Universidad de la República

Facultad de Ingeniería

WAFIntl Security Inspector

Tesis de grado para Ingeniería en Computación

Autor:
Leonardo Alberro

Supervisores:
Gustavo Betarte
Rodrigo Martínez
Marcelo Rodríguez

9 de junio de 2020

Resumen

En este proyecto se propone un modelo de datos y una herramienta que contempla la recepción, almacenamiento, análisis, correlación y presentación de información proveniente de múltiples *Web application firewalls* (WAFs). Para esto, en primera instancia se presenta un estado del arte que analiza en profundidad herramientas, metodologías y procedimientos existentes para abordar esta solución. En este sentido, se presentan tanto tecnologías como procedimientos para el manejo de los eventos provenientes de estos WAFs, se analiza el uso de indicadores de compromiso y se presentan técnicas para la correlación de eventos. Luego, se realiza un relevamiento y un análisis de los requerimientos y tecnologías que se deben contemplar en una solución de este estilo. En esta línea, se presentan los requerimientos para la centralización, análisis y visualización de los eventos, así como los requerimientos relacionados a la correlación de la información. Finalmente, se presenta un diseño y una prueba de concepto acorde al análisis propuesto definiendo las tecnologías evaluadas. Para esto, se define una arquitectura basada en la tecnología del stack *Elasticsearch-Logstash-Kibana* (ELK) en la que se diseña el flujo que los eventos deben seguir con el objetivo de cubrir los requerimientos analizados.

***Palabras claves**: ModSecurity, eventos, seguridad informática, WAF, centralización, correlación, ELK.*

Abstract

This project proposes a data model and a tool that contemplates the reception, storage, analysis, correlation and presentation of information from multiple web application firewalls (WAFs). In order to achieve this, at first a state of art is presented that deeply analyzes existing tools, methodologies and procedures to address this solution. In this sense, both technologies and procedures are presented for the management of the events coming from these WAFs. Besides, the use of indicators of compromise is analyzed and techniques for the correlation of events are presented. Then, a survey and analysis of the requirements and technologies that must be contemplated in a solution of this kind is done. In this line, requirements for the centralization, analysis and visualization of events are presented, as well as the requirements related to the correlation of the information. Finally, a design and a proof of concept are presented according to the proposed analysis defining the technologies evaluated. To that end, an architecture is designed based on the *Elasticsearch-Logstash-Kibana* (ELK) stack technology in which the flow that the events must follow is designed in order to meet the requirements analyzed.

Keywords: *ModSecurity, events, cybersecurity, WAF, centralization, correlation, ELK.*

Índice

1. Introducción

El proyecto se desarrolla en el contexto de los temas de investigación y trabajo desarrollados en el Instituto de Computación, en particular dentro del GSI[1], así como del proyecto de I+D que miembros de este grupo desarrollan como miembros del equipo de Seguridad de Tilsor SA en el contexto del ICT4V[2].

El objetivo general de este proyecto es proponer un modelo de datos y una herramienta que contemple la recepción, almacenamiento, análisis y presentación de la información provenientes de múltiples *Web application firewalls* (WAFs) utilizando la tecnología ELK. A su vez, también se analiza un procedimiento de generación de alertas. El análisis se centra en la correlación de eventos provenientes tanto de un mismo, como de múltiples WAFs.

La principal problemática a resolver consiste entonces, en generar un análisis y diseño de los requerimientos necesarios para la centralización y correlación de eventos provenientes de múltiples WAFs. Para esto, se debe generar conocimiento sobre tecnologías, herramientas y procedimientos definidos acerca del procesamiento de este tipo de eventos. La correlación de eventos provenientes de múltiples WAFs (múltiples organizaciones) agrega un desafío a la tarea de correlación. Este tipo de correlación por ejemplo, debe contemplar la posibilidad de detectar ataques donde, no sea posible detectar actividad sospechosa examinando únicamente los eventos de un único WAF, pero al analizar la actividad global entre varios sitios, se detecte por ejemplo, algún tipo de ataque distribuido. Contrario a su interpretación usual, este concepto de ataque distribuido, no refiere a un ataque desde múltiples orígenes, sino a un ataque donde un mismo atacante realiza una actividad maliciosa que implica a múltiples destinos.

Defenderse de los ciberataques es un problema cada vez más complejo, sobre todo cuando estos son dirigidos a una o un conjunto de organizaciones específicas. Los ciberataques de hoy requieren de nuevas líneas de defensas de seguridad. En el contexto local y global es normal el uso de los *web application firewalls* con una tecnología como *ModSecurity*, lo que genera una gran cantidad de logs de seguridad. En la práctica, un problema resulta disperso en muchos eventos generados, lo que conduce a tickets complejos y grandes que dificulta a los analistas de seguridad identificar la causa raíz del problema. La motivación de llevar a cabo este proyecto parte de la necesidad y ausencia en el mercado de una herramienta libre que brinde este tipo de soluciones. Por una parte, para facilitar la tarea a los analistas de seguridad, quienes deben correlacionar datos de manera no automatizada y siguiendo su *expertise* en el tema. Por otra, con el objetivo de introducir una herramienta configurable capaz de incorporar tecnologías y requerimientos que permitan, mediante la correlación de eventos distribuidos, generar nuevas alertas de seguridad que de otra manera podrían pasar desapercibidas.

Este proyecto genera un análisis de los requerimientos necesarios para la creación de una he-

[1]Grupo de Seguridad Informática: https://www.fing.edu.uy/inco/grupos/gsi/

[2]Information and Communication Technologies for Verticals, sitio web https://ict4v.org/

rramienta automatizada, que aborde los tipos de problemas mencionados. En este sentido, se desarrolla además, una prueba de concepto donde se valida el uso de las tecnologías relevadas y se instancian las principales funcionalidades analizadas. Verticalmente a esta solución, se logra un estado del arte que analiza técnicas existentes para la correlación de eventos y tecnologías como los indicadores de compromisos.

Este documento se organiza de la siguiente manera: en la sección 2 se listan y definen los principales conceptos manejados en este trabajo, en la sección 3 se presentan y analizan tecnologías, herramientas, procedimientos y metodologías relacionadas tanto a la centralización, correlación y manejo de eventos de seguridad, como a la definición de las características técnicas de las posibles amenazas a manejar. Luego, en 4 se presenta el análisis de la solución. Este análisis comienza con la presentación de un relevamiento externo de requisitos (4.1), también expone un análisis completo de la solución, incluyendo la presentación de los requerimientos necesarios. El diseño de la solución se presenta en 5, donde se plantean las principales decisiones de diseño tomadas para modelar los requerimientos: tecnologías, arquitectura, etc. Finalmente, una prueba de concepto para esta solución, se presenta en la sección 6. Por último, las principales conclusiones y trabajos a futuro obtenidos de este proyecto se muestran en 7.

Para referirse a otros documentos entregados en este proyecto se utiliza la siguiente notación: [**Nombre del documento**].

2. Conceptos básicos

A continuación se presenta una lista con las definiciones de los conceptos básicos involucrados en este documento. Algunas de estas definiciones están instanciadas de acuerdo al contexto de este trabajo.

- **Firewall**: es un dispositivo de seguridad de red que controla el flujo de tráfico entre dos partes de una red. En una organización, por ejemplo, los firewalls pueden restringir el acceso a servicios externos que se consideran peligrosos o innecesarios para la misma. [10]
- **WAF** (*Web application firewall*): es un tipo de firewall que supervisa, filtra o bloquea el tráfico HTTP hacia y desde una aplicación web. [20]
- **Log**: archivo de texto en el que constan cronológicamente los acontecimientos que han ido afectando a un servidor web. [3]
- **Evento**: ocurrencia única dentro de un entorno, que generalmente implica un intento de cambio de estado. Un evento está contenido en una o más líneas de un log.
- **Normalizar**: es un proceso donde la salida son eventos normalizados (estandarizados), i.e. con un mismo formato. [13]

- **Access log**: en un servidor web, el registro de acceso es donde el servidor almacena información sobre todas las peticiones que procesa.
- **Error log**: log donde el servidor web envía cualquier información de diagnóstico y registra cualquier error que encuentre al procesar peticiones.
- **SIEM** (*Security Information and Event Management*): es un sistema que centraliza el almacenamiento y la interpretación de los datos relevantes de seguridad.
- **Plugin**: aplicación que agrega una funcionalidad adicional o una nueva característica a cierto software.
- **SQLi**: un ataque de inyección SQL (Structured Query Language) es la vulnerabilidad que resulta cuando se le da a un atacante la capacidad de influir en las consultas SQL que una aplicación realiza a su base de datos en el back-end. [4]
- **Phishing**: es un tipo de ataque informático donde el usuario envía voluntariamente datos a través de un canal, pero siendo engañado acerca del punto final del mismo. [10]
- **Malware**: tipo de software que realiza acciones dañinas en un sistema informático de forma intencionada. [10]
- **Ransomware**: es un tipo de malware que restringe el acceso a determinadas partes o archivos del sistema operativo infectado y pide un rescate a cambio de quitar esta restricción.[9]
- **Denial-of-service (DoS)**: La denegación de servicio es una amenaza enfocada a denegar el acceso a usuarios válidos, como por ejemplo, a un servicio web. [10]
- **Distributed denial-of-service (DDoS)**: es un ataque coordinado utilizando una gran cantidad de hosts comprometidos para lograr la denegación de servicio. [1]
- **Honeypots**: son sistemas utilizados para rastrear a los atacantes con el objetivo de aprender y reunir evidencia sobre nuevas técnicas de ataque. Los Honeypots imitan sistemas reales pero no contienen y, por lo tanto, no revelan datos operativos reales. Por definición, cada actividad monitoreada en el honeypot no está autorizada. [10]
- **Hash**: una función de hash criptográfica h mapea entradas x con una longitud de bits arbitraria a las salidas de una función $h(x)$ con longitud de bit fija n. En algunos casos se utilizan los términos *hash*, *digest* o *fingerprint* para referirse al resultado de la función $h(x)$. [10]

3. Estado del Arte

En esta sección se presenta el estado del arte respecto a técnicas de almacenamiento, análisis, correlación y presentación de eventos de seguridad en aplicaciones web.

Este análisis comienza con una exploración de las dos tecnologías fundamentales y que se establecen como precondiciones para el desarrollo del proyecto: *ModSecurity* y el stack *ELK*. Luego, se ahonda en el manejo de eventos y técnicas de correlación.

También, se presenta un estado del arte relacionado a los indicadores de compromiso, los cuales son una tecnología estandarizada que consiste en definir las características técnicas de una amenaza por medio de las evidencias existentes en un equipo comprometido. Un análisis más a fondo sobre esta tecnología, realizado en este proyecto, se puede consultar en el documento [**Estado del Arte**].

Por último, se introduce la revisión de un antecedente encontrado durante la ejecución de este proyecto.

3.1. ModSecurity

ModSecurity es un módulo de los WAFs, multiplataforma y de código abierto. Mediante la incorporación de reglas, provee protección a las aplicaciones web contra un gran número de ataques, además de monitorear el tráfico HTTP y analizarlo en tiempo real. Permite a los encargados de la seguridad de aplicaciones web, obtener visibilidad del tráfico HTTP(S) y proporciona un lenguaje de reglas y una API para implementar protecciones avanzadas. Funciona como cualquier otro módulo de Apache[3], un módulo compartido binario y archivos de configuración.[14] Esta herramienta se basa en cuatro principios fundamentales: flexibilidad, pasividad, previsibilidad y calidad sobre cantidad.

Las funcionalidades destacadas que brinda son:

- **Registro de tráfico HTTP**: Apache HTTP Server incluye una multitud de funcionalidades alrededor del registro de las peticiones entrantes, pero pocas orientadas a las aplicaciones web. El registro habitual de Apache no incluye, por ejemplo, el cuerpo de la respuesta. Con *ModSecurity* se puede registrar la transacción completa de la petición/respuesta, con mucha flexibilidad para filtrar el tráfico que se desea registrar, así como para enmascarar los campos sensibles. Además, *ModSecurity* brinda la capacidad de registrar cualquier cosa que se necesite, incluidos los datos de transacciones sin procesar, lo que es esencial para el análisis forense. También permite elegir qué y cuáles partes de las transacciones se registran, así como qué partes se deben sanitizar.

[3]The Apache HTTP Server Project: https://httpd.apache.org/

- **Fortalecimiento de aplicaciones web**: *ModSecurity* se puede usar para la reducción de la superficie de ataque, con lo cual se puede reducir selectivamente las funciones HTTP que se está dispuesto a aceptar (por ejemplo, métodos de solicitud, encabezados, tipos de contenido, etc.). Esta herramienta puede ayudar a hacer cumplir muchas restricciones similares, ya sea directamente o mediante la colaboración con otros módulos de Apache. Por ejemplo, es posible solucionar muchos problemas de administración de sesiones, así como vulnerabilidades de falsificación de solicitudes entre sitios.
- **Monitorización en tiempo real y detección de ataques**: Para la monitorización en tiempo real y la detección de ataques *ModSecurity* incorpora un motor de reglas que carga en el arranque del proceso y que definen el funcionamiento del WAF. Estas reglas están definidas en una sintaxis muy simple que se va aprendiendo a medida que se examina la configuración inicial. Esta configuración incluye las directivas básicas de configuración y unas reglas básicas de protección. Adicionalmente se puede incorporar el *Core Rule Set*(CRS), que son un conjunto de reglas mantenido por OWASP[4].

3.1.1. Configuración y funcionamiento

La configuración de *ModSecurity*, análogamente con Apache, está organizada en directivas que ofrecen una gran flexibilidad para abarcar una gran variedad de situaciones que se presentan. Se puede denegar o permitir peticiones en base muchos aspectos, por ejemplo, el tamaño de un archivo que se pretende subir vía POST, el patrón en alguna de las cabeceras de petición o respuesta, procedencia geográfica de la petición, horario del día, entre otros. Además, estas situaciones se pueden combinar entre si.

La directiva más simple y usada para generar estas reglas es:

```
SecRule TARGET OPERATOR [ACTIONS]
```

Esta directiva crea una regla comparando el valor de la variable (o colección) `TARGET` con `OPERATOR`, y en caso de hacer match se ejecutan las acciones. En caso de no especificar nada en `ACTIONS` aplica `SecDefaultAction` que permite establecer las acciones por defecto.

Respecto al funcionamiento, *ModSecurity* divide una transacción HTTP en cinco fases:

1. **REQUEST HEADERS**: Después de leer las cabeceras de la petición.
2. **REQUEST BODY**: Después de leer el cuerpo de la petición. Casi todas las reglas se definen para ser procesadas en esta fase.
3. **RESPONSE HEADERS**: Antes de enviar las cabeceras de respuesta al cliente.

[4]Open Web Application Security Project: www.owasp.org

4. **RESPONSE BODY**: Antes de enviar el cuerpo de la respuesta al cliente. En esta fase se han de programar las reglas que impidan la fuga de información.
5. **LOGGING**: Antes de registrar la transacción. En esta fase no se puede realizar ninguna acción ya que la respuesta la tiene el cliente, por esto, solo puede afectar al modo en que se registra.

Luego de esta introducción a *ModSecurity*, y debido a que es una condición del proyecto trabajar con WAFs de esta tecnología, se usarán indistintamente los términos WAF y *ModSecurity*.

3.2. ELK

Las soluciones modernas como los sistemas distribuidos, producen cada día, un gran volumen de entradas en los logs provenientes de múltiples fuentes en la red. Dirigir la actividad de los sistemas a un solo punto es una propiedad deseable y buscada. Una vez que toda la información se encuentra centralizada, el análisis y la búsqueda, se convierten en tareas manejables si se elige las herramientas adecuadas.

ELK[5] es la combinación de tres proyectos open source: *Elasticsearch*, *Logstash* y *Kibana* que se utiliza para proporcionar un enfoque integral en la consolidación, gestión y análisis de los logs de las aplicaciones. Este stack, simplifica la búsqueda y el análisis de datos al proporcionar información en tiempo real a partir de los datos en estos logs. Además, potencialmente puede proporcionar información en tiempo real a partir de datos consolidados.

3.2.1. Funcionalidades

El stack ELK satisface una necesidad específica en el espacio de análisis y gestión de logs. En las infraestructuras basadas en la nube, por ejemplo, la consolidación de logs a una ubicación central desde diferentes fuentes como servidores web, servidores de correo, servidores de bases de datos y dispositivos de red puede ser particularmente útil. Por ejemplo, cuando se trata de tomar mejores decisiones basadas en datos.

Las soluciones que utilizan esta tecnología normalmente ejecutan el *stack ELK* completo y no cada componente individual por separado. Cada uno de estos servicios desempeña un papel importante y, para hacerlo bajo una gran demanda, es más ventajoso implementar cada servicio en su propio servidor. Aún cuando al hacerlo, se introduzcan otros problemas relacionados con implementaciones multinodos, redes, seguridad y administración.

Los componentes del stack ELK son:

[5]https://www.elastic.co/es/what-is/elk-stack

Elasticsearch: Motor de búsqueda y análisis con el que se puede interactuar vía API, y es capaz de indexar información fácilmente. Está concebido para almacenar los datos de forma escalable.

Logstash: es un pipeline de procesamiento de datos del lado del servidor. Recopila, procesa y reenvía eventos. Recibe los eventos de los logs y los retransmite a **Elasticsearch**. Logstash es definido por sus desarrolladores como *"un canal abierto de procesamiento de datos del lado del servidor que ingiere datos de una multitud de fuentes simultáneamente, las transforma y luego las envía a otra tecnología"*[6].

Kibana: Motor de visualización que se construye sobre una instancia de **Elasticsearch** y proporciona capacidades de visualización sobre el contenido indexado en un clúster del mismo. Kibana permite a los usuarios visualizar los datos de Elasticsearch en cuadros y gráficos.

Además de estos componentes se puede mencionar a **Beats**. Esta herramienta es una plataforma para los agentes de datos con un solo propósito. Puede enviar datos de cientos o miles de máquinas y sistemas a Logstash o Elasticsearch. Si bien este no es un componente necesario del stack, tiene una buena sinergia con estos componentes y vale la pena tenerlo en cuenta para soluciones que involucren a ELK.

La figura 1 ilustra como es el flujo de los eventos en una solución de basada en esta tecnología.

Figura 1: Flujo de logs por el stack ELK

En el mercado hay herramientas que son o integran sistemas SIEM, como IBM QRadar SIEM, RSA enVision, Security MARS o Alien Vault USM. Sin embargo, debido a, por ejemplo, su elevado precio y costo de mantenimiento, es habitual usar implementaciones con software libre usando una plataforma *ELK*. Esto logra aprovechar las capacidades de recolectar información usando *Logstash*, las capacidades de almacenamiento, búsqueda y análisis de *Elasticsearch*, más las capacidades de visualización y exploración de *Kibana*. Adicionalmente a la plataforma, es habitual también instalar herramientas para sacar provecho a dicha información, para por ejemplo, correlacionar eventos (e.g. Simple event correlator) o realizar análisis estadísticos.[19]

[6]https://vcatalan.com/2017/12/pila-elk.html

3.3. Eventos

Según [3] un *evento* es una ocurrencia única dentro de un entorno, que generalmente implica un intento de cambio de estado. Incluye, generalmente, una noción de tiempo, la ocurrencia y cualquier detalle relacionado explícitamente con el *evento* o entorno, que pueda ayudar a explicar o comprender las causas o efectos del mismo. Un *campo de evento* describe una característica de un *evento*, por ejemplo: fecha, hora, IP de origen, identificación del usuario e identificación del host. En esta línea, para llegar a la definición de *log* se define también el concepto de *registro de evento* como una colección de *campos de evento* que, en conjunto, describen un *evento* único. Los sinónimos de este concepto incluyen "registro de auditoría" y "entrada de registro". Finalmente, se define *log* como una colección de *registros de eventos*. Y se incluyen los términos "log de datos", "log de actividades", "log de auditoría", "seguimiento de auditoría", "archivo de log" y "log de eventos" como sinónimos del mismo.

3.4. Manejo de eventos

En general, los *logs* están compuestos de entradas de registro, cada entrada contiene información relacionada con un evento específico que ha ocurrido dentro de un sistema observado. Esta información está etiquetada principalmente como un mensaje de registro. Los logs proporcionan una visión general sobre el estado actual y el historial de eventos relacionados con el funcionamiento de los componentes individuales del sistema. La información almacenada en los logs se puede utilizar para identificar incidentes de seguridad, infracciones de políticas, actividad fraudulenta, y problemas operacionales, entre otros.

Los *logs de seguridad* son generados por muchas fuentes, como por ejemplo antivirus, firewalls, IDS[7]/IPS[8], sistemas operativos en servidores, equipos de redes, aplicaciones, etc. El volumen y la variedad de estos logs es cada vez mayor debido a que los ataques cibernéticos son frecuentes y están en crecimiento. Esto provocó la necesidad de una administración para los mismos, compuesta de los siguientes procesos: *generar, transmitir, almacenar, analizar* y *eliminar*. [3][12].

La operación más importante, desde el punto de la seguridad del sistema, es el **análisis**. Actualmente, las herramientas automatizadas se utilizan exclusivamente para **analizar** debido a que hay una gran cantidad de eventos guardados que no se pueden analizar manualmente. Formalmente, según el estudio [13], las operaciones requeridas para aplicar herramientas de análisis automatizadas son:

- **Aggregation**: representa el proceso de descarga y almacenamiento de registros en una ubicación central.

[7]Sistema de detección de intrusos
[8]Sistema de prevención de intrusos

- **Filtering**: es la decisión, sobre los datos sin procesar almacenados en los registros, de qué datos son importantes para el análisis y cuáles no.
- **Normalization**: la salida de este proceso son registros normalizados (estandarizados).
- **Correlation**: en esta parte se intenta encontrar combinaciones de un registro o una serie de registros para plantear una acción adecuada. Se busca generar un mayor nivel de conocimiento.
- **Reporting**: es una actividad posterior a la correlación que puede tomar la forma de una alarma, correo electrónico, etc.

La figura 2 ilustra el flujo para las operaciones descritas anteriormente.

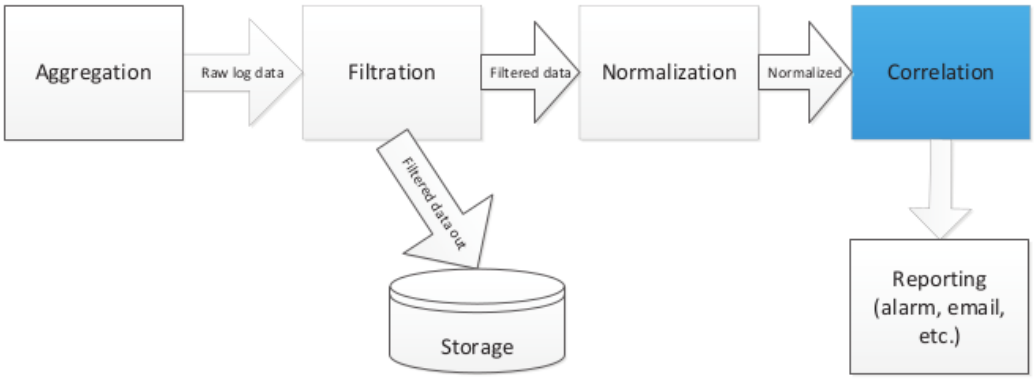

Figura 2: Operaciones necesarias para un análisis automatizado [13].

3.5. Correlación

El significado del término *correlación* se utiliza para hacer referencia a las relaciones entre diferentes eventos. La *correlación de eventos* se realiza generalmente para obtener un conocimiento de la información de más alto nivel de la que pueden proporcionar los eventos, e.g. identificar situaciones extraordinarias, identificar la causa raíz de un problema, o realizar predicciones sobre el futuro y descubrir tendencias. Existen diversos enfoques en las que puede ser útil la *correlación de eventos*: análisis de datos de mercado, detección de ataques informáticos, detección de fraude, análisis de logs del sistema (por ejemplo, agrupar mensajes similares y aumento de acontecimientos importantes) o análisis de gestión y fallas de red, entre otros. [15]

Para realizar estas funcionalidades, la *correlación de eventos* se puede separar en cuatro fases [21]:

- *Event filtering*: consiste en descartar eventos que se vuelven irrelevantes para las ocurrencias que deben observarse.
- *Event aggregation*: se basa en la fusión de eventos duplicados.

- *Root cause analysis*: es la tarea de analizar las dependencias entre eventos, generalmente basadas en un modelo del entorno y los grafos de dependencia, para detectar si algunos eventos pueden ser explicados por otros.
- *Event masking*: se trata de ignorar los eventos que pertenecen a sistemas que están en una etapa posterior a un sistema fallido.

Además de estos cuatro pasos, se menciona el *action triggering*, que se define como la tarea encargada de incluir las capacidades de resolución de problemas, por ejemplo, las acciones correctivas.

3.5.1. Metodologías

Para correlacionar eventos existe una gran variedad de técnicas disponibles, y combinaciones de ellas. Que un enfoque sea mejor que otro depende, en gran medida, del problema que se quiera abordar. Los enfoques aquí presentados, son algunos de los expuestos en [15] y [21].

Para entender estos métodos de correlación de eventos es necesario introducir algunas propiedades que estos pueden tener:

- **Domain Awareness**: un motor de correlación puede construirse para un dominio específico ("sabe"qué tipo de información procesa), o como un motor de correlación de propósito general.
- **Self-Learning vs. External Knowledge**: para la correlación se requiere conocimiento, como por ejemplo información sobre la estructura de la red, información sobre los desencadenantes de los eventos o información sobre las dependencias del servicio. Esta propiedad clasifica dependiendo de como se introduce esta información al motor de correlación.
- **Real-time vs. Stored Data**
- **Stateless vs. Stateful**: un motor de correlación en tiempo real puede ser con estado (i.e. tiene una memoria del historial de eventos) o sin estado (sin memoria).
- **Centralized vs. Distributed**

Finite State Machine Based

La correlación de eventos basada en FSM(*Finite State Machine*) hace posible definir formalmente y computar automáticamente un FSM que representa a un "diagnosticador". El precio que se debe pagar por este enfoque es bastante alto, ya que requiere que un FSM que modela los comportamientos del sistema esté disponible, lo que en la práctica rara vez ocurre.

Rule Based Event Correlation

En la correlación de eventos basada en reglas (RBR), el sistema utiliza constantemente un conjunto de reglas predefinidas para evaluar las observaciones entrantes hasta que se llega a una conclusión. La capacidad de correlación depende únicamente de la profundidad y la capacidad del conjunto de reglas.

Case Based Reasoning

En la correlación basada en casos (CBR), se intenta resolver un problema dado, buscando el caso más similar en una biblioteca de casos y recuperando la solución. Un caso consiste en un problema, su solución y, típicamente, anotaciones sobre cómo se derivó la solución. Algunas veces puede requerir la adaptación de una solución almacenada en la biblioteca, para resolver el problema actual. Este principio consiste en resolver problemas incrementales con un componente de aprendizaje sostenido. Después de aplicar la nueva solución al problema, se verificará el resultado y, si tiene éxito, el nuevo caso (problema y solución) se almacenará en la biblioteca. De lo contrario, se debe proponer una mejor solución, que se verificará e incorporará a la biblioteca. Por lo tanto, un sistema CBR "*aprende de la experiencia y puede adaptarse a problemas desconocidos*".

Model Based Reasoning

Esta metodología refiere al uso de un modelo del mundo físico que representa la estructura y el comportamiento del sistema bajo observación, como un método de inferencia.

Este enfoque no sugiere una técnica detallada, sino un paradigma. Tiene una conexión con los sistemas basados en reglas, ya que una implementación práctica podría usar un modelo basado en reglas, pero difiere de estos, en que especifica un modelo de sistema con eventos como consecuencia de ciertos estados de modelo y transiciones. En contraste, RBR especifica patrones de eventos como condiciones para ciertas acciones.

Voting Approaches

La idea del enfoque de votación, es que cada elemento deba expresar su opinión sobre un tema específico. Luego, se aplica una regla de mayoría (e.g. mayoría absoluta) en este conjunto de opiniones (es decir, votos).

La correlación por votación se puede utilizar para localizar una falla. Generalmente, los votos (expresados por eventos de diferentes nodos) no pueden proporcionar información exacta sobre la ubicación de una falla, pero pueden indicar una dirección.

Dependency Graphs

Un grafo de dependencia es un grafo dirigido, que modela las dependencias entre los objetos gestionados. En el caso de una red, los nodos representan los elementos de la red (por ejemplo,

los hosts), y una arista desde el nodo A al nodo B indica que las fallas en el nodo A pueden causar fallas en el nodo B. Suponiendo que algunos eventos de error se generaron en una ventana de tiempo dada, el objetivo es encontrar la causa raíz probable.

Otros enfoques encontrados en la literatura son: *Codebook Based Event Correlation, Bayesian Network Based Event Correlation, Neural Network Approaches, Probabilistic Event correlation* y combinaciones de estos con los presentados anteriormente, así como nuevos métodos que surgen de escenarios particulares.

Por ejemplo, en [16] se propone un método que busca mejorar el rendimiento del análisis predictivo, basándose en la hipótesis de que las técnicas existentes toman como entrada algunos datos procesados o minados para la predicción. El mismo, propone reemplazar la fase de procesamiento de datos con la técnica de correlación de eventos. Este método, basa la correlación en un enfoque *basado en reglas* para procesar eventos *en tiempo real*, lo que produce *clases de ataque* como salida para la detección de ataques de múltiples etapas. Con esto se reduce la sobrecarga de procesamiento, lo cual conduce a una baja utilización de la memoria. La propuesta también incluye el modelo oculto de Márkov (HMM) que hace uso de estos datos altamente procesados, para lograr *predicciones de ataques futuros*.

A su vez, en [13] se propone una metodología propia para identificar los eventos de seguridad requeridos. El enfoque se basa en las *evaluaciones de riesgo teóricas* proporcionadas por el NIST[9] y en información más práctica proporcionada por OWASP[10]. La idea principal detrás del método, radica en la utilización de los resultados del *análisis de riesgos* para identificar los eventos de seguridad deseados (incluida su correlación). La metodología contiene dos bloques básicos:

1. *Risk Analysis Process*:
   - *a*) Análisis del sistema.
   - *b*) Identificación de activos.
   - *c*) Análisis de amenazas.
   - *d*) Análisis de ataques.
   - *e*) Análisis de vulnerabilidades.
2. *Record Identification Process*:
   - *a*) Determinar las categorías de mecanismo de amenaza.
   - *b*) Definir vectores de ataques.

[9]NIST: Instituto Nacional de Estándares y Tecnología
[10]The Open Web Application Security Project

c) Identificación de eventos y sus correlaciones.

d) Proponer registros.

Ventajas y Desventajas

Las ventajas y desventajas para las metodologías presentadas se expresan en [15] y se resumen en el Cuadro 1.

Enfoque		**Fortalezas y Debilidades**
FSMs	**Pros**	Simple, bueno como modelo básico, fácil de entender.
	Contras	Demasiado simple para aplicaciones de la práctica, sin tolerancia a ruido.
RBR	**Pros**	Comportamiento transparente, cercano al lenguaje natural, modularidad.
	Contras	Tiempo de mantenimiento, poca robustez, no aprende de la experiencia.
CBR	**Pros**	Aprende automáticamente de la experiencia, el razonamiento sobre experiencia pasada es natural.
	Contras	La adaptación y reutilización automática de la solución es difícil.
MBR	**Pros**	Se basa en el conocimiento profundo.
	Contras	La descripción del comportamiento y la estructura puede ser difícil en la práctica.
Codebook	**Pros**	Rápido, robusto, se adapta a los cambios de topología.
	Contras	Reproducir el comportamiento manualmente es tedioso; sin noción de tiempo.
Voting	**Pros**	Ideal para usar de manera distribuida.
	Contras	Requiere conocimiento de la topología.
Dependecy Graphs	**Pros**	Bueno para tratar con sistemas dinámicos y complejos de administrar.
	Contras	Supone que solo hay un problema a la vez.
Bayesian Networks	**Pros**	Buen fundamento teórico.
	Contras	La inferencia probabilística es *NP-hard*.

Cuadro 1: Ventajas y desventajas de los enfoques para la correlación de eventos.

3.6. Indicadores de compromiso

Los Indicadores de Compromiso o «*Indicators of Compromise*» (IOCs) hacen referencia a una tecnología estandarizada que consiste en definir las características técnicas de una amenaza por medio de las evidencias existentes en un equipo comprometido. Es decir, se identifican diferentes acciones como ficheros creados, entradas de registro modificadas, procesos o servicios nuevos, de manera que puedan servir para identificar otros nodos afectados por la misma amenaza o prevenirlos de la misma. [6]

De este modo, es posible realizar un intercambio sencillo y práctico de información con otras personas y grupos de gestión de incidentes e implementar las firmas en diferentes herramientas como:

- **Sistemas de detección de intrusiones** (IDS).
- **Sistemas de prevención de intrusiones** (IPS).
- **Sistema de detección de intrusiones en un Host** o *Host-based Intrusion Detection System* (HIDS).
- **Sistema de prevención de intrusiones en un Host** o *Host-based Intrusion Prevention System* (HIPS).
- **Firewalls**.

El valor real de la tecnología radica precisamente en ese intercambio de información ya que permite generar y compartir conocimiento a partir de análisis forenses, respuesta a incidentes o análisis de malware.

Algunos ejemplos de indicadores de compromiso generalmente monitoreados son:

- Tráfico de red saliente inusual.
- Anomalías en la actividad de alguna cuenta de usuario privilegiada.
- Incremento del volumen de lectura de la base de datos.
- Tamaños de respuesta HTML.
- Solicitudes DNS inusuales.
- Signos de la actividad DDoS.

Este tipo de tecnología se analiza con el objetivo de incorporar este concepto al análisis de la solución de este proyecto.

3.6.1. Tipos de indicadores

No todos los indicadores son iguales, y dependiendo del contexto, algunos de ellos son más valiosos que otros. En [5] se ilustra este concepto creando el diagrama llamado *"Pirámide del Dolor"*. Este diagrama muestra la relación entre los tipos de indicadores que se podría usar para detectar las actividades de un adversario y la "cantidad de dolor"que les causará cuando se le pueda negar esos indicadores.

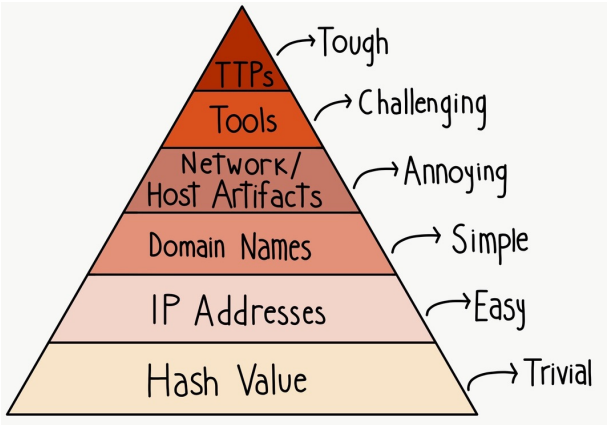

Figura 3: Pirámide del dolor [5]

Los indicadores que componen esta pirámide están dados por:

1. **Hash Values**: SHA-1, MD5 u otros hashes similares que corresponden a archivos sospechosos o maliciosos específicos. A menudo se utiliza para proporcionar referencias únicas a muestras específicas de malware o archivos involucrados en una intrusión.
2. **IP Addresses**.
3. **Domain Names**: Podría ser un nombre de dominio o también subdominios.
4. **Network Artifacts**: Observables causados por actividades adversas en una red. Por ejemplo patrones URI, información incrustada en los protocolos de red, valores distintivos de HTTP User-Agent o SMTP Mailer, entre otros.
5. **Host Artifacts**: Observables causados por actividades adversas en uno o más hosts. Pone foco en cosas que tienden a distinguir las actividades maliciosas de las legítimas. Pueden ser claves de registro, o valores que se piensa que son creados por piezas específicas de malware, archivos o directorios que se han creado en ciertos lugares, o que usan ciertos nombres, servicios maliciosos o casi cualquier otra cosa que sea distintiva.
6. **Tools**: Software utilizado por el adversario para cumplir su objetivo. En su mayoría, esto será lo que instalen/descarguen, en lugar del software o los comandos que ya estén instalados. Incluiría las utilidades diseñadas para crear documentos maliciosos para *spearp-*

hishing[11], *backdoors* utilizadas para establecer crackers de contraseña u otras utilidades basadas en el host que quieran usar luego del compromiso.

7. **Tactics, Techniques and Procedures (TTPs)**: Es una descripción de cómo avanza el adversario en su misión, desde el reconocimiento hasta la exfiltración de datos y en cada paso intermedio. Por ejemplo *"Spearphishing"* es un TTP común para establecer una presencia en la red. *"Spearphishing con un archivo PDF troyano"* o *"... con un enlace a un archivo .SCR malicioso disfrazado de ZIP"* serían versiones más específicas de este. Otro ejemplo trivial sería el *"volcado de credenciales de autenticación en caché y su reutilización en los ataques de Hash"*.[5]

3.6.2. Aplicaciones prácticas

Las aplicaciones prácticas que se encuentran en la literatura abarcan una gran cantidad y variedad de temas. En [22] se utilizan los *Indicadores de Compromiso* para ransomwares. Estos IOC se utilizan para establecer una base para el análisis y la clasificación de un ransomware, en su clase respectiva. En dicho trabajo se logra además, automatizar el sistema utilizando algoritmos de aprendizaje automático para la clasificación de variantes de ransomware en un entorno de tiempo real. Por otra parte, en [8] se ataca el problema de *Tracking the Known* (TTK) que es el problema de mantener las IOC actualizadas con un adversario dinámico. Esto es necesario debido a que con el tiempo, el adversario cambiará las tácticas, técnicas y procedimientos (TTPs) y naturalmente también cambiarán los datos generados.

Otro trabajo que busca identificar las amenazas cibernéticas basadas en patrones de ataque observados se detalla en [17]. Allí se propone un nuevo framework, basado en el aprendizaje automático, que identifica las amenazas basándose en los patrones de ataque observados con el objetivo de automatizar el proceso de análisis de amenazas mediante el mapeo de TPPs. Este framework relaciona semánticamente las amenazas y los TTP extraídos de fuentes de amenazas conocidas con mecanismos de detección asociados, para formar una red semántica. Esta red se usa luego para determinar las ocurrencias de amenazas formando relaciones probabilísticas entre las amenazas y las TTP.

En [2] se exploran los ataques en las partes aisladas de Internet, conocidas como Dark Web, que son operadas por protocolos descentralizados y de preservación anónima como Tor[12]. Para esto se desplegó un honeypot de alta interacción en la red Tor durante un período de siete meses, con el objetivo de realizar un estudio de medición del tipo de ataques y del comportamiento de los atacantes en este tipo de redes. Basado en la premisa de detectar la presencia de programas maliciosos en los sistemas tradicionales, la comunidad forense se basa en los Indicadores de Compromiso (IOC) y estudian el uso de estos en el contexto de las aplicaciones web. En este

[11] *estafa de correo electrónico o comunicaciones dirigida a personas, organizaciones o empresas específicas.*
[12] The Onion Router

sentido, desarrollan un nuevo tipo de indicadores de compromiso, denominados *Web Indicators of Compromise* (WIOC)[2], partiendo de la prueba empírica obtenida después de operar un honeypot web durante varios años. Estos indicadores se basan en la evidencia de que a menudo los atacantes usan componentes externos en sus páginas maliciosas o comprometidas. A pesar de que la mayoría de estos componentes no son maliciosos por si solos, los mismos proveen herramientas para lograr el compromiso de los sitios.

3.7. Antecedentes

Durante el desarrollo de este proyecto se conoció la existencia del proyecto *Securely*[13]. *Securely* es un proyecto en desarrollo que tiene como objetivo recopilar, enriquecer y correlacionar Apache Access y *ModSecurity* logs de un mismo WAF, mediante el uso del stack ELK. Surge debido a una necesidad de automatización en el manejo de este tipo de eventos de seguridad. El software se encuentra actualmente en construcción y tiene únicamente el front-end de libre acceso.

El front-end está basado en la implementación de un plugin para Kibana y la incorporación de otro plugin existente, ElastAlert[14]. El plugin desarrollado, genera visualizaciones nuevas para un mejor análisis de la información e incorpora algunas funcionalidades, como por ejemplo, un creador de reglas (para el WAF) automático en función de ciertas exclusiones elegidas de los eventos mostrados en Kibana. Estas reglas se pueden personalizar antes de enviarlas al firewall.

El back-end actualmente no cuenta con acceso público, en este sentido, se logró concretar una reunión con Ruben van Vreeland[15] cofundador y CEO de la empresa holandesa BitSensor[16] que brinda soluciones en seguridad de aplicaciones web, con el objetivo de tener más información acerca de este desarrollo.

De este intercambio, se logró tener una visión general del proyecto mencionado. Además, se obtuvo acceso a una parte del back-end. En particular al pipeline de ingreso, filtrado y envío de los eventos en Logstash el cuál se presentará más adelante en este proyecto.

Una evaluación de diseño acerca de las acciones a tomar en este proyecto respecto a la utilización de este antecedente se presentará en secciones subsiguientes.

4. Análisis de requerimientos

Este análisis contempla una solución de recepción, almacenamiento y presentación de la información proveniente de los WAF ModSecurity, así como el análisis de soluciones y mecanismos

[13] https://git.securely.ai/securely
[14] https://github.com/Yelp/elastalert
[15] https://www.linkedin.com/in/rubenvanvreeland/
[16] https://bitsensor.io/

para la correlación de eventos de múltiples WAFs.

El análisis realizado busca relevar y definir los requerimientos necesarios para el desarrollo de una completa solución a los problemas planteados. En este sentido, se realiza un relevamiento de requisitos y se especifica una variedad de requerimientos posibles para una solución a los problemas planteados. Se debe tener presente que estos requerimientos deberán ser mapeados sobre el stack ELK.

Más adelante, se diseñará e implementará en una prueba de concepto, un subconjunto necesario de los requerimientos analizados.

4.1. Relevamiento

El relevamiento realizado se llevó a cabo en conjunto con dos *stakeholders* identificados e interesados en participar. En concreto, se trabajó con el Centro Nacional de Respuesta a Incidentes de Seguridad Informática (CERTuy)[17] y el Centro de Operaciones de Seguridad (SOC)[18].

Estos actores son de gran importancia debido a que tienen un amplio conocimiento como analistas de seguridad en el dominio nacional. De esta manera, se logró obtener requisitos funcionales adaptados a las necesidades de los analistas que potencialmente podrían requerir una solución de este estilo.

A continuación se expondrán algunos de los requisitos relevados, los cuales se formalizarán en las secciones siguientes.

1. **Clasificación de atacante**: es de gran importancia establecer que tipo de atacante está realizando un ataque o actividad sospechosa. Cuanto más conocimiento se tenga a cerca del mismo, se podrán tomar mejores acciones para mitigar y prevenir estas acciones. Para esto, se definieron tres tipos de atacantes:
   - **user**: correspondiente a cuando el atacante tiene un comportamiento "humano".
   - **tool**: cuando la actividad corresponde a acciones llevadas a cabo por herramientas de software.
   - **crawler**: cuando la actividad corresponde a un programa que analiza los documentos de los sitios web. Este tipo de actividad es bastante frecuente debido a que los motores de búsqueda cuentan con rastreadores(crawlers) que navegan y analizan los sitios web.

   Poder identificar comportamiento humano es particularmente importante. Esto se debe a las acciones que se pueden llegar a tomar una vez adquirido este conocimiento. Es decir, que las acciones a tomar serán diferentes según el atacante. En particular, cuando se trata de un

[17] https://www.gub.uy/centro-nacional-respuesta-incidentes-seguridad-informatica/
[18] https://www.agesic.gub.uy/innovaportal/v/6672/1/agesic/centro-de-operaciones-de-seguridad–soc.html

atacante del tipo *user*, el mismo resulta más impredecible, a diferencia de los atacantes de tipo *tool*, que en general realizan siempre las mismas acciones. También resulta importante identificar a los usuarios, debido a que generalmente los mismos son capaces de causar mayor daño y la mitigación de sus ataques debe ser abordado de forma diferente al resto. En esta línea, se relevaron algunos indicadores de comportamiento que podrían "delatar" al humano, así como descartarlo. Estos indicadores surgen de la experiencia en análisis forense de los actores involucrados. En concreto, se puede clasificar el comportamiento de una actividad si se está frente a alguno de estos escenarios:

- En el contexto de un ataque SQLi[19], se pueden analizar las consultas recibidas con el objetivo de detectar consultas "complejas". Considerar a una consulta como compleja puede derivar en detectar que la misma incluya, por ejemplo, un SELECT o algún tipo de consulta considerada elaborada. Este comportamiento clasifica al tipo de atacante como **user**.
- El uso de *user agents* extraños o fuera de las versiones usuales inclinaría la clasificación del atacante a **tool** o **crawler**. Esta "heurística" es basada en la regla 80/20[20]. Esto es debido a que en general, las herramientas de ataque utilizan versiones fijas de los navegadores, que no son actualizadas mientras no surja una actualización para esta herramienta. Esto contrasta con los usuarios, que generalmente utilizan navegadores actualizados.
- Un ataque en determinada franja horaria puede delatar comportamiento humano. Generalmente si se analizan los ataques a largo plazo, el comportamiento de un **user** sigue determinados horarios.
- Si en una sesión de actividad se detecta que solicitudes posteriores van "arreglando" determinados parámetros (e.g. de una SQLi), el comportamiento es asociado a un humano. Formalmente, serían solicitudes con mutaciones en algo similar.

2. **Detectar ataques**: respecto a la correlación para la detección de ataques se relevaron algunos comportamientos, en principio asociados a único nodo (WAF). Las actividades que pueden indicar estar frente a un ataque son las siguientes:
   - En una sesión de actividad no se respeta un código de estado HTTP 30x, i.e 301, 302, etc., en forma sistemática. Esto además inclina la clasificación del tipo de atacante a **tool**.
   - Notar la presencia de solicitudes que demoran un tiempo anómalo.
   - Tener solicitudes de tipo GET a cosas nuevas.

[19] ataque de inyección SQL, ver conceptos básicos: sección 2

[20] Establece que, de forma general y para un amplio número de fenómenos, aproximadamente el 80 % de las consecuencias proviene del 20 % de las causas.

- Tener solicitudes de tipo POST a una URL inusual, e.g. a favicon.
- Observar un tamaño de las respuestas anómalo.
- Identificar de hashes idénticos. Por ejemplo, notar hashes idénticos de los datos de una solicitud POST en distintos nodos (correlación distribuida).

3. **Generar alertas**: es deseable contar con algún tipo de alertas. Las alertas útiles para un analista de seguridad de este dominio son:
   - Emitir una alerta cuando la cantidad de solicitudes de determinado AS o país supere cierto umbral establecido. Este tipo de alertas pueden ser individuales para cada nodo o para un conjunto de los mismos. Las alertas individuales simplifican la tarea del analista al identificar el nodo que está recibiendo los ataques.
   - Generar una alerta cuando el tamaño de las solicitudes GET/POST tengan un comportamiento fuera de lo normal (previamente establecido).

   Estos requerimientos son en general los primeros análisis que hacen las herramientas típicas de análisis de anomalías.

4. **No funcionales**: uno de los requerimientos centrales que debe tener una solución de este estilo es que la misma sea altamente configurable/parametrizable. Esto es debido a la importancia que toma la customización de estos parámetros en cada escenario. Contar con esto, puede potenciar la utilidad de la solución y darle más generalidad.

   En concreto, se deben definir de manera configurable los parámetros que definan conceptos como "fuera de lo normal", "cierto umbral", "anómalo", entre otros. Un ejemplo, es el concepto de "sesión", o la definición de "fuera de lo normal" a la hora de clasificar el tipo de atacante.

4.2. Centralización de logs

La centralización de este tipo de información implica cuatro tareas claves: **Recepción**, **Almacenamiento**, **Análisis** y **Presentación**. A su vez, como se presentó en 3.4, la operación considerada más importante, desde el punto de vista de la seguridad del sistema, es la de análisis. Por esto, las tareas de recepción y almacenamiento pueden tener requerimientos específicos pensados en la simplificación del análisis (e.g. filtrado y normalización).

Desde el punto de vista arquitectónico se cuenta con tres requerimientos claves:

1. Una fuente de procesamiento de datos, de código abierto, que ingiera datos de una multitud de fuentes simultáneamente.

2. Un motor de búsqueda y análisis, capaz de resolver un número creciente de casos de uso, donde se almacene de forma centralizada los datos y se pueda llevar a cabo el análisis adecuado. Este motor debe estar diseñado para tener una escalabilidad horizontal (debido al volumen de la información a tratar), máxima confiabilidad y fácil administración.
3. Una interfaz de usuario que permita visualizar y dar forma a los datos almacenados y analizados.

Además, surge como requisito adicional una sinergia adecuada entre estos componentes, lo que deriva en interfaces simples y compatibles.

4.2.1. Recepción

La recepción debe permitir recibir simultáneamente logs de múltiples WAFs ModSecurity distribuidos. Además, siguiendo las consideraciones básicas para un análisis adecuado estudiadas en 3, debe permitir la normalización de los logs en caso de contar con información heterogénea. Esto deriva en los siguientes requerimientos:

1. Recepción de logs provenientes de WAFs ModSecurity.
2. Normalización de los datos obtenidos.
3. Disposición de los datos normalizados para su posterior almacenamiento.

4.2.2. Almacenamiento

El almacenamiento debe contemplar grandes volúmenes de información. A su vez, también se debe permitir la disposición de la información almacenada teniendo en cuenta procesos de búsqueda y filtrado para su posterior análisis. En esencia se debe:

1. Soportar el almacenamiento para grandes volúmenes de datos.
2. Disposición de los datos almacenados.
3. Permitir la búsqueda y filtrado de los datos almacenados.

4.2.3. Análisis

Para el análisis de la información almacenada se debe contar como mínimo con un motor de búsqueda que trabaje sobre los datos. Con esto se garantiza al menos un análisis manual de los datos en cuestión. Por otra parte, considerando el volumen de información a tratar y la complejidad de los eventos a analizar, es deseable contar con un análisis automatizado sobre los mismos. Esto se puede contemplar insertando un "motor de correlación"que opere en conjunto

con la solución para correlacionar eventos, tanto de un mismo WAF como de múltiples WAFs. Esto permite construir un mayor nivel de conocimiento, con el objetivo de generar alertas o detectar intrusiones que de otra manera se pasarían por alto.

Los requerimientos generales del análisis son:

1. Un motor de búsqueda que trabaje sobre los datos almacenados.
2. Escalabilidad a grandes volúmenes de datos.
3. Motor de correlación: detallado en la sección 4.3.
4. Generación de alertas de acuerdo a los resultados obtenidos del motor de correlación, detallado en 4.5.1.

4.2.4. Presentación

El objetivo de la presentación es contar con una interfaz donde se pueda visualizar los datos almacenados y analizados. Esta presentación debe considerar la visualización de consultas básicas, como por ejemplo cantidad de eventos por WAF, evolución de los eventos en función del tiempo, etc. Para la definición de estos requerimientos se toma como guía los contemplados por la herramienta WAFFLE [18] la cual es una consola open source para ModSecurity. Así, los requerimientos de presentación quedan representados por:

1. Interfaz de usuario para visualizar los datos almacenados y analizados.
2. Un conjunto de visualizaciones predefinidas en base a consultas, estadísticas y correlaciones entre los datos. Especificado en 4.5.2

4.3. Motor de correlación

Según lo detallado en 3, un motor de correlación de eventos debe contar con un conjunto de operaciones necesarias para lograr un análisis automatizado. Las operaciones: **Agregación, Filtrado, Almacenamiento, Normalización, Correlación** y **Reporte** deben ser contempladas en el tratamiento de los eventos a correlacionar.

La **Agregación**, el **Filtrado** y la **Normalización** son acciones previas a la **Correlación**, donde se extraen, filtran y se llevan a un formato estándar definido, los datos a usar. Las acciones de estas etapas son dependientes de cada requerimiento de correlación.

Los requerimientos de correlación a considerar son:

1. Obtener la actividad de una determinada sesión antes y después de recibir un evento proveniente de algún nodo indicando el matcheo de una solicitud con alguna regla de

seguridad establecida. Especificado en 4.5.3.

2. Generar un IOC en base a la actividad obtenida en el punto 1 y correlacionarlo con los eventos registrados en los demás nodos. Especificado en 4.5.4.
3. Correlacionar los eventos generados por determinada sesión (previamente definida) en los múltiples nodos, de manera de establecer que tipo de atacante es. Este análisis se puede realizar en base a un conjunto de axiomas predefinidos que clasifiquen al tipo de atacante, como por ejemplo: se detectó actividad en varios nodos, realizó las mismas acciones en múltiples nodos, realizó determinada cantidad de solicitudes por segundo, etc. Especificado en 4.5.5.
4. Análisis de IOCs conocidos sobre los eventos de cada nodo. Implica también la sincronización con fuentes que brinden indicadores de compromiso abiertos. Especificado en 4.5.6
5. Definir y analizar partes de IOC en busca de ataques por pasos, especificado en 4.5.7.
6. Detectar nuevos ataques en base a comportamientos sospechosos, especificado en 4.5.8.

4.4. Manejo de información

En el marco de la centralización de información de distintas fuentes de datos se identificaron requerimientos relacionados al manejo de los mismos.

1. Mantener privados los datos confidenciales. Especificado en 4.5.9.

4.5. Especificación de requerimientos

En esta sección se especifican los requerimientos analizados anteriormente. El objetivo es brindar una definición más en detalle y orientada a una posible implementación.

4.5.1. REQ ALERTAS

En general, se deben generar alertas cuando se detecten anomalías establecidas. Si bien el tipo de anomalías depende del dominio en el que opera la solución, se pueden especificar algunos de forma general:

1. Se debe alertar cuando el tamaño de los GET/POST tenga un comportamiento anómalo. Esto se puede realizar fijando un umbral para la cantidad de bytes presentes en el evento.
2. Se debe generar una alerta cuando la cantidad de solicitudes de determinado Sistema Autónomo (AS) o país supere cierto umbral.

3. Se deben generar alertas para anomalías detectadas en la correlación distribuida.

4.5.2. REQ VISUALIZACIONES

1. **Eventos por nodo**: despliega un gráfico con la cantidad de **eventos de seguridad** por nodo.
2. **Eventos por estado**: despliega un gráfico con los códigos de respuesta *http* reportados por todos los nodos ordenados cuantitativamente.
3. **Top reglas en determinado tiempo**: despliega un gráfico con los eventos de seguridad reportados por todos los nodos ordenados cuantitativamente. El rango de tiempo abarcado puede ser seteado.
4. **Eventos en determinado tiempo**: despliega un gráfico con la cantidad de eventos de seguridad totales en un determinado tiempo.
5. **Top orígenes en determinado tiempo**: despliega un gráfico ordenando por país de origen a los **eventos de seguridad** registrados.

4.5.3. REQ ACTIVIDAD

Para este requerimiento se debe definir el concepto de "sesión" en base a parámetros de solicitudes HTTP. Algunos ejemplos pueden ser:

- IP, agente y destino
- IP-puerto, agente y destino
- IP, agente y un rango de tiempo

Esta definición debe ser configurable.

Para un evento qué matcheó con alguna regla y fue reportado por un WAF:

1. Se crea una identificador de sesión en base a los parámetros que la definen.
2. Se guarda la sesión en un conjunto de sesiones.
3. Se le asocia el identificador de sesión al evento.

Para cumplir con el requerimiento se crea una consulta por identificador de sesión, de manera de recuperar todos los eventos pertenecientes a la misma.

La información necesaria por este requerimiento es la requerida para identificar la sesión. Además, se deben definir los parámetros que permitan describir la actividad del atacante de la mejor manera. Para esto se definen los parámetros:

- **IP**: dirección IP origen de los eventos.
- **ASN**: número de sistema autónomo al que pertenece la IP.
- **Puerto**: puerto origen.
- **Destinos**: Hosts en los que realizó actividad durante la sesión.
- **User Agent**
- **Franja horaria**
- **Start Timestamp**: hora de la primer actividad detectada.
- **Reglas Matcheadas**: conteo. Se puede discriminar por el atributo *severidad* y por host.
- **Tags**: etiquetas en base a correlaciones o parámetros definidos.
- **Requests por segundo**: cantidad de solicitudes realizada durante la sesión.
- **Tiempo aproximado entre solicitudes**: cálculo realizado sobre los request procesados.
- **Tipo de atacante**: clasificación del atacante en base a atributos definidos. El conjunto de posibles atacantes son: *User, Tool, Crawler*.
- **Información adicional**: *Paths*, *Cookies*.

Este requerimiento se puede cubrir con la información brindada por los logs *access* y *error*. Además se puede alimentar la *Información adicional* con información del registro completo del matcheo con una regla del WAF.

4.5.4. REQ IOC ACTIVIDAD

1. Se obtienen sesiones "finalizadas" en base a un parámetro de tiempo establecido.
2. Para cada sesión finalizada se crea un IOC en base a la actividad registrada en los eventos pertenecientes a la misma.
3. Se escanea la actividad en los demás nodos buscando coincidencias con el IOC generado.

El IOC generado deberá ser definido. La idea de este requisito es identificar comportamientos similares entre actividades de distintos WAFs.

4.5.5. REQ TIPO ATACANTE

Para cada sesión finalizada se debe establecer el tipo de atacante:

- **tool**: definido en la cantidad de solicitudes por segundo.

- **crawler**: definido por el user agent.
- **user**: prorrateado en base al comportamiento o simplemente por defecto.

La decisión debe ser tomada en base a parámetros configurables. Otros posibles criterios que se relevaron para definir el tipo de atacante son:

- **tool**: user agents extraños o fuera de las versiones usuales, cantidad de request por segundos elevado para una misma sesión.
- **crawler**: host origen incluido en una lista de crawlers conocidos, eg: *googlebot.com, baiduspider, search.msn.com, spider.yandex.com, crawl.sogou.com.*
- **user**: un matcheo con SQLi de una consulta "compleja", pequeñas mutaciones en los request.

4.5.6. REQ IOC BASE

Para cubrir este requerimiento se debe contar con IOCs de acceso libre. Cada evento que llegue al centralizador se escaneará con estos IOC. Por lo general, estos tipos de indicadores constan de listas de IP, hashes de archivos subidos, etc.

Cada escaneo que retorne positivo generará una alerta con la información correspondiente al nodo e IOC involucrado.

Dependiendo del nivel de información que se tenga, se podrán usar unos u otros IOCs.

4.5.7. REQ PARTES IOC

Se debe escanear en busca de coincidencias con los IOC, pero de manera descentralizada. Para esto, se deberá intentar buscar coincidencias de partes de los IOC generados en *REQ IOC ACTIVIDAD*, con los eventos reportados por cada uno de los nodos. De esta manera, es posible identificar ataques por partes. Esto es debido a que en un ataque de este tipo, no tendremos el registro del ataque completo, sino solo un subconjunto de los mismos. En este caso, el subconjunto coincidiría con una parte del IOC total.

4.5.8. REQ NUEVOS ATAQUES

Con el objetivo de detectar nuevos ataques se debe correlacionar los datos centralizados en busca de comportamiento sospechoso. La definición de estos comportamientos se puede realizar en base a parámetros configurables o IOC definidos. Como ejemplos de comportamientos sospechosos se relevaron:

- En una sesión no se respeta un 30x en forma sistemática, e.g. 301.
- Requests que demoran un tiempo fuera de un umbral establecido.
- POST a una URL que normalmente se hacen GETs, e.g. URL de favicon.
- Tamaño de las respuestas fuera de un rango fijo para un histórico de una misma URL.
- Identificación de hashes idénticos, en distintos nodos, del contenido de un cuerpo de un mensaje HTTP.

4.5.9. REQ PRIVACIDAD

Los datos confidenciales deben mantenerse privados. Este requerimiento se puede cubrir con el desarrollo de un módulo en el nodo cliente que se encargue de sanitizar la información en base a políticas definidas.

5. Diseño

En esta sección se presentan las decisiones de diseño que se tomaron para modelar los requerimientos relevados y seleccionados para realizar la prueba de concepto. Estas decisiones incluyen el mapeo de estos requerimientos sobre la tecnología ELK, la definición de la arquitectura y el flujo que los eventos van a seguir por la misma. Para esto, se comienza definiendo las tareas más generales como la tecnología y la arquitectura, luego se ahonda en los detalles técnicos necesarios para el mapeo de las funcionalidades sobre el stack ELK.

Por otra parte, se realiza un análisis de la solución *Securely* presentada en la sección 3.7. Como se mencionó, este proyecto tiene un enfoque similar a la solución analizada, por lo que se evalúan posibles opciones para su explotación.

5.1. Tecnología

La tecnología sobre la que realizará el mapeo de los requerimientos es el stack Elasticsearch-Logstash-Kibana (ELK) presentado en la sección 3.2.

Con el uso de esta tecnología se cubre el requisito de Centralización 4.2. Este requisito consta de cuatro partes: recepción, almacenamiento, análisis y presentación.

La **recepción** estará concentrada en Logstash, cumpliendo la necesidad de: recepción de logs provenientes de múltiples WAF's ModSecurity, la posible normalización de los datos y la disposición de los mismos a un componente de almacenamiento.

El **almacenamiento**, que debe contemplar el manejo de grandes volúmenes de información y disponer de los datos almacenados con métodos de búsqueda y filtrado, será manejado por Elasticsearch.

La etapa de **análisis** se puede cubrir con Logstash y Elasticsearch. Ambos componentes tienen la capacidad para contribuir al análisis y correlación de los datos. Esta etapa, también puede ser cubierta mediante el uso exclusivo de Elasticsearch sumado al desarrollo de una aplicación externa. Más adelante se profundizará en esta decisión.

Finalmente, la **presentación** se cubrirá mediante Kibana. Si bien existen otras herramientas para esta tarea, Kibana es quien presenta una sinergia más adecuada entre estos componentes, proporcionando interfaces simples y compatibles.

La figura 1 ilustra el flujo de los eventos en esta tecnología.

5.2. Evaluación de antecedentes

Como se presentó en la sección 3.7, durante el desarrollo de este proyecto se tomó conocimiento de la existencia del proyecto *Securely*. Este proyecto surge debido a una necesidad de automatización en el manejo de eventos de seguridad por parte de analistas y desarrolladores del área.

Como parte del diseño, se evalúan distintas opciones respecto al uso de esta herramienta, como por ejemplo, las de integración o extensión. En este sentido, se realizaron pruebas de las funcionalidades disponibles a través de la versión demo de uso restringido a la que se tuvo acceso.

También se realizó una exploración del código del front-end, entendiendo las estructuras básicas y el flujo de las funcionalidades en este plugin Kibana. La exploración de código se extendió también al back-end conseguido.

Este análisis arrojó las siguientes consideraciones:

- El desarrollo del proyecto está inactivo desde hace varios meses según se puede observar en la actividad de los repositorios correspondientes. Esto puede derivar en que el proyecto quede o esté discontinuado.
- No existe un análisis de requerimientos detrás del desarrollo. Las funcionalidades se implementaron de acuerdo a necesidades locales de los analistas de seguridad. Esto, junto con la no documentación del código, impide la adaptación a la realidad relevada en este proyecto.
- La correlación no tiene en cuenta múltiples WAFs, sino que se basa en un único nodo. Además, esta funcionalidad aún no está madura en el desarrollo actual.
- No existe una documentación adecuada de las funcionalidades ni del desarrollo.

- El tipo de logs considerados no son los mismos que los de este proyecto. El parseo de los eventos tiene como precondición el cumplimiento de determinados patrones “a medida”.
- La normalización de los eventos está condicionada al análisis posterior y no a un modelo dado.

Por la combinación de estas consideraciones se decidió que, para el desarrollo de los requerimientos analizados en este documento, se debe iniciar una nueva implementación.

El uso del proyecto *Securely* queda limitado a la reutilización de código fuente, como pueden ser configuraciones, scripts que realicen tareas puntuales, entre otros. También, el proyecto puede ser utilizado como guía para la estructuración de un desarrollo de este estilo, ya que no es común encontrar proyectos de libre acceso sobre estas tecnologías.

5.3. Arquitectura

La arquitectura de la solución debe asegurar una adecuada interoperabilidad y performance del stack ELK. El flujo usual de un evento es el siguiente:

1. El evento es enviado desde un WAF mediante una herramienta de manejo de logs, por ejemplo, Beats[21].
2. Se recibe el evento desde un pipeline en Logstash para las tareas de parseo, filtrado y su posterior centralización en Elasticsearch.
3. Se guarda el evento en Elasticsearch con el objetivo de centralizarlo y dejarlo disponible para su indexación.
4. El evento es consultado y visualizado desde Kibana.

Notar que el primer ítem del flujo presentado es externo a la frontera de esta solución. De todas maneras, con el objetivo de normalizar la recepción de los eventos, se puede fijar una herramienta y explotar algunas ventajas que brindan, e.g. preprocesamiento y normalización.

Cada una de las etapas de los ítem 2 al 4 deberán realizar las tareas de normalización, agregación, filtrado y correlación. Si bien algunas de estas tareas, como la de parseo o filtrado, están ligadas a la naturaleza de uno de los componentes de stack ELK, en este caso Logstash, otras como la de correlación no tienen una clara correspondencia.

La arquitectura propuesta en este trabajo contempla el flujo descrito para un evento, así como las acciones que se realizan en cada pasaje por un componente del stack ELK y esto se muestra en la figura 4. Notar que la primer capa está fuera de la frontera de esta definición, pero se incluye con la herramienta Beats a modo de ejemplo.

[21] https://www.elastic.co/es/beats

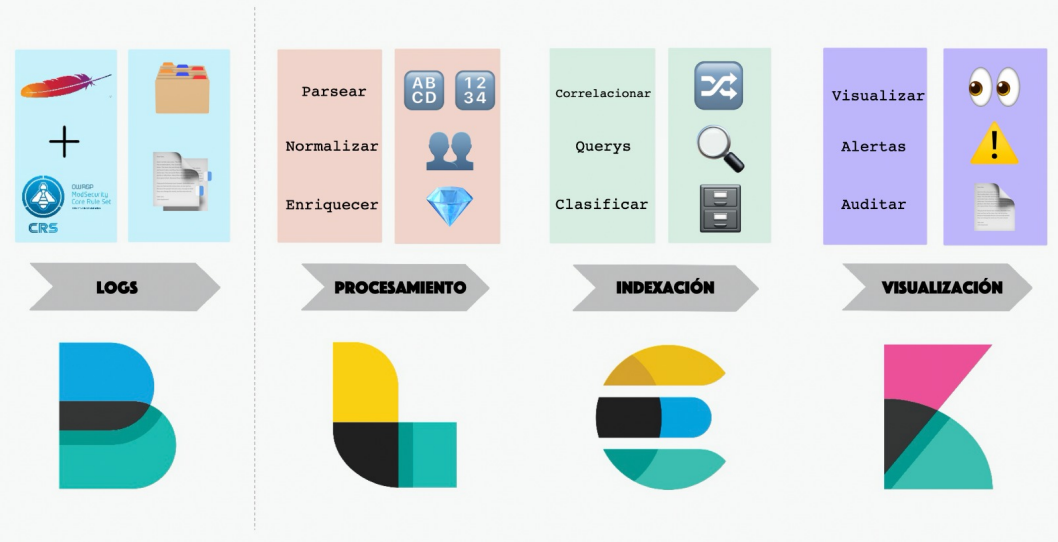

Figura 4: Arquitectura propuesta

5.4. Motor de correlación

El diseño del componente de correlación debe contemplar la incorporación de los requerimientos de correlación presentados, e incorporarlos al flujo de los eventos en su pasaje por el stack ELK. Si se quiere clasificar el diseño de este motor de correlación con las metodologías presentadas en 3.5, se puede decir que es un motor de correlación para un dominio específico (*Domain Awareness*), *Centralized*, por donde se ubica la lógica, *Stateful*, ya que tiene el historial de todos los eventos pasados y con componentes de *Real-time* y *Stored Data*.

La correlación implica el procesamiento en conjunto de los eventos procesados previamente en forma individual. En el procesamiento de los eventos individuales se realizan tareas de normalización y enriquecimiento basado en información contenida en el evento, complementado la misma con información externa. En dicho procesamiento se normalizan los campos del evento a los propuestos en un modelo de datos, se agrega información de interés deducidos de estos campos, entre otros. Por otra parte, el procesamiento en conjunto debe utilizar la información resultante del procesamiento individual de n eventos, para agregar información a cada evento.

El análisis realizado en 4.3 condiciona el diseño del primer requerimiento al concepto de sesión. El diseño de este concepto es quien, en definitiva, establece la arquitectura del motor de correlación. La solución a este problema con la tecnología propuesta puede tomar varios caminos, cada uno con sus ventajas y desventajas. Esto es debido a que cada evento, debe asociarse a una sesión única en base a consultas a otros eventos. Para modelar esto con el stack ELK se encontraron tres posibilidades:

1. **Enriquecimiento en Logstash**: Logstash cuenta con dos plugins que pueden ser utilizados con el objetivo de identificar la sesión para un evento nuevo antes de ser centralizado:

elasticsearch y ruby. Esta solución implica realizar consultas a la base de datos por cada evento procesado.

2. **Plugin en Elasticsearch**: este enfoque implica el uso de procedimientos en un *ingest node*[22] definido en Elasticsearch. Con este enfoque, el enriquecimiento de los eventos es en base a consultas a Elasticsearch y se debe realizar mediante la implementación de un *ingest node* que agregue un procedimiento customizado para este caso. Esta implementación implicaría desarrollar un plugin que extienda ciertas funcionalidades de Elasticsearch.

3. **Plugin en Kibana o app externa**: esta opción usa Elasticsearch mediante su *Search API* para obtener los eventos a analizar y correlacionar. Puede valerse de todas las herramientas disponibles para recuperar datos como agregaciones, consultas Query DLS[23], etc. La información obtenida de la correlación puede ser agregada a los eventos del mismo índice mediante la *Document API* o volcadas en un nuevo índice.

La evaluación de las opciones se realizó contactando directamente a un arquitecto de Elasticsearch[24]. Este intercambio arrojó como primer resultado que las tres opciones son válidas, pero se deben evaluar los pro y contras de cada una de ellas en el contexto que se apliquen. En este sentido:

- La opción 1, busca enriquecer los eventos en Logstash previo a su indexación. Una desventaja de este enfoque es que se van a realizar consultas a la base de datos por cada evento procesado. En cambio, una gran ventaja es la existencia de plugins diseñados para esta tarea, y con una documentación acorde asociada.

- La opción 2 al igual que la 1 busca enriquecer los eventos previo a su indexación, pero esta vez en Elasticsearch. Esto implica el desarrollo de un *ingest node* completamente desde cero en un ambiente con poca documentación.

- La opción 3, implica el desarrollo de una aplicación totalmente externa que utilice la API de Elasticsearch continuamente para consultar y modificar eventos. Esto sin dudas representa un desafío cuando los volúmenes de datos son de gran escala y están en crecimiento constante.

Como información relevante, se encontró que existe una tendencia de migrar algunos de los plugins previamente desarrollados como plugins Elasticsearch a plugins Logstash. Esto es de importancia para no inclinarse a una solución que puede quedar deprecada. En este sentido, y tomando como consideración los pro y contras mencionados anteriormente, el diseño y la prueba de concepto del motor de correlación analizado en este proyecto seguirá la opción 1.

[22]Nodo para preprocesar documentos antes de que ocurra la indexación real: https://www.elastic.co/guide/en/elasticsearch/reference/master/ingest.html

[23]https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

[24]Gabriel Moskovicz, Solutions Architect LATAM & US. https://www.linkedin.com/in/gabriel-moskovicz-2b2b602a

Con este enfoque, el motor de correlación realiza la mayor parte de su trabajo sobre Logstash y Elasticsearch. En Logstash (acudiendo por consultas a Elasticsearch) realiza el etiquetado de las sesiones, mientras que mediante consultas a Elasticsearch desde Kibana realiza tareas de correlación mediante la consulta a eventos ya centralizados.

5.5. Modelo de datos

El modelo de datos presentado pretende fijar un modelo para la normalización de los eventos. Algunos campos del modelo son producto del enriquecimiento luego de correlacionar eventos. La solución contempla los eventos registrados en los archivos *access log*[25].

Así, un evento debe contener al menos los siguientes campos:

- **IP**: dirección IP origen.
- **país**: asociado a la dirección IP origen.
- **franja horaria**: clasificación de la hora en *mañana, medio día, tarde, noche.*
- **ASN**: número de sistema autónomo perteneciente a la IP origen.
- **destino**: dirección IP del host (WAF) que envió el evento.
- **user agent**: este campo contiene la información relacionada al user agent registrado en el evento. e.g. nombre de navegador, versión, sistema operativo. Este campo puede contener toda la información como dato plano o contener un refinamiento estructurado.
- **timestamp**: asociado al tiempo que se produjo el evento.
- **respuesta**: código de respuesta http.
- **tamaño**: contiene el tamaño de la respuesta http en bytes.
- **recurso**: contiene el recurso solicitado.
- **id sesión**: identificador de una sesión.
- **start timestamp**: timestamp del evento más antiguo presente en la sesión.
- **tipo de atacante**: definido en base a atributos del evento.
- **tags**: relacionados a cada evento.
- **mensaje**: evento sin mutar del access log.

Otros atributos que se definen para los resúmenes de los eventos por sesión son:

- **request por segundo**: ratio en base al timestamp de los eventos de la sesión.

[25]https://httpd.apache.org/docs/2.4/logs.html

- **tiempo entre solicitudes**: ratio en base al timestamp de los eventos de la sesión.
- **tipo de atacante de la sesión**: definido en base a de los eventos en la sesión.
- **tags de sesión**: definido en base a los eventos de la sesión.
- **información adicional**: definido en base a los eventos de la sesión.

5.6. Pipeline Logstash

Para la configuración de Logstash se debe especificar un pipeline por donde pasarán los eventos a procesar. La primer etapa es la de **Input** donde se debe configurar la recepción de eventos, indicando puertos de recepción y otros campos dependiendo de la herramienta que los envíe. Luego se debe especificar la etapa de **Filtrado**, donde se realizan las tareas de normalización, enriquecimiento, y filtrado sobre los eventos que se reciban en la primer etapa. Por último, se debe indicar en la etapa de **Output**, la salida que toman los eventos luego del filtrado, por ejemplo, Elasticsearch. La figura debajo ilustra este concepto.

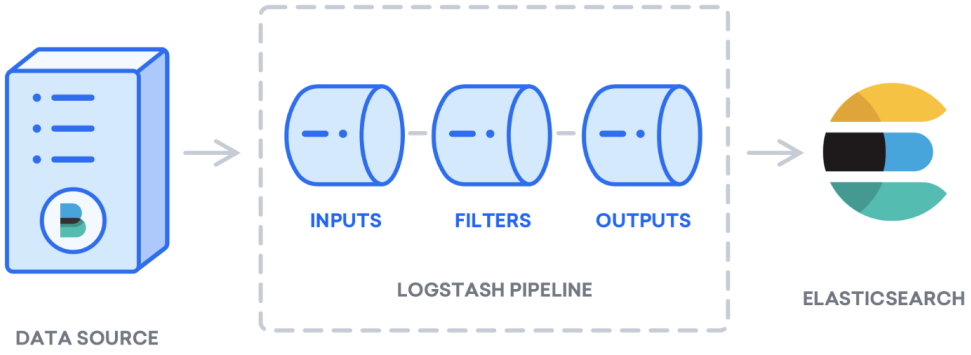

Figura 5: Pipeline Logstash [7]

5.6.1. Input

El pipeline de Logstash permite incluir varias opciones para el input, de manera de ingerir eventos de múltiples fuentes heterogéneas. Para esta prueba de concepto, se considera la carga mediante un archivo local y mediante la recepción de logs enviados desde File Beats. Una de las ventajas de fijar una herramienta como Beats, es que Logstash simplifica la configuración mediante el uso de un plugin de input, así, para este caso la configuración básica se reduce al siguiente código:

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

```
    }

    file {
        path => "ruta-al-archivo-del-access-log"
        start_position => "beginning"
        sincedb_path => "/dev/null"
    }
}
```

5.6.2. Filtrado

El filtrado es la etapa donde se concentrará la mayor parte de la lógica y configuraciones necesarias para el procesamiento de los eventos. Este filtrado se puede modelar en tres etapas: parseo, normalización y enriquecimiento. Cada una de estas etapas implica el uso de distintas funcionalidades de Logstash, y de los plugins disponibles para simplificar estas tareas.

Parseo

Para la tarea de parseo de eventos, Logstash cuenta con el plugin Grok[26]. Este plugin facilita el parseo y la estructuración de, entre otros, los Apache logs.

En este sentido se debe definir un patrón acorde a la configuración del Apache que registra los eventos que arriban. Si la configuración es la por defecto, Grok cuenta con patrones predefinidos que realizan esta tarea.

Este diseño utiliza el patrón `COMBINEDAPACHELOG`.

Normalización

La normalización implica llevar los campos parseados en la etapa anterior a un formato definido, en este caso, por el modelo de datos presentado anteriormente.

Para esta tarea, Logstash incluye el plugin Mutate[27] que permite la transformación de los campos estructurados. De esta manera, se deben transformar los nombres de los campos que deja fijado el plugin de parseo a los existentes en el modelo de datos.

Enriquecimiento Para la etapa de enriquecer los eventos previamente parseados y normalizados, Logstash suma a las funcionalidades del plugin Mutate, el plugin Alter[28].

[26] https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html
[27] https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html
[28] https://www.elastic.co/guide/en/logstash/current/plugins-filters-alter.html

El enriquecimiento implica además, generar campos nuevos y poblarlos con información derivada de otros campos del evento, así como con información derivada de la correlación.

Es por esta razón, que en esta etapa se debe incluir la incorporación de lógica que permita llevar a cabo estas tareas. Para la incorporación de código fuente, Logstash dispone del plugin ruby[29]. Además, dado que la corrección implica la consulta de otros eventos ya centralizados en Elasticsearch, Logstash dispone del plugin elasticsearch[30], que permite cierto tipo de consultas a índices de la base de datos.

Se deben enriquecer los eventos con:

- Información geográfica derivada de su dirección IP.
- Información relativa a la franja horaria correspondiente a la hora de realizado el evento.
- Información relativa a la sesión a la que corresponde el evento.
- Información derivada del evento relativa al tipo de atacante o información sospechosa. Por ejemplo, si la dirección IP está en una lista de IPs sospechosas o pertenecientes a alguna herramienta.
- Información derivada del agente de usuario presente en el evento.

Sesión

Definir la sesión, es enriquecer los eventos con el campo `id sesión`. Esta solución debe seguir algunas reglas de diseño para que cumpla con los requerimientos presentados.

1. Los parámetros que definen a la sesión deben ser configurables. En este sentido el plugin elasticsearch para realizar la consulta correspondiente, debe cargar la misma desde un archivo aparte del de la configuración de filtrado.
2. Si se definen parámetros que no se pueden incluir en la consulta del punto anterior, deben estar en un archivo de configuración principal o como variables globales al inicio del script que realice la lógica sobre los eventos con dicho parámetro. Un ejemplo de esto es un rango de tiempo.
3. El `id sesión` debe ser un único. Una manera de diseñar esto en Logstash es usando un hash MD5 del `mensaje` del primer evento de la sesión. Esta tarea se configura utilizando el plugin fingerprint[31].

[29] https://www.elastic.co/guide/en/logstash/current/plugins-filters-ruby.html
[30] https://www.elastic.co/guide/en/logstash/current/plugins-filters-elasticsearch.html
[31] https://www.elastic.co/guide/en/logstash/current/plugins-filters-fingerprint.html

5.6.3. Output

En este caso se debe configurar el envío de los eventos a un índice Elasticsearch. A continuación, se puede observar un ejemplo de configuración, donde se envían los eventos a un nodo Elasticsearch en la dirección IP `IP`, puerto `9200` e índice `carga_index`:

```
output{
    elasticsearch {
        hosts => ["IP:9200"]
        index => "carga_index"
    }
}
```

5.7. Elasticsearch

En Elasticsearch se deben diseñar las consultas para correlacionar los eventos. Debido a que previamente, en Logstash, se enriquecieron los eventos con un identificador de sesión, las consultas de correlación se diseñarán con agregaciones[32]. Estas agregaciones permiten proporcionar datos agregados basados en una consulta de búsqueda.

Para modelar los requerimientos se utilizarán dos grandes familias de agregaciones:

- *Bucketing*: cada bucket está asociado con una clave y un criterio de documento. Cuando se ejecuta la agregación, todos los criterios de los buckets se evalúan en cada documento (evento) y cuando un criterio coincide, se considera que el documento "cae" en el bucket correspondiente.
- *Metric*: son agregaciones que realizan un seguimiento y calculan métricas sobre un conjunto de documentos. Esta familia de agregaciones se pueden anidar a las anteriores.

Para agrupar los eventos de una misma sesión se utilizará la agregación *Terms*[33] de la familia Bucketing. En una agregación de este tipo los buckets se crean dinámicamente, uno por valor único.

Un ejemplo de como realizar esta consulta en Elasticsearch es la siguiente:

```
{
  "aggs": {
    "agrupar_por_sesion": {
      "terms": {
```

[32]https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html
[33]https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-terms-aggregation.html

```
        "field": "idsesion.keyword"
      }
}
```

Luego de tener agrupados los eventos por sesión se debe correlacionar la información de los mismos. En este sentido y debido a la naturaleza de la información que se quiere obtener (tipo de atacante, solicitudes por segundo, etc.) se debe utilizar la agregación *Scripted Metric*[34]. Esta agregación se ejecuta utilizando un script a medida para brindar una métrica deseada.

Si bien esta agregación es la única (brindada por Elasticsearch) que puede ser utilizada para modelar las métricas planteadas en los requerimientos, tiene sus limitantes dadas por: el lenguaje de scripting utilizado (painless[35]), las estructuras disponibles para mantener un estado común a todos los documentos y la poca documentación y ejemplos al respecto.

El diseño de los scripts debe seguir el siguiente modelo:

```
    "aggs": {
        "nombre de la correlación": {
            "scripted_metric": {
                "init_script": #Permite configurar un estado inicial,
                "map_script": #Ejecutado una vez por documento. Es un script
                               #obligatorio,
                "combine_script": #Ejecutado una vez en cada fragmento después de
                               #completar la recopilación de documentos.
                               #Es obligatorio.
                "reduce_script": #Ejecutado una vez luego de que todos los
                               #fragmentos hayan devuelto sus resultados.
                               #Es obligatorio.
                               #Tiene acceso a una variable 'states' que es una
                               #matriz del resultado del combine_script
                               #en cada fragmento.
            }
        }
    }
```

Para entender mejor como se construyen estas agregaciones a medida, mediante el uso de estos scripts, se muestra el siguiente ejemplo:

```
    "aggs": {
        "alerta de sospechosos": {
```

[34]https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-metrics-scripted-metric-aggregation.html

[35]https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting-painless.html

```
            "scripted_metric": {
                "init_script": state.sospechosos=0,
                "map_script": if(doc['sospechoso'].value()==True)
                {state.sospechosos+=1},
                "combine_script": if(state.sospechosos>100){return True};
                return False,
                "reduce_script": return states
            }
        }
    }
```

En este ejemplo se inicializa una variable en el vector `states` (brindado por la agregación), para luego en la sección de `map_script` actualizarla condicionada al procesamiento de cada evento. En la etapa de `combine_script` se trabaja sobre los objetos del vector `states` (en este caso una variable con un valor) y se deja el resultado en states. Finalmente en `reduce_script`, como solo se tiene un valor, se lo retorna.

5.8. Kibana

En Kibana se modelarán las visualizaciones correspondientes a los requerimientos presentados. Para tener una solución altamente configurable se debe construir un plugin Kibana customizado como el desarrollado en el proyecto presentado en 3.7. Este desarrollo implica la construcción de un software front-end utilizando la API para desarrolladores de Kibana.

Por una parte, se deben modelar las visualizaciones correspondientes a métricas generales de los eventos, y por otra, las derivadas de la correlación.

- **Métricas generales:** en la prueba de concepto realizada para este proyecto se configuran algunas visualizaciones directamente desde la herramienta. Estas visualizaciones incluyen el uso de mapas, gráficas de barra, gráficas lineales, entre otras.
- **Métricas de correlación:** como se mostró en la sección anterior, la correlación está centrada en derivar información de los eventos agrupados en una misma sesión, mediante consultas a Elasticsearch. Por lo tanto, desde el punto de vista de Kibana, la lógica se traduce en realizar estas consultas a Elasticsearch y luego, hacer un refinamiento de los resultados. Un lugar para esta lógica, puede ser el plugin Kibana mencionado. Otra opción, es configurar directamente las consultas en la herramienta y ver los resultados de las respuestas. La prueba de concepto realizada seguirá este último enfoque. Si bien esta opción tiene sus limitantes, el desarrollo de un plugin Kibana implica un proyecto de desarrollo de software front-end, de un porte no menor, que sobrepasa los objetivos de este proyecto.

6. Prueba de concepto

La prueba de concepto llevada a cabo en este proyecto, abarca el despliegue del stack ELK, desarrollando sobre esta tecnología un subconjunto de los requerimientos relevados. Esta tarea se realiza siguiendo el diseño propuesto en la sección anterior. El principal concepto modelado para llevar a cabo la correlación, es el de sesión. Con una sesión definida, se tiene la base para desarrollar los requerimientos del motor de correlación. Los requerimientos de correlación desarrollados son el de tipo de atacante 4.5.5 y el de actividad por sesión 4.5.3. También se implementaron algunos de los requerimientos de visualización presentados en la sección 4.5.2, así como los requerimientos bases de recepción, centralización, etc.

Por otro parte, esta prueba de concepto no incorpora los requerimientos relacionados a los indicadores de compromiso. Tampoco el de privacidad (sección 4.5.9) dado que este se encuentra fuera de la frontera del sistema.

6.1. Ambiente

El ambiente se preparó sobre una máquina virtual con un sistema operativo Ubuntu 18.04 LTS. Para la instalación de los requerimientos necesarios para la prueba de concepto, esta máquina virtual debe tener al menos 4GB de memoria RAM y dos núcleos de procesamiento disponibles.

6.1.1. ModSecurity

Opcionalmente y con el objetivo de generar un conjunto de eventos para realizar pruebas, se instaló Apache y ModSecurity. Una guía detallada para la instalación puede consultarse en [11].

Para preparar el ambiente con esta tecnología se deben seguir los siguientes pasos:

1. Instalar Apache:
   - `sudo apt-get update`
   - `sudo apt-get install Apache2`
2. Instalar ModSecurity:
   - `sudo apt-get install libapache2-mod-security2`
3. Configurar ModSecurity:
   - `sudo cp /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf`

- `sudo vim /etc/modsecurity/modsecurity.conf` aquí se debe colocar la siguiente línea:

```
SecRuleEngine = on
```

4. Inclusión de las reglas CRS:

   - `sudo vim /etc/apache2/mods-enabled/security2.conf`. Aquí se deben agregar las siguientes líneas:

```
IncludeOptional /usr/share/modsecurity-crs/*.conf
IncludeOptional /usr/share/modsecurity-crs/rules/*.conf
```

   Esta es una configuración mínima. Para un mayor detalle acerca de las configuraciones de ModSecurity se puede consultar el documento [**Estado del Arte**]

6.1.2. ELK

La instalación y configuración del stack ELK