_ip} %{GREEDYDATA:openvpn_msg}
OPENVPN_P15           OUT: %{DAY} %{MONTH} %{MONTHDAY}
%{DATA:openvpn_connection_hour}:%{NUMBER}:%{NUMBER} %{YEAR}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{DATA:openvpn_msg} %{IP:remove}:%{NUMBER:remove}: %{IP:openvpn_lan_ip}
OPENVPN_P16           OUT: %{DAY} %{MONTH} %{MONTHDAY}
%{DATA:openvpn_connection_hour}:%{NUMBER}:%{NUMBER} %{YEAR}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{DATA:openvpn_msg} %{IP:remove}:%{NUMBER:remove}, %{IP:openvpn_lan_ip}
OPENVPN_P17           OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
MULTI: %{GREEDYDATA:openvpn_authentication}: %{GREEDYDATA:openvpn_msg},
CLI:\[
OPENVPN_P18           VPN Auth Failed:
%{GREEDYDATA:openvpn_authentication}: %{GREEDYDATA:openvpn_errorcause}

### Patterns for Successful Authentfication ###
OPENVPN_P19           : %{NUMBER:openvpn_status}, reason:
%{GREEDYDATA:openvpn_authentication}, serial_list:
%{GREEDYDATA:openvpn_serial_list}, user: %{GREEDYDATA:openvpn_user},
proplist: {%{GREEDYDATA}}, common_name: %{GREEDYDATA:openvpn_user},
serial: %{GREEDYDATA:openvpn_serial}} cli=%{GREEDYDATA:openvpn_client}
OPENVPN_P20           : %{NUMBER:openvpn_status}, reason:
%{GREEDYDATA:openvpn_authentication}, serial_list:
%{GREEDYDATA:openvpn_serial_list}, user: %{GREEDYDATA:openvpn_user},
proplist: {{%{GREEDYDATA}}}, common_name: %{GREEDYDATA:openvpn_user},
serial: %{GREEDYDATA:openvpn_serial}
OPENVPN_P21           : %{NUMBER:openvpn_status}, session_id:
%{GREEDYDATA:openvpn_sessionid}, reason:
%{GREEDYDATA:openvpn_authentication}, serial_list:
%{GREEDYDATA:openvpn_serial_list}, user: %{GREEDYDATA:openvpn_user},
proplist: {%{GREEDYDATA}}, common_name: %{GREEDYDATA:openvpn_user},
serial: %{GREEDYDATA:openvpn_serial}} cli=%{GREEDYDATA:openvpn_client}

### Pattern 22for TLS ###
OPENVPN_P22           : %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} %{WORD:openvpn_protocol}:
%{GREEDYDATA:openvpn_msg}

### Pattern 23 for Control Channel ###
OPENVPN_P23           %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
```

```
%{WORD:openvpn_user}:%{NUMBER:openvpn_port} Control Channel:
%{GREEDYDATA:openvpn_control_channel}
OPENVPN_P24           %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} Control Channel:
%{GREEDYDATA:openvpn_control_channel}

### Pattern 25 for Memory Usage ###
OPENVPN_P25           MEMORY usage: %{NUMBER:openvpn_memory_usage}

### Pattern 26 for messages like (14463, <type function>) ###
OPENVPN_P26           \[-\] \(%{GREEDYDATA:openvpn_type}\)

### Pattern 27 for Connection time-out ###
OPENVPN_P27           WEB OUT: %{TIMESTAMP_ISO8601:openvpn_date}
%{GREEDYDATA:remove} expired session %{GREEDYDATA:openvpn_session}

### Cleanup patterns ###
OPENVPN_P28           OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} %{GREEDYDATA:openvpn_msg}
OPENVPN_P29           OUT: %{TIMESTAMP_ISO8601:openvpn_date}
\[%{GREEDYDATA:remove}]
%{GREEDYDATA:openvpn_msg}:%{GREEDYDATA:openvpn_status}
OPENVPN_P30           OUT: %{TIMESTAMP_ISO8601:openvpn_date}
\[%{GREEDYDATA:remove}] %{GREEDYDATA:openvpn_status}
<%{GREEDYDATA:openvpn_msg}>
OPENVPN_P31           OUT: %{TIMESTAMP_ISO8601:openvpn_date}
\[%{GREEDYDATA:remove}] %{GREEDYDATA:openvpn_status}
(%{GREEDYDATA:openvpn_msg})
OPENVPN_P32           OUT: %{TIMESTAMP_ISO8601:openvpn_date}
\[%{GREEDYDATA:remove}] %{GREEDYDATA:openvpn_msg}: {status:
%{NUMBER:openvpn_status}, reason: %{GREEDYDATA:openvpn_msg}, user:
%{WORD:openvpn_user}
OPENVPN_P33           CLIENT_RESET: %{GREEDYDATA:openvpn_remove},
v1=%{WORD:openvpn_v1} v2=%{WORD:openvpn_v2}
OPENVPN_P34           VPN Auth Failed: %{GREEDYDATA:openvpn_msg},
client username=%{WORD:openvpn_user}, DB username=%{WORD}
OPENVPN_P35           %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port} SENT CONTROL
\[%{WORD:openvpn_user}]: AUTH_FAILED,LICENSE: Access Server license
failure: maximum concurrent_connections exceeded
\(%{NUMBER:openvpn_concurrent_connections}
```

```
OPENVPN_P36           OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
MANAGEMENT: %{GREEDYDATA:openvpn_remove} concurrent_connections
exceeded
OPENVPN_P37           HTTPChannel,%{NUMBER:openvpn_num},] License Info
{concurrent_connections: %{NUMBER:openvpn_concurrent_connections}}
OPENVPN_P38           -] License Info {concurrent_connections:
%{NUMBER:openvpn_concurrent_connections}}
OPENVPN_P39           OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{WORD:openvpn_user} %{IP:openvpn_wan_ip}:%{NUMBER:openvpn_port}
%{GREEDYDATA:openvpn_msg}
OPENVPN_P40           OUT: %{DATESTAMP_OTHER_CUSTOM:openvpn_date}
%{GREEDYDATA:openvpn_msg}

### Pattern 41 for GARBAGE usage ###
OPENVPN_P41           GARBAGE usage: %{NUMBER:openvpn_garbage_usage}

### Pattern for any UNPROCESSED Message ###
OPENVPN_UNPROCESSED   %{GREEDYDATA:openvpn_unprocessed}'

sudo cat << EOIPFW > /opt/rb/share/logstash-rules/$SERVICE
$imfileStr
EOIPFW

  if [ ! -f "/etc/logstash/pipelines/vault/patterns/$SERVICE" ]; then
      sudo ln -s /opt/rb/share/logstash-rules/$SERVICE
/etc/logstash/pipelines/vault/patterns/$SERVICE
  fi
}

#display usage syntax
usage()
{
cat << EOF
usage: configure-$SERVICE [-i to install]
usage: configure-$SERVICE [-r to rollback]
usage: configure-$SERVICE [-v for version]
usage: configure-$SERVICE [-u to update]
usage: configure-$SERVICE [-h for help]
EOF
}
```

```
##########  Get Inputs from User - Start  ##########

if [ $# -eq 0 ]; then
    usage
  exit
else
while [ "$1" != "" ]; do
    case $1 in
      -r | --rollback )
          REDBORDER_ROLLBACK="true"
          ;;
      -i | --install )
          INSTALL="true"
          ;;
      -v | --version )
          echo "The version of $SERVICE is $SCRIPT_VERSION, to check
for any update use ./configure-$SERVICE-$SCRIPT_VERSION -u"
          exit
          ;;
      -u | --update )
          echo "Checking for updates..."
          UPDATING="true"
          ;;
      -h | --help )
          usage
          exit
          ;;

    esac
    shift
done
fi

startCheck(){
    setupDependencies
    checkIfUserHasRootPrivileges
    SVERSION=`cat $LOGSTASH_CONFFILE | head -1 | awk '{print $2}'`
}

if [ "$REDBORDER_ROLLBACK" != "" ]; then
    startCheck
```

```
    removeRedborderConf
    removePatterns

elif [ "$INSTALL" == "true" ]; then
    startCheck
    if [ "$SVERSION" == "$SCRIPT_VERSION" ]; then
      echo "Already up to date"
    else
      rm -f configure-$SERVICE-*
      installRedborderConf
      setupPatterns
    fi

elif [ "$UPDATING" == "true" ]; then
    startCheck
    checkForUpdate

else
    usage
fi

##########  Get Inputs from User - End  ##########
```

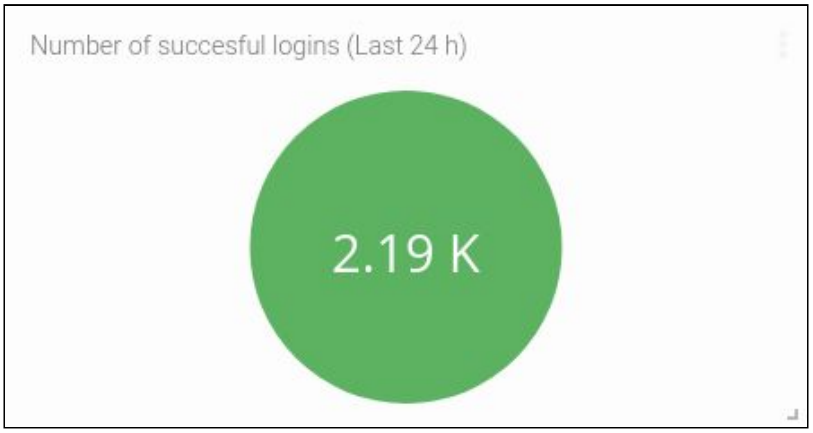
Apéndice B. Dashboards.

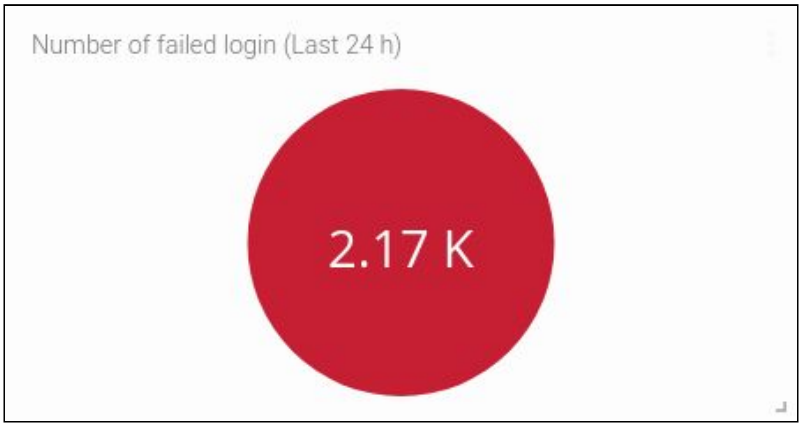
B.1. Juniper

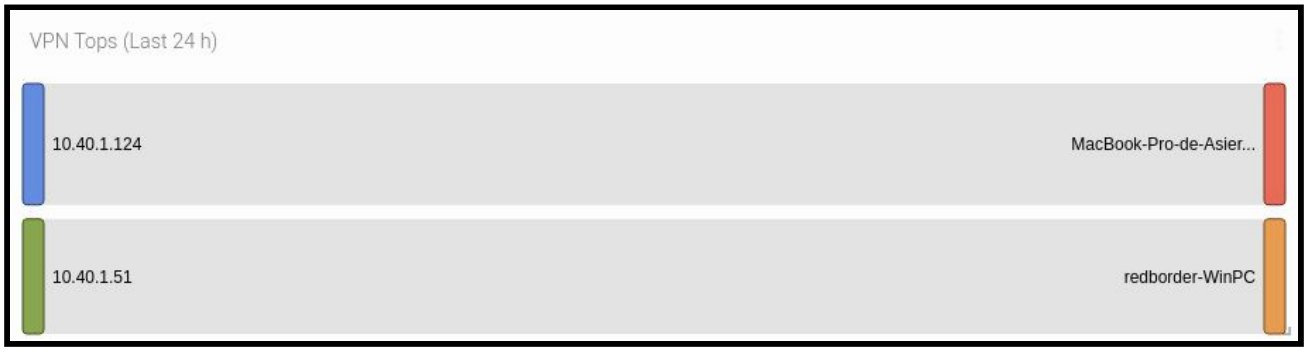
Juniper users, action and status (Last 24 h)

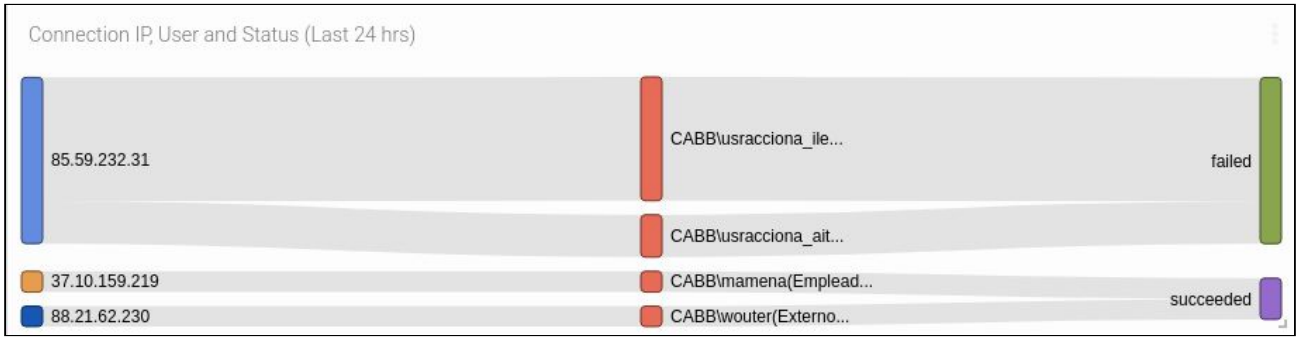
Juniper user	Juniper action	Juniper status	events
CABB\usracciona_ilekuona(Externos_DA)[]	Primary authentication	failed	6.32 K
CABB\wouter(Externos_DA)[Externos, Lims_laboratorio]	connection	Closed	4.3 K
CABB\usrordunte_mrodri(Externos_DA)[Externos, ORDUNTE]	Authentication	successful	4.25 K
CABB\usracciona_ilekuona(Externos_DA)[]	Account locked-out	password management	4.12 K
CABB\usrasaser_mgutierrez(Externos_DA)[Externos, Asaser_Zonas]	Authentication	successful	3.28 K
CABB\usrasaser_igalicia(Externos_DA)[Externos, Asaser_Zonas]	Authentication	successful	3.19 K

VPN Tops (Last 24 h)





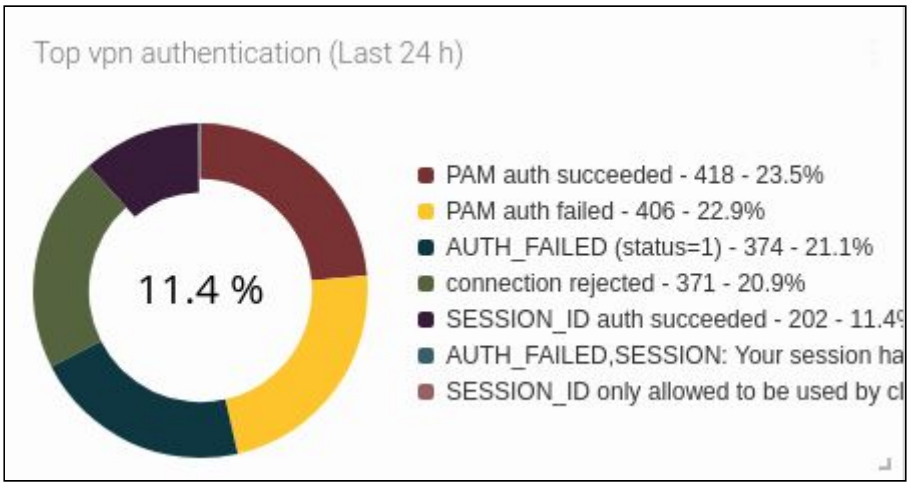


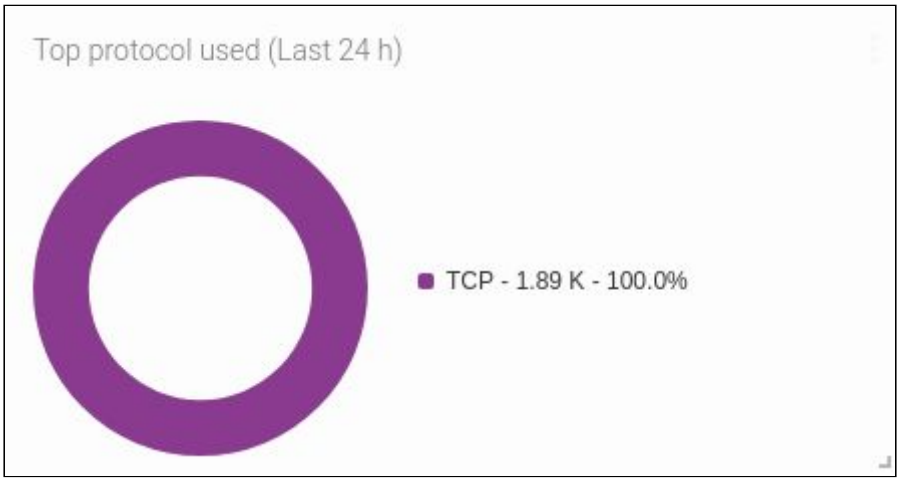


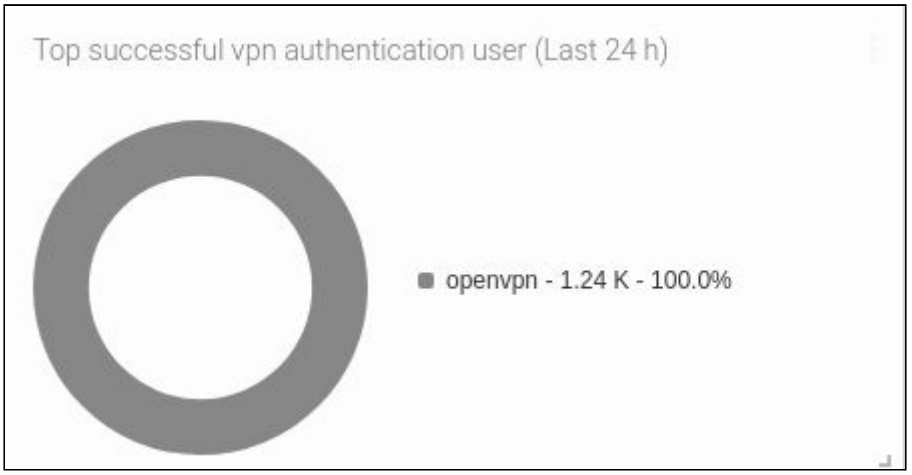
Top users Login (Last 24 h)

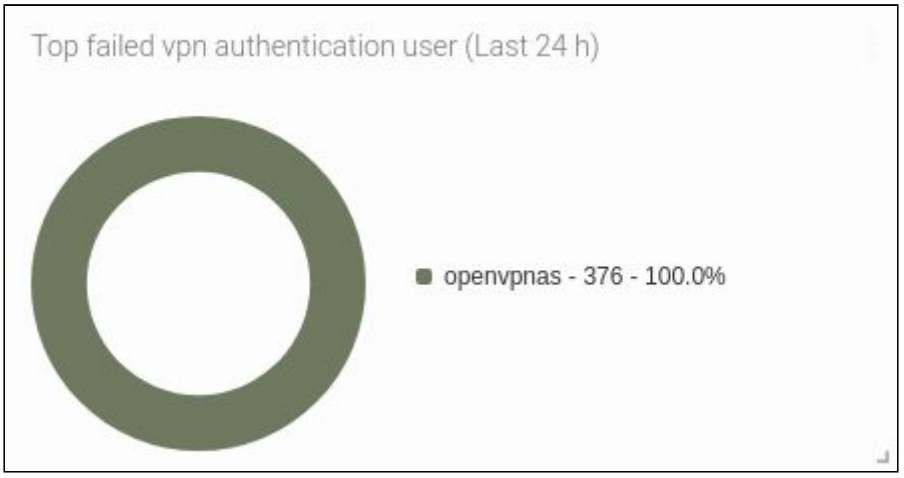
Juniper user	Juniper from ip	events
CABB\mamena(Empleados_DA)[Empleados, Administradores, Informatica, Marian]	37.10.159.219	1.07 K
CABB\wouter(Externos_DA)[Externos, Lims_laboratorio]	88.21.62.230	1.04 K

B.2. OpenVPN


Top vpn authentication (Last 24 h)
11.4 %
PAM auth succeeded - 418 - 23.5%
PAM auth failed - 406 - 22.9%
AUTH_FAILED (status=1) - 374 - 21.1%
connection rejected - 371 - 20.9%
SESSION_ID auth succeeded - 202 - 11.4
AUTH_FAILED,SESSION: Your session ha
SESSION_ID only allowed to be used by cl

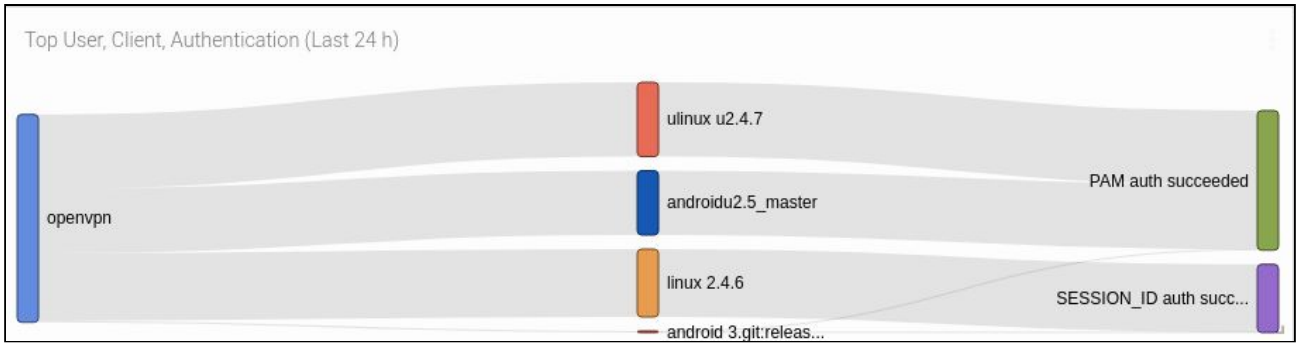

Top protocol used (Last 24 h)
TCP - 1.89 K - 100.0%

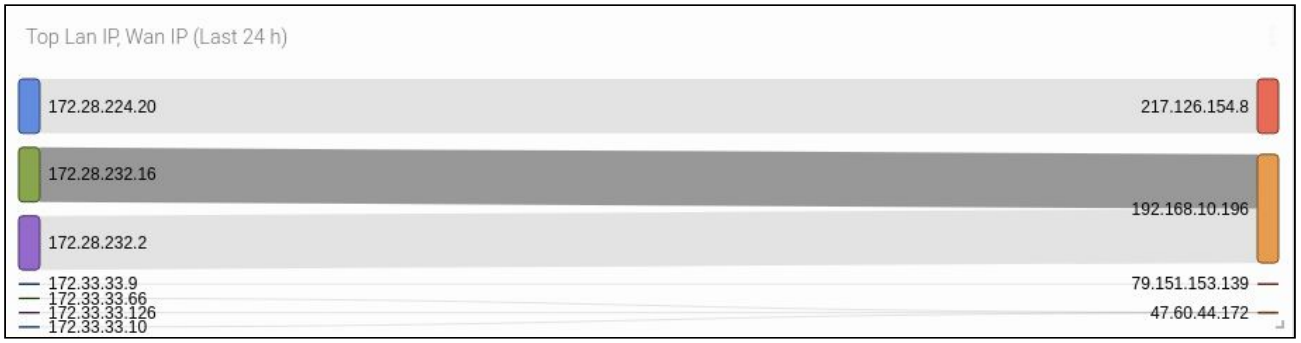

Top successful vpn authentication user (Last 24 h)
openvpn - 1.24 K - 100.0%



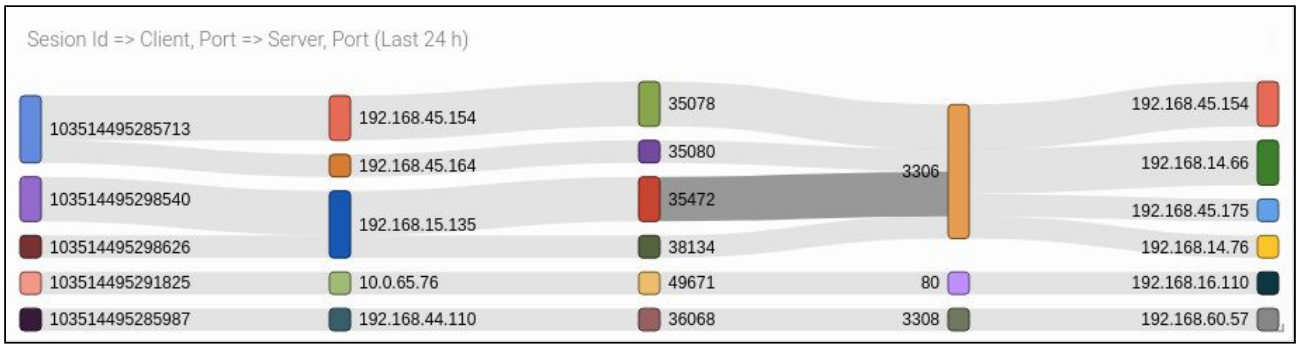
User list (Last 24 h)

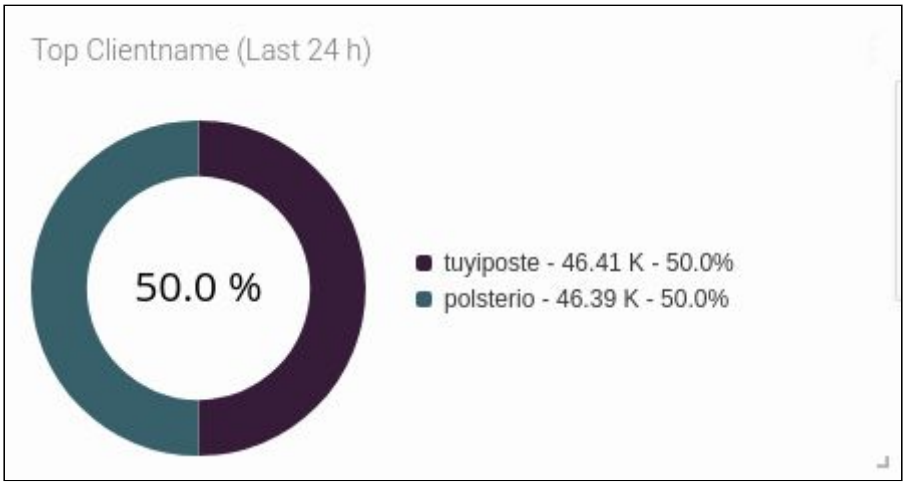
openvpn_user	events
openvpn	4.86 K
david	197

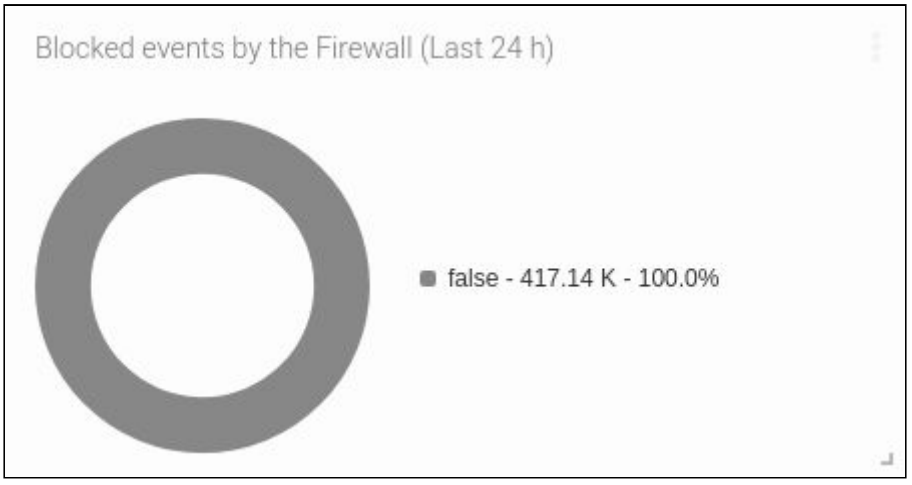


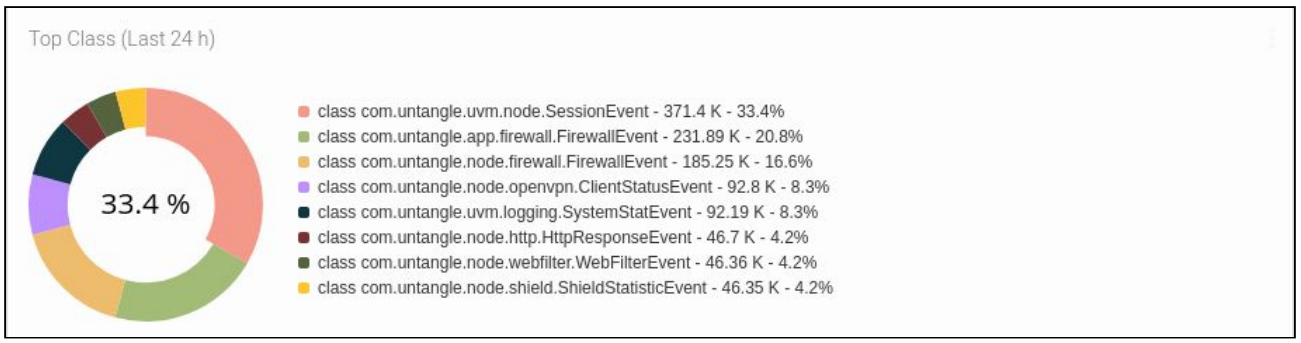


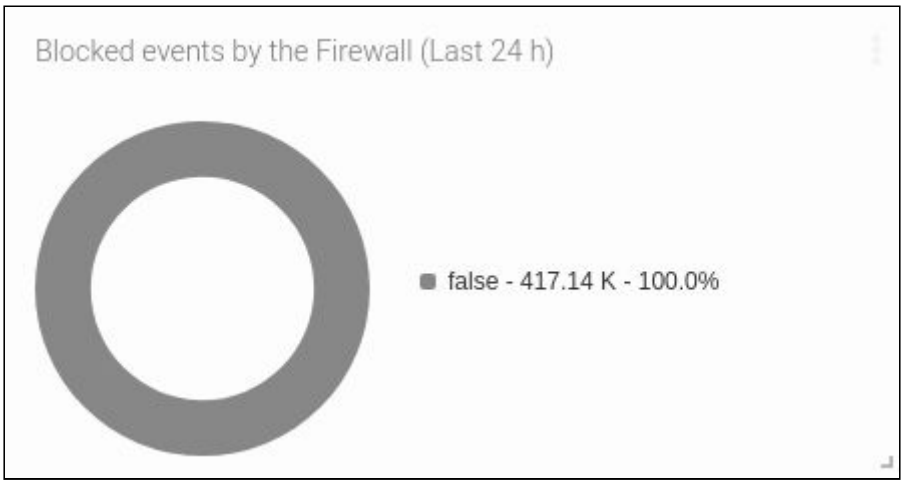
B.3. Untangle


Sesion Id => Client, Port => Server, Port (Last 24 h)
103514495285713
103514495298540
103514495298626
103514495291825
103514495285987
192.168.45.154
192.168.45.164
192.168.15.135
10.0.65.76
192.168.44.110
35078
35080
35472
38134
49671
36068
3306
80
3308
192.168.45.154
192.168.14.66
192.168.45.175
192.168.14.76
192.168.16.110
192.168.60.57


Top Clientname (Last 24 h)
50.0 %
tuyiposte - 46.41 K - 50.0%
polsterio - 46.39 K - 50.0%


Blocked events by the Firewall (Last 24 h)
false - 417.14 K - 100.0%

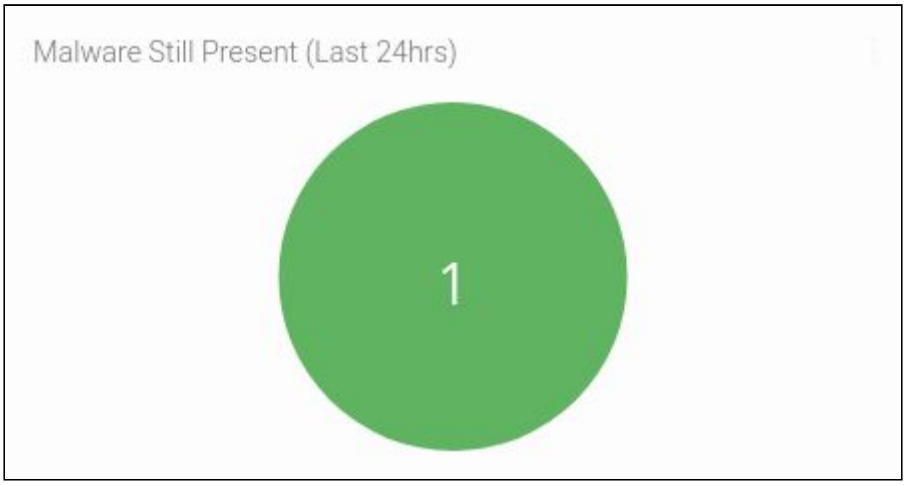


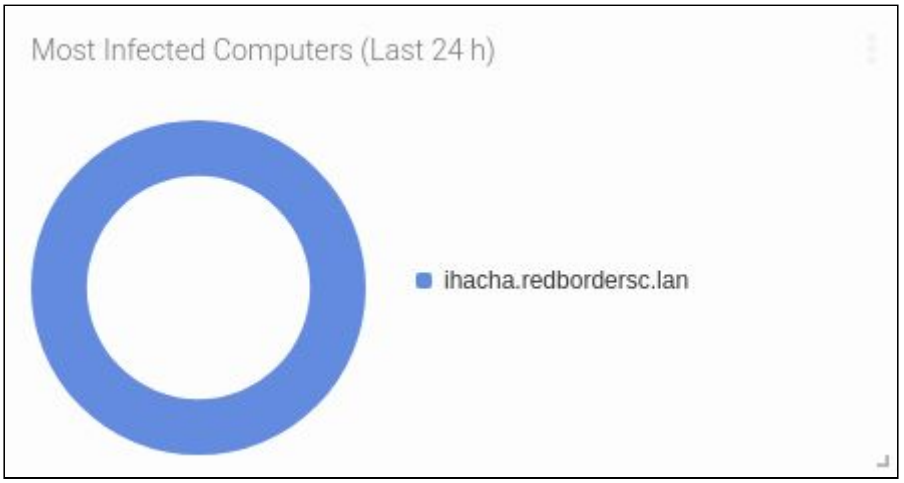


B.4. BitDefender GravityZone

Malware encounters (Last 24hrs)

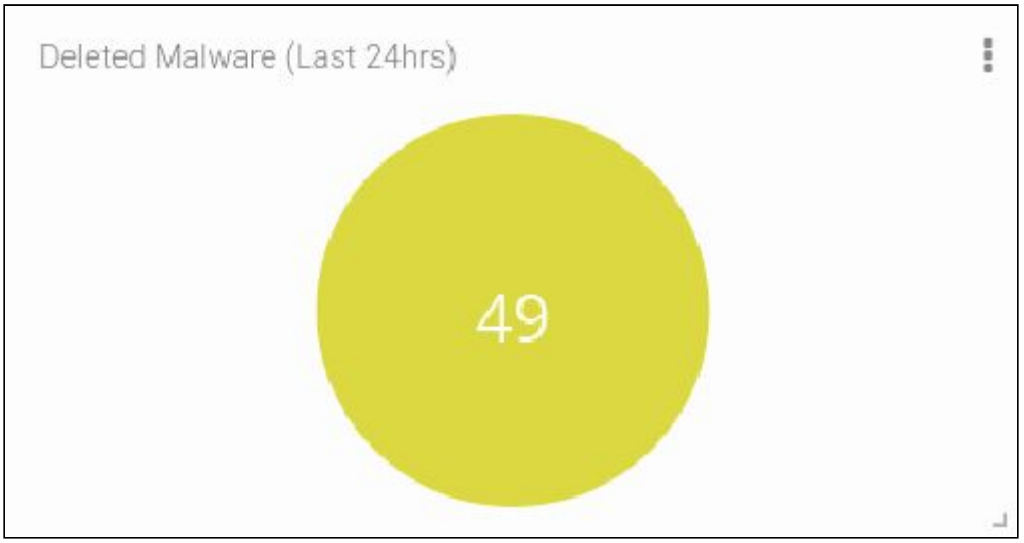
Bitdefender username	Bitdefender computer name	Bitdefender computer ip	Bitdefender malware name	Bitdefender malware type	events
root	ihacha.redbordersc.lan	192.168.10.196	Application.Isearch.Toolbar.C	file	37
root	ihacha.redbordersc.lan	192.168.10.196	Trojan.Generic.3130443	file	33
root	ihacha.redbordersc.lan	192.168.10.196	Packer.FSG.A	file	28
root	ihacha.redbordersc.lan	192.168.10.196	Dropped:Trojan.Bat.Secdrop.FW	file	18
root	ihacha.redbordersc.lan	192.168.10.196	Gen:Variant.Graftor.53584	file	18
root	ihacha.redbordersc.lan	192.168.10.196	Gen:Adware.Heur.mq1@Re@qaVbi	file	14



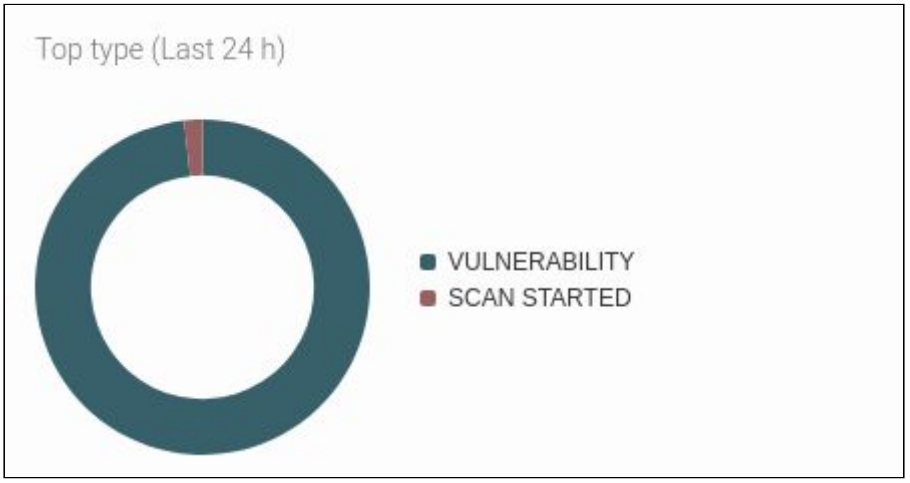


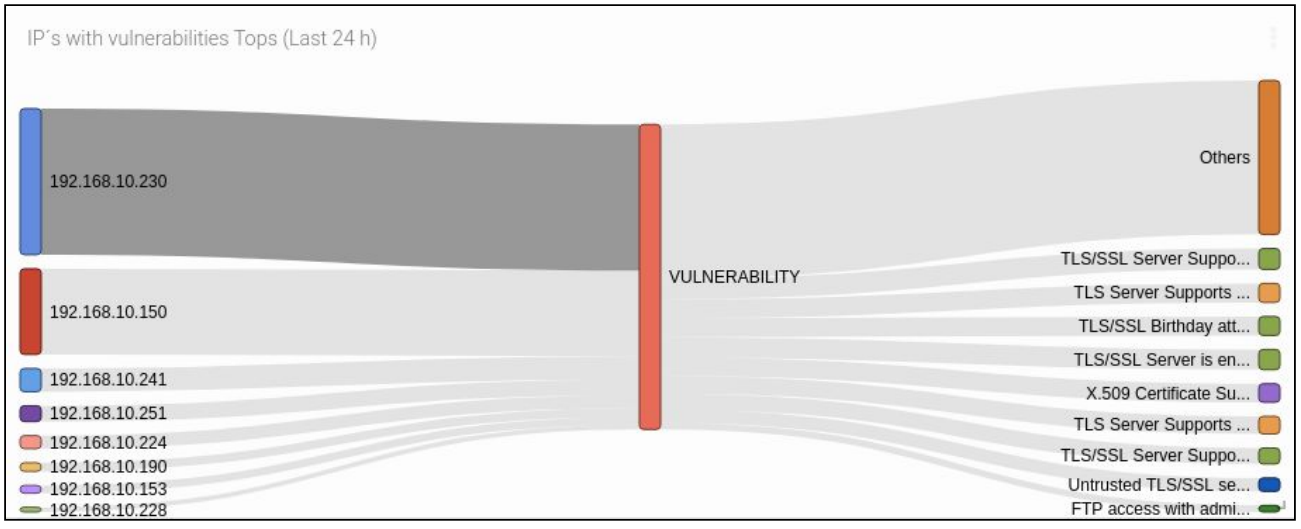
Suspicious Files (Last 24hrs)

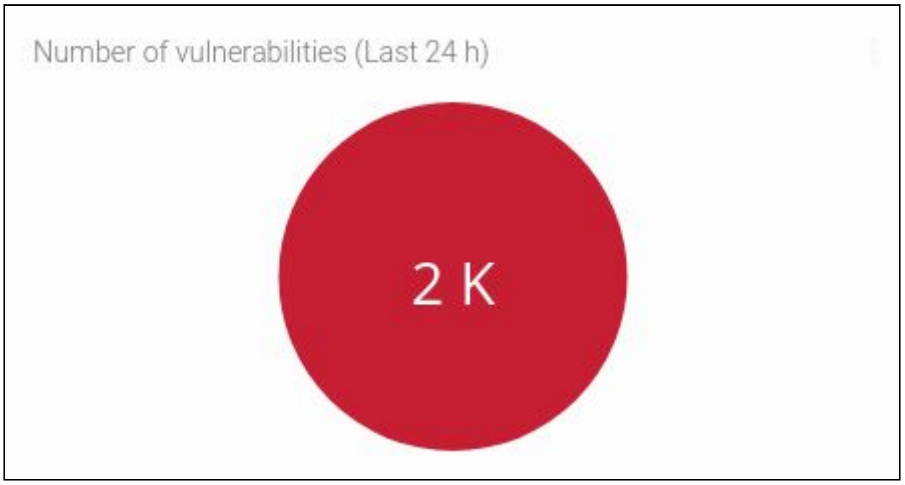
Bitdefender path	Bitdefender path arg.	Bitdefender status	events
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/APOLOXI_1144304324.0	(Quarantine-4)	deleted	7
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/CANARIAS-LAPTOP_1144652155.2	(Quarantine-4)	deleted	6
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/ELMO_1141286757.2	(Quarantine-4)	deleted	6
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/CANARIAS-LAPTOP_1144652146.0	(Quarantine-4)	deleted	6
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/CANARIAS-LAPTOP_1144652152.4	(Quarantine-4)	deleted	6
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/CANARIAS-LAPTOP_1144652152.6	(Quarantine-4)	deleted	5
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/CANARIAS-LAPTOP_1144652130.5	(Quarantine-4)	deleted	5
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/Virus/ELMO_1141286755.4=>(Quarantine-4)	BlackBox.class	deleted	5
/media/pandoro/blackbox/blackbox/General/Backup Disco Duro Tenerife/Carsa Jorge Galvu00e1n/Jorge Galvu00e1n Traspaso PC Nov. 2010/OTROS/Aplicaciones/PCCSRV/	N/A	deleted	5



B.5. Rapid7 InsightVM







Rapid7 message list (Last 24 h)

Rapid7 insightvm message	events
ICMP timestamp response (generic-icmp-timestamp)	274
X.509 Certificate Subject CN Does Not Match the Entity Name (certificate-common-name-mismatch)	195
TLS/SSL Server is enabling the BEAST attack (ssl-cve-2011-3389-beast)	194
TLS/SSL Server Supports The Use of Static Key Ciphers (ssl-static-key-ciphers)	188
TLS/SSL Birthday attacks on 64-bit block ciphers (SWEET32) (ssl-cve-2016-2183-sweet32)	184
TLS Server Supports TLS version 1.1 (tlsv1_1-enabled)	175

Apéndice C.
Creación reglas user friendly.

C.1. Routes

routes.rb

```
#############################################################################
# Copyright (c) 2014 ENEO Tecnologia S.L.
# This file is part of redBorder.
# redBorder is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# redBorder is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
# You should have received a copy of the GNU Affero General Public
License
# along with redBorder. If not, see <http://www.gnu.org/licenses/>.
#############################################################################

RedBorder::Application.routes.draw do
  DittoCode::Exec.if 'ENTERPRISE' do

    resources :correlation_engine_rules do
      post :apply, on: :collection
      get :new_rule, on: :collection
      get :basic_rule_2, on: :collection
      get :add_conditions, on: :collection
      post :custom_rule, on: :collection
      post :create_basic_rule_2, on: :collection
    end
```

```
  end
end
```

C.2. Controlador

app/controllers/correlation_engine_rule.rb

```
DittoCode::HoldFile.if 'ENTERPRISE'

################################################################################
# Copyright (c) 2014 ENEO Tecnologia S.L.
# This file is part of redBorder.
# redBorder is free software: you can redistribute it and/or modify
# it under the terms of the GNU Affero General Public License as
published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
# redBorder is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU Affero General Public License for more details.
# You should have received a copy of the GNU Affero General Public
License
# along with redBorder. If not, see <http://www.gnu.org/licenses/>.
################################################################################

class CorrelationEngineRulesController < ApplicationController
  ##
=====================================================================
  # => Before and after
  ##
=====================================================================

  before_action :require_super_administrative_privileges
  before_action :index, only: [:create, :create_basic_rule2] # so we
can access @cep_status from create method.
  before_action :load_form_settings, only: [:basic_rule_2]
  before_action :create_topic_filters_and_selects, only:
[:create_basic_rule_2]

  ##
```

```
=============================================================================
  # => Actions
  ##
=============================================================================

  def index
    @rules = CorrelationEngineRule.all.order(name: :asc)
    @cep_status = ''
    @managers = Manager.all.sort_by(&:name) # we could have a cluster,
with multiple managers
    @managers.each do |manager|
      if manager.services['cep'][:status] == true
           @cep_status = true
           break
      else
           @cep_status = false
      end
    end
  end

  def new
    render layout: false
  end

  def edit
    @rule = CorrelationEngineRule.find(params[:id])
    render layout: false
  end

  def create
    @rule = CorrelationEngineRule.new(rule_params)
    @rule.output = output_params
    @rule.user_id = current_user.idhola
    Rails.logger.error "\n\n\n rule es #{@rule.to_json}"

    respond_to do |format|
      if @rule.save
        format.html { redirect_to correlation_engine_rules_path(value:
@cep_status), notice: 'Rule was successfully created.' }
        format.js { flash[:notice] = 'Rule was successfully created.' }
        format.json { head :no_content }
```

```
      else
        format.html do
          redirect_to correlation_engine_rules_path(@rule),
                      alert: @rule.errors.full_messages.map { |msg|
"<li>#{msg}</li>" }.join
        end
        format.js
        format.json { render json: @rule.errors, status:
:unprocessable_entity }
      end
    end
  end

  def update
    @rule = CorrelationEngineRule.find(params[:id])
    @rule.update(rule_params)
  end

  def destroy
    @rule = CorrelationEngineRule.find(params[:id])
    @rule.destroy
  end

  def new_rule
    render layout: false
  end

  def load_form_settings
    @intrusion_values = []
    @vault_values = []
    @monitor_values = []
    @traffic_values = []
    @aux = []

    # Getting name of dimensions from Intrusion
    Redborder::Module.get(:ips, :columns).values.each { |value|
      if value.key?(:name)
        @intrusion_values << value[:name]
      else
       @intrusion_values << value[:columns].values[0]
      end
```

```
    }

    # Getting name of dimensions from Monitor
    Redborder::Module.get(:monitor, :tabs).values.each { |value|
     @monitor_values << value[:name]
    }

    # Getting name of dimensions from Vault
    Vault::COLUMNS.values.each{ |value|
     if value.key?(:name)
       @vault_values << value[:name]
     else
      @vault_values << value[:columns].values[0]
     end
    }

    # Getting name of dimensions from Traffic
    Traffic::COLUMNS.values.each{ |value|
      if value.key?(:name)
        @traffic_values << value[:name]
      else
       @traffic_values << value[:columns].values[0]
      end
    }

    @traffic_aggregations = Redborder::Module.get(:flow,
:aggregations).map {|k,v| k.to_s}.zip(Redborder::Module.get(:flow,
:aggregations).map {|k,v| k.to_s})
    @vault_aggregations = Redborder::Module.get(:vault,
:aggregations).map {|k,v| k.to_s}.zip(Redborder::Module.get(:vault,
:aggregations).map {|k,v| k.to_s})
    @intrusion_aggregations = Redborder::Module.get(:ips,
:aggregations).map {|k,v| k.to_s}.zip(Redborder::Module.get(:ips,
:aggregations).map {|k,v| k.to_s})
    @monitor_aggregations =  Redborder::Module.get(:monitor,
:aggregations).map {|k,v| k.to_s}.zip(Redborder::Module.get(:monitor,
:aggregations).map {|k,v| k.to_s})

    # Building Arrays like [["Lan IP", "lan_ip"]]
    @vault_columns = (@vault_values.zip(Vault::COLUMNS.keys.map {|k|
k.to_s}) + [["Vault Timestamp", "timestamp"]]).sort
```

```
    @traffic_columns = (@traffic_values.zip(Traffic::COLUMNS.keys.map
{|k| k.to_s}) + [["Traffic Timestamp", "timestamp"]]).sort
    @monitor_columns =
@monitor_values.zip(Redborder::Module.get(:monitor, :tabs).keys.map
{|k| k.to_s}).sort
    @intrusion_columns =
@intrusion_values.zip(Redborder::Module.get(:ips, :columns).keys.map
{|k| k.to_s}).sort

    # Since Traffic is always referenced as F and Vault as V, prepend
that string in order to make the selections
    @traffic_vault = @traffic_columns.map{|k| [k[0], "F.#{k[1]}"]} +
@vault_columns.map{|k| [k[0], "V.#{k[1]}"]}
    @traffic_vault_aggregations = @traffic_aggregations.map{|k| [k[0],
"F.#{k[1]}"]} + @vault_aggregations.map{|k| [k[0], "V.#{k[1]}"]}

    @functions = [["None", ""],["Average", "avg"],["Summation",
"sum"],["Maximum", "max"],["Minimum", "min"],["Count",
"count"],["Distinct Count", "distinctCount"],["Max Forever",
"maxForever"],["Min Forever","minForever"],["Standard
Deviation","stddev"]]
    @operator = [["less than", "<"], ["less or equal than", "<="],
["greater than", ">"], ["greater or equal than", "=>"], ["equal",
"=="], ["not equal to", "!="], ["contains", "contains"], ["is", "is"]]
    @window = [["Messages from last X mins", "window.time"], ["Messages
from last X mins as Batch", "window.timeBatch"], ["Last X messages",
"window.length"], ["Last X messages as Batch", "window.lengthBatch"]]
    @inputs = [["Traffic", "rb_flow_post"],["Intrusion",
"rb_event_post"],["Monitor", "rb_monitor"], ["Vault", "rb_vault_post"]]
  end

  def basic_rule_2
    render layout: false
  end

  def create_topic_filters_and_selects
    @condition1 = ""
    @condition2 = ""
    @join = ""
    @select = ""
```

```
    topic1_parameters = params[:correlation_engine_rule].select { |key,
value| key.to_s.match(/^topic1_filter\d+/)}
    topic2_parameters = params[:correlation_engine_rule].select { |key,
value| key.to_s.match(/^topic2_filter\d+/)}
    select_parameters = params[:correlation_engine_rule].select { |key,
value| key.to_s.match(/^select\d+/)}

    i = 0
    while (i < topic1_parameters.size) do
      var_not = "topic1_not_filter" + i.to_s
      var_filter = "topic1_filter" + i.to_s
      var_operator = "topic1_filter_operator" + i.to_s
      @condition1 << "not (" if rule_params[var_not.to_sym] == "1"
      @condition1 << "#{@join}#{rule_params[var_filter.to_sym]}
#{rule_params[var_operator.to_sym]}
#{rule_params[:topic1_filter_value][i]} "
      @condition1 << ")" if rule_params[var_not.to_sym] == "1"
      @join = "and "
      i += 1
    end
    #Rails.logger.error "\n\n\n @condition1 es #{@condition1}"

    i = 0
    @join = ""
    while (i < topic2_parameters.size) do
      var_not = "topic2_not_filter" + i.to_s
      var_filter = "topic2_filter" + i.to_s
      var_operator = "topic2_filter_operator" + i.to_s
      @condition2 << "not (" if rule_params[var_not.to_sym] == "1"
      @condition2 << "#{@join}#{rule_params[var_filter.to_sym]}
#{rule_params[var_operator.to_sym]}
#{rule_params[:topic2_filter_value][i]}"
      @condition2 << ")" if rule_params[var_not.to_sym] == "1"
      @join = "and "
      i += 1
    end
    #Rails.logger.error "\n\n\n @condition2 es #{@condition2}"

    i = 1
    @join = ""
    while (i < select_parameters.size) do
```

```
      var_function = "select_function" + i.to_s
      var_select = "select" + i.to_s
      var_name = "select_name" + i.to_s
      @select << @join
      @select << "#{rule_params[var_function.to_sym]}(" if
!rule_params[var_function.to_sym].empty?
      @select << "#{rule_params[var_select.to_sym]}"
      @select << " as #{rule_params[var_name.to_sym]}" if
!rule_params[var_name.to_sym].empty?
      @select << ")" if !rule_params[var_function.to_sym].empty?
      @join = ","
      i += 1
    end
    #Rails.logger.error "\n\n\n select es #{@select}"
  end

  def create_basic_rule_2
    #Rails.logger.error "\n\n\n estamos en create_basic_rule_2 con
params: #{rule_params.to_json} y params #{params.to_json}"
    @input = ["#{rule_params[:topic1]}", "#{rule_params[:topic2]}"]
    @output = "{\"outputStream\":\"#{rule_params[:output]}\"}"
    @execution_plan = "from #{rule_params[:topic1]}#{[ @condition1
]}##{rule_params[:window_type]}(#{rule_params[:window_value]}) as F\
                       \rjoin\
                       \r#{rule_params[:topic2]}#{[ @condition2
]}##{rule_params[:window_type2]}(#{rule_params[:window_value2]}) as V\
                       \ron F.#{rule_params[:join_attr]}
#{rule_params[:join_type]} V.#{rule_params[:join_attr2]}\
                       \rselect #{@select}\
                       \rinsert into outputStream;".gsub('"', '')
    Rails.logger.error "\n\n\n execution_plan es #{@execution_plan}"

    @rule = CorrelationEngineRule.new(name: rule_params[:name],
                                      execution_plan: @execution_plan,
                                      input: @input)
    @rule.output = @output
    @rule.user_id = current_user.id

    respond_to do |format|
      if @rule.save
        format.html { redirect_to correlation_engine_rules_path(value:
```

```
@cep_status), notice: 'Rule was successfully created.' }
      else
        format.html do
          redirect_to correlation_engine_rules_path(@rule),
                      alert: @rule.errors.full_messages.map { |msg|
"<li>#{msg}</li>" }.join
        end
      end
    end
  end

  #
  # This method synchronizes the rules table with remote correlation
engine,
  # using given API REST
  #

  def apply
    rules = CorrelationEngineRule.where(enabled: true)
    rules_ = []
    correlation_config =
YAML.load_file("#{Rails.root}/config/correlation.yml")[Rails.env]

    rules.each do |rule|
      rules_.push(
        id: rule.id.to_s,
        input: rule.input,
        output: rule.output,
        executionPlan: rule.execution_plan,
        version: rule.updated_at.to_i
      )
    end

    http = Net::HTTP.new(correlation_config['host'],
correlation_config['port'])
    request = Net::HTTP::Post.new('/v1/synchronize')
    request.add_field('Content-Type', 'application/json')
    request.body = rules_.to_json
    begin
      @response = http.request(request)
    rescue Timeout::Error, Errno::EINVAL, Errno::ECONNRESET, EOFError,
```

```
          Net::HTTPBadResponse, Net::HTTPHeaderSyntaxError,
Net::ProtocolError,
          Errno::ECONNREFUSED => e
      Rails.logger.error e
      Rails.logger.error e.backtrace.join("\n")
    end
  end

  private

  #
  # This method builds a hash from the output stream-topic array
  #

  def output_params
    output = {}
    if params[:correlation_engine_rule][:stream].size ==
params[:correlation_engine_rule][:topic].size
      params[:correlation_engine_rule][:stream].each_with_index do |el,
i|
        output[el] = params[:correlation_engine_rule][:topic][i]
      end
    end
    output
  end

  def rule_params
    params.require(:correlation_engine_rule)
      .permit(:name, :execution_plan, :enabled, :output, :topic1,
:window_type, :window_value,
              :topic1_not_filter0, :topic1_not_filter2,
:topic1_not_filter3, :topic1_not_filter4,
              :topic2_not_filter0, :topic2_not_filter2,
:topic2_not_filter3, :topic2_not_filter4, :window_type, :window_value2,
              :topic1_filter0, :topic1_filter1, :topic1_filter2,
:topic1_filter3, :window_type2,
              :topic1_filter_operator0, :topic1_filter_operator1,
:topic1_filter_operator2, :topic1_filter_operator3,
              :topic2, :topic2_not_filter, :topic2_not_filter1,
:topic2_not_filter2, :topic2_not_filter3,
              :topic2_filter0, :topic2_filter1, :topic2_filter2,
```

```
:topic2_filter3,
              :topic2_filter_operator0, :topic2_filter_operator1,
:topic2_filter_operator2, :topic2_filter_operator3,
              :join_attr, :join_attr2, :join_type, :select1, :select2,
:select3, :select4, :select5, :select6, :select7,:select_name1,
              :select_name2, :select_name3, :select_name4,
:select_name5, :select_name6, :select_name7, :select_function1,
              :select_function2, :select_function3, :select_function4,
:select_function5, :select_function6, :select_function7,
              topic1_filter_value: [], topic1_filter_value:
[],topic2_filter_value: [], input: [])
  end
end
```

C.3. Javascript

app/assets/javascript/correlation_engine_rule.js

```
/*!
 * Copyright (c) 2014 ENEO Tecnología S.L.
 * This file is part of redBorder.
 * redBorder is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as
published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 * redBorder is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Affero General Public License for more details.
 * You should have received a copy of the GNU Affero General Public
License
 * along with redBorder. If not, see <http://www.gnu.org/licenses/>.
 */

$(document).on('change', '#correlation_engine_rules .enabled
input:checkbox', function () {
  var id = $(this).parent().data('correlation_engine_rule');
  var enabled = $(this).prop('checked');
  $.ajax({
```

```
    url: "/correlation_engine_rules/" + id,
    remote: true,
    type: 'PATCH',
    data: {
      correlation_engine_rule: {
        enabled: enabled
      }
    }
  });
});

$(document).on("click", ".add-new-condition", function(e) {
  index++;
  var html =
$('.conditions').clone().removeClass("conditions").addClass("conditions
" + index);
  $(".extra-conditions").append(html);
  $('.remove-condition').first().removeClass('disabled');
  e.preventDefault();
});

$(document).on("click", ".remove-condition", function(e) {
  $(".conditions" + index).remove();
  index--;
  if (index == 0){
    $('.remove-condition').first().addClass('disabled');
  }
  e.preventDefault();
});

$(document).on("click", ".correlation_engine_rules_output
.add-new-rule", function(e) {
  var html =
  "<div class='row'>\
    <div class='col-md-5'>\
      <div class='form-group string required
correlation_engine_rules_stream'>\
        <input class='string required form-control'
id='correlation_engine_rules_stream'\
          name='correlation_engine_rules[stream][]'
placeholder='Stream' type='text'>\
```

```
      </div>\
    </div>\
    <div class='col-md-5'>\
      <div class='form-group string required
correlation_engine_rules_topic'>\
        <input class='string required form-control'
id='correlation_engine_rules_topic'\
          name='correlation_engine_rules[topic][]' placeholder='Topic'
type='text'>\
      </div>\
    </div>\
    <div class='col-md-2'>\
      <a class='btn btn-small add-new-rule' href='#' style='color:
green'><i class='fa fa-plus'></i></a>\
      <a class='btn btn-small remove-output' href='#' style='color:
red'><i class='fa fa-minus'></i></a>\
    </div>\
  </div>";
  $('.correlation_engine_rules_output').append(html);
  e.preventDefault();
  $('.correlation_engine_rules_output
.remove-output').first().removeClass('disabled');
});

$(document).on("click", ".correlation_engine_rules_output
.remove-output", function(e) {
  $(this).parent().parent().remove()
  if ($('.correlation_engine_rules_output .remove-output').length ===
1){
    $('.correlation_engine_rules_output
.remove-output').first().addClass('disabled');
  }
  e.preventDefault();
});

// $(document).on("click", ".add-new-topic", function(e) {if (index ==
0){
//     $('.remove-condition').first().addClass('disabled');
//   }
//   $('.correlation_engine_rules_query .first-ref').css('display',
'block');
```

```
//   $('.correlation_engine_rules_query .first-ref
label').addClass('required');
//   $('.correlation_engine_rules_query .first-ref
input').addClass('required');
//   $('.where-field').css('display', 'block');
//   $('.correlation_engine_rules_query .empty-element').css('display',
'none');
//   $('.correlation_engine_rules_query2').css('display', 'block');
//   $('.select-field input').attr('placeholder', 'V.app_name as
custom_app_name, avg(F.bytes) as avg_bytes');
//   $('.remove-topic').removeClass('disabled');
//
//   $('.correlation_engine_rules_query2 input').each(function(){
//     $(this).attr('required', 'true');
//   });
//
//   e.preventDefault();
// });
//
// $(document).on("click", ".remove-topic", function(e) {
//
//   $('.correlation_engine_rules_query2').css('display', 'none');
//   $('.correlation_engine_rules_query .first-ref').css('display',
'none');
//   $('.where-field').css('display', 'none');
//   $('.select-field input').attr('placeholder', 'lan_ip as
user_lan_ip');
//   $('.remove-topic').addClass('disabled');
// });
//
// $(document).on("click",
"#correlation_engine_rule_input_rb_vault_post", function(e){
//   $('.cep_filter').css('display', 'block');
//   $('.vault-columns').css('display', 'block');
//   $('.traffic-columns').css('display', 'none');
//   $('.intrusion-columns').css('display', 'none');
//   $('.monitor-columns').css('display',
//   'none');
// });
//
// $(document).on("click",
```

```
"#correlation_engine_rule_input_rb_flow_post", function(e){
//   $('.cep_filter').css('display', 'block');
//   $('.traffic-columns').css('display', 'block');
//   $('.vault-columns').css('display', 'none');
//   $('.intrusion-columns').css('display', 'none');
//   $('.monitor-columnsindex--;').css('display', 'none');
// });
//
// $(document).on("click",
"#correlation_engine_rule_input_rb_event_post", function(e){
//   $('.cep_filter').css('display', 'block');
//   $('.intrusion-columns').css('display', 'block');
//   $('.vault-columns').css('display', 'none');
//   $('.traffic-columns').css('display', 'none');
//   $('.monitor-columns').css('display', 'none');
// });
//
// $(document).on("click", "#correlation_engine_rule_input_rb_monitor",
function(e){
//   $('.cep_filter').css('display', 'block');
//   $('.monitor-columns').css('display', 'block');
//   $('.vault-columns').css('display', 'none');
//   $('.traffic-columns').css('display', 'none');
//   $('.intrusion-columns').css('display', 'none');
// });

var filters_1 = 0; //number of filters for topic1
var filters_2 = 0; //number of filters for topic2
var selects = 1; //number of selects

// For the first Topic
$(document).on("click", ".add-cep-filters", function(e){
  filters_1 = 1;
  $('.first-filters').find('input, select, label').attr("disabled",
false).removeClass("disabled");
  $('.btn-add-filters').css('display','none');
  $('.cep_filter').css('display','block');
  $('.btn-add-more-filters').css('display', 'block');
  e.preventDefault();
});
```

```
$(document).on("click", ".add-new-filter", function(e){
  filters_1++;
  var html =
$('.first-filters').clone().removeClass("first-filters").addClass("firs
t-filters" + (filters_1-1));

html.find("input[name='correlation_engine_rule[topic1_not_filter0]']").
attr('name', 'correlation_engine_rule[topic1_not_filter' +
(filters_1-1) + ']');

html.find("select[name='correlation_engine_rule[topic1_filter0]']").att
r('name', 'correlation_engine_rule[topic1_filter' + (filters_1-1) +
']');

html.find("select[name='correlation_engine_rule[topic1_filter_operator0
]']").attr('name', 'correlation_engine_rule[topic1_filter_operator' +
(filters_1-1) + ']');
  $('.cep_filter').append(html);
  $('.remove-filter').removeClass("disabled");
  e.preventDefault();
});

$(document).on("click", ".remove-filter", function(e){
  if (filters_1 == 1){
    $('.cep_filter').css('display', 'none');
  }
  else{
    $('.first-filters' + (filters_1-1)).remove();
  }
  filters_1--;
  // If we deleted all filters
  if (filters_1 == 0){
    $('.first-filters').find('input, select, label').attr("disabled",
'disabled').addClass("disabled");
    $('.btn-add-more-filters').css('display', 'none');
    $('.cep_filter').css('display', 'none');
    $('.btn-add-filters').css('display', 'block');
  }
  e.preventDefault();
});
```

```
// For the second Topic
$(document).on("click", ".add-cep-filters2", function(e){
  filters_2 = 1;
  $('.second-filters').find('input, select, label').attr("disabled",
false).removeClass("disabled");;
  $('.btn-add-filters2').css('display','none');
  $('.cep_filter2').css('display','block');
  $('.btn-add-more-filters2').css('display', 'block');
  e.preventDefault();
});

$(document).on("click", ".add-new-filter2", function(e){
  filters_2++;
  var html =
$('.second-filters').clone().removeClass("second-filters").addClass("se
cond-filters" + (filters_2-1));

html.find("input[name='correlation_engine_rule[topic2_not_filter0]']").
attr('name', 'correlation_engine_rule[topic2_not_filter' +
(filters_2-1) + ']');

html.find("select[name='correlation_engine_rule[topic2_filter0]']").att
r('name', 'correlation_engine_rule[topic2_filter' + (filters_2-1) +
']');

html.find("select[name='correlation_engine_rule[topic2_filter_operator0
]']").attr('name', 'correlation_engine_rule[topic2_filter_operator' +
(filters_2-1) + ']');
  $('.cep_filter2').append(html);
  $('.remove-filter2').removeClass("disabled");
  e.preventDefault();
});

$(document).on("click", ".remove-filter2", function(e){
  if (filters_2 == 1){
    $('.cep_filter2').css('display', 'none');
  }
  else{
    $('.second-filters' + (filters_2-1)).remove();
  }
  filters_2--;
```

```
  // If we deleted all filters
  if (filters_2 == 0){
    $('.second-filters').find('input, select, label').attr("disabled",
'disabled').removeClass("disabled");
    $('.btn-add-more-filters2').css('display', 'none');
    $('.cep_filter2').css('display', 'none');
    $('.btn-add-filters2').css('display', 'block');
  }
  e.preventDefault();
});

$(document).on("click", ".add-new-select", function(e){
  selects++;
  $('.remove-select').removeClass("disabled");
  var html =
$('.select-to-be-cloned').clone().removeClass("select-to-be-cloned").ad
dClass("select-to-be-cloned" + selects);

html.find("select[name='correlation_engine_rule[select1]']").attr('name
', 'correlation_engine_rule[select' + selects + ']');

html.find("select[name='correlation_engine_rule[select_function1]']").a
ttr('name', 'correlation_engine_rule[select_function' + selects + ']');

html.find("input[name='correlation_engine_rule[select_name1]']").attr('
name', 'correlation_engine_rule[select_name' + selects + ']');
  $('.column-selection').append(html);
  e.preventDefault();
});

$(document).on("click", ".remove-select", function(e){
  if (selects == 2){
    $('.remove-select').addClass("disabled");
    $('.select-to-be-cloned' + selects).remove();
    selects--;
  }
  else{
    $('.select-to-be-cloned' + selects).remove();
    selects--;
  }
  e.preventDefault();});
```

C.4. Vistas.

/views/correlation_engine_rules/basic_rule_2.html.erb

```
<div class="modal-header">
  <button type="button" class="close" data-dismiss="modal"><span
aria-hidden="true">&times;</span><span
class="sr-only">Close</span></button>
  <h3 class="modal-title">Create Basic Rule with 2 topics</h3>
</div>
<%= simple_form_for(:correlation_engine_rule, url:
create_basic_rule_2_correlation_engine_rules_path, :html => { :remote
=> true, :method => :post }) do |f| %>
  <div class="modal-body">
    <div class="panel panel-default">
      <div class="panel-body">
        <div class="form-group string row">
          <div class="col-md-5">
            <%= f.input :name, label: "Name",input_html: {type: "text"
} %>
          </div>
          <div class="col-md-5">
            <%= f.input :output, label: "Output Topic",input_html:
{type: "text" } %>
          </div>
        </div>
        <!-- <div class="form-group string required
correlation_engine_rules_output">
          <label class="required"><abbr title="required">*</abbr>
Output (*)</label>
          <div class="row">
            <div class="col-md-5">
              <%= f.input :stream, input_html: { type: "text",
placeholder: 'Stream', name: 'correlation_engine_rule[stream][]' },
label: false %>
            </div>
            <div class="col-md-5">
              <%= f.input :topica, input_html: { type: "text",
placeholder: 'Topic', name: 'correlation_engine_rule[topic][]'}, label:
false %>a
```

```
            </div>
            <div class="col-md-2">
              <%= link_to content_tag(:i, nil, class: "fa fa-plus"),
                 '', style: 'color: green', class: "btn btn-small
add-new-rule", 'data-backdrop' => "static" %>
              <%= link_to content_tag(:i, nil, class: "fa fa-minus"),
                 '', style: 'color: red', class: "btn btn-small
remove-output disabled", 'data-backdrop' => "static" %>
            </div>
          </div>
        </div> -->
      </div>
    </div>
    <div class="panel panel-default">
      <%= render partial: 'topic1', locals: { f: f } %>
    </div>
    <div class="panel panel-default">
      <%= render partial: 'topic2', locals: { f: f } %>
    </div>
    <div class="panel panel-default">
      <%= render partial: 'select', locals: { f: f } %>
    </div>
  </div>
  <div class="modal-footer">
    <%= f.submit "Create", :class => "btn btn-success" %>
    <%= link_to "Cancel", correlation_engine_rules_path, :class => "btn
btn-secondary", :'data-dismiss' => "modal" %>
  </div>
<% end %>
```

/views/correlation_engine_rules/_topic1.html.erb

```
<div class="panel-heading">
  <div class="panel-title">First Topic</div>
</div>
<div class="panel-body">
  <div class="row">
    <div class="col-md-6">
      <%= f.input :topic1, collection: @inputs,include_blank:
false,label: "First Input To Read From (Referenced as F)",input_html:
{type: "text" } %>
    </div>
    <div class="col-md-4">
      <%= f.input :window_type, collection: @window,include_blank:
false, label: "Window Type", input_html: {type: "text"} %>
    </div>
    <div class="col-md-2">
      <%= f.input :window_value, label: "Window Value", placeholder:
"X", input_html: {type: "text"} %>
    </div>
  </div>
  <div class="btn-add-filters row col-md-2">
    <%= link_to "Add Filters", "#", :class => "btn btn-success
add-cep-filters", :style =>"width:auto !important;" %>
  </div>
  <div class="cep_filter row" style="display:none">
    <div class="first-filters">
      <div class="col-md-1">
        <%= f.label :label, "NOT"  %>
        <br>
        <%= f.check_box :topic1_not_filter0, disabled: true %>
      </div>
      <div class="col-md-2 traffic-columns">
        <%= f.input :topic1_filter0, collection:
@traffic_columns,disabled: true,include_blank: false, required: false,
label: "Traffic Dimensions",  input_html: {type: "text" } %>
      </div>
      <!-- <div class="col-md-2 vault-columns" style="display:none">
        <%= f.input :topic1_filter, collection:
@vault_columns,disabled: true,include_blank: false,label: "Monitor
Dimensions",disabled: true, required: false, input_html: {type: "text"
```

```
} %>
      </div>
      <div class="col-md-2 monitor-columns" style="display:none">
        <%= f.input :topic1_filter, collection:
@monitor_columns,disabled: true,include_blank: false,required: false,
disabled: true,label: "Monitor Dimensions",input_html: {type: "text" }
%>
      </div>
      <div class="col-md-2 intrusion-columns" style="display:none">
        <%= f.input :topic1_filter, collection:
@intrusion_columns,disabled: true,include_blank: false, required:
false,disabled: true, label: "Intrusion Dimensions",input_html: {type:
"text" } %>
      </div> -->
      <div class="col-md-1">
        <%= f.input :topic1_filter_operator0, collection:
@operator,include_blank: false,disabled: true,label: "Operator" %>
      </div>
      <div class="col-md-2">
        <%= f.input :topic1_filter_value, label: "Value",disabled:
true,input_html: {multiple: true,type: "text" } %>
      </div>
    </div>
  </div>
  <div class="form-group string row btn-add-more-filters"
style="display:none">
    <div class="col-md-2">
      <%= link_to content_tag(:i, nil, class: "fa fa-plus"),
         '', style: 'color: green', class: "btn btn-small
add-new-filter", 'data-backdrop' => "static" %>
      <%= link_to content_tag(:i, nil, class: "fa fa-minus"),
         '', style: 'color: red', class: "btn btn-small remove-filter",
'data-backdrop' => "static" %>
    </div>
  </div>
</div>
```

/views/correlation_engine_rules/_topic2.html.erb

```
<div class="panel-heading">
  <div class="panel-title">Second Topic</div>
</div>
<div class="panel-body">
  <div class="row">
    <div class="col-md-6">
      <%= f.input :topic2, collection: @inputs, include_blank:
false,label: "Second Input To Read From (Referenced as V)",input_html:
{type: "text" } %>
    </div>
    <div class="col-md-4">
      <%= f.input :window_type2, collection: @window,label: "Window
Type", include_blank: false,input_html: {type: "text"} %>
    </div>
    <div class="col-md-2">
      <%= f.input :window_value2, label: "Window Value", input_html:
{type: "text"} %>
    </div>
  </div>
  <div class="btn-add-filters2 row col-md-2">
    <%= link_to "Add Filter", "#", :class => "btn btn-success
add-cep-filters2", :style =>"width:auto !important;" %>
  </div>
  <div class="cep_filter2 row" style="display:none">
    <div class="second-filters">
      <div class="col-md-1">
        <%= f.label :label, "NOT"  %>
        <br>
        <%= f.check_box :topic2_not_filter0, disabled: true %>
      </div>
      <!-- <div class="col-md-2 traffic-columns2" style="display:none">
        <%= f.input :topic2_filter2, collection:
@traffic_columns,disabled: true,include_blank: false, label: "Traffic",
required: false %>
      </div> -->
      <div class="col-md-2 vault-columns2">
        <%= f.input :topic2_filter0, collection: @vault_columns,
disabled: true,include_blank: false,label: "Vault", required: false,
input_html: {type: "text" } %>
```

```
      </div>
      <!-- <div class="col-md-2 monitor-columns2" style="display:none">
        <%= f.input :topic2_filter2, collection:
@monitor_columns,disabled: true, include_blank: false,required: false,
label: "Monitor",input_html: {type: "text" } %>
      </div>
      <div class="col-md-2 intrusion-columns2" style="display:none">
        <%= f.input :topic2_filter2, collection:
@intrusion_columns,disabled: true,include_blank: false,required: false,
label: "Intrusion",input_html: {type: "text" } %>
      </div> -->
      <div class="col-md-1">
        <%= f.input :topic2_filter_operator0, collection:
@operator,disabled: true,include_blank: false, label: "Op" %>
      </div>
      <div class="col-md-2">
        <%= f.input :topic2_filter_value, label: "Value",required:
false,disabled: true, input_html: {multiple: true,type: "text" } %>
      </div>
    </div>
  </div>
  <div class="form-group string row btn-add-more-filters2"
style="display:none">
    <div class="col-md-2">
      <%= link_to content_tag(:i, nil, class: "fa fa-plus"),
         '', style: 'color: green', class: "btn btn-small
add-new-filter2", 'data-backdrop' => "static" %>
      <%= link_to content_tag(:i, nil, class: "fa fa-minus"),
         '', style: 'color: red', class: "btn btn-small remove-filter2
disabled", 'data-backdrop' => "static" %>
    </div>
  </div>
</div>
<div class="panel panel-default">
  <%= render partial: 'join_condition', locals: { f: f } %>
</div>
```

/views/correlation_engine_rules/_join_condition.html.erb

```
<div class="panel-heading">
  <div class="panel-title">Join Condition</div>
</div>
<div class="panel-body">
  <div class="row">
    <div class="col-md-3 traffic-columns-join">
      <%= f.input :join_attr, collection: @traffic_columns,
include_blank: false, label: "Traffic", input_html: {type: "text" } %>
    </div>
    <div class="col-md-3 vault-columns-join" style="display:none">
      <%= f.input :join_attr, collection: @vault_columns, disabled:
true,include_blank: false,label: "Vault", required: false, input_html:
{type: "text" } %>
    </div>
    <div class="col-md-3 monitor-columns-join" style="display:none">
      <%= f.input :join_attr, collection: @monitor_columns,disabled:
true,include_blank: false,required: false, label: "Monitor",input_html:
{type: "text" } %>
    </div>
    <div class="col-md-3 intrusion-columns-join" style="display:none">
      <%= f.input :join_attr, collection: @intrusion_columns,disabled:
true,include_blank: false,required: false, label:
"Intrusion",input_html: {type: "text" } %>
    </div>
    <div class="col-md-2">
      <%= f.input :join_type, collection: [["Equal To", "=="], ["Not
Equal To", "!="]], include_blank: false,label: "Operator", input_html:
{type: "text"} %>
    </div>
    <div class="col-md-3 traffic-columns-join2" style="display:none">
      <%= f.input :join_attr2, collection: @traffic_columns,label:
"Traffic",disabled: true, include_blank: false,required:
false,input_html: {type: "text" } %>
    </div>
    <div class="col-md-3 vault-columns">
      <%= f.input :join_attr2, collection: @vault_columns, label:
"Vault", include_blank: false,input_html: {type: "text" } %>
    </div>
    <div class="col-md-3 monitor-columns-join2" style="display:none">
```

```
      <%= f.input :join_attr2, collection:
@monitor_columns,include_blank: false,disabled: true, required: false,
label: "Monitor",input_html: {type: "text" } %>
    </div>
    <div class="col-md-3 intrusion-columns-join2" style="display:none">
      <%= f.input :join_attr2, collection: @intrusion_columns,disabled:
true, include_blank: false,required: false, label:
"Intrusion",input_html: {type: "text" } %>
    </div>
  </div>
</div>
```

/views/correlation_engine_rules/_join_condition.html.erb

```
<div class="panel-heading">
  <div class="panel-title">Selection of Columns</div>
</div>
<div class="panel-body">
  <div class="column-selection row">
    <div class="select-to-be-cloned">
      <div class="col-md-2">
        <%= f.input :select_function1, collection: @functions ,
include_blank: false, label: "Apply Function", input_html: {type:
"text" } %>
      </div>
      <div class="col-md-2 traffic-columns">
        <%= f.input :select1, collection: @traffic_vault.sort +
[["------ Aggregations ------ ", ""]] +
@traffic_vault_aggregations.sort, include_blank: false, label: "Select
Dimensions (*)", input_html: {type: "text" } %>
      </div>
      <!-- <div class="col-md-3 vault-columns" style="display:none">
        <%= f.input :select, collection: @vault_columns,disabled:
true,include_blank: false,label: "Select Vault columns (*)",required:
false,input_html: {type: "text" } %>
      </div>
      <div class="col-md-3 monitor-columns" style="display:none">
        <%= f.input :select, collection: @monitor_columns,disabled:
true,include_blank: false,label: "Select Monitor Columns (*)",required:
false,input_html: {type: "text" } %>
```

```
      </div>
      <div class="col-md-3 intrusion-columns" style="display:none">
        <%= f.input :select, collection: @intrusion_columns,disabled:
true,include_blank: false,label: "Select Intrusion Columns
(*)",required: false,input_html: {type: "text" } %>
      </div> -->
      <div class="col-md-2">
        <%= f.input :select_name1, label: "New Column Name", required:
false, input_html: {type: "text"} %>
      </div>
    </div>
  </div>
  <div class="form-group row string">
    <div class="col-md-3">
      <%= link_to content_tag(:i, nil, class: "fa fa-plus"),
         '', style: 'color: green', class: "btn btn-small
add-new-select", 'data-backdrop' => "static" %>
      <%= link_to content_tag(:i, nil, class: "fa fa-minus"),
         '', style: 'color: red', class: "btn btn-small remove-select
disabled", 'data-backdrop' => "static" %>
    </div>
  </div>
</div>
```

Bibliografía

[1] *Solution - What is redborder?* (2020). RedBorder. https://redborder.com/en/solution/what_is_redborder.php

[2] Support, A. (2018, 30 mayo). *Plugin Overview (Input and Output, only)*. rsyslog. https://www.rsyslog.com/plugin-overview/

[3] *¿Qué es Apache Kafka?* (2020). What Is Apache Kafka. https://www.redhat.com/es/topics/integration/what-is-apache-kafka

[4] Elastic. (s. f.). *Logstash: Recopila, parsea y transforma logs*. https://www.elastic.co/es/logstash

[5] Basak, R. (2020, 24 marzo). *Apache Druid – The sine qua non of contemporary Big-Data analytics*. Medium. https://medium.com/swlh/apache-druid-the-sine-qua-non-of-contemporary-big-data-analytics-cc82a081e

[6] *Tutorial: Writing an ingestion spec · Apache Druid*. (s. f.). Ingestion Spec. https://druid.apache.org/docs/latest/tutorials/tutorial-ingestion-spec.html

[7] *CEP Configuration Overview - Complex Event Processor 3.1.0 - WSO2 Documentation*. (s. f.). CEP Configuration. https://docs.wso2.com/display/CEP310/CEP+Configuration+Overview

[8] *¿Qué es Apache Kafka?* (2020). What Is Apache Kafka. https://www.redhat.com/es/topics/integration/what-is-apache-kafka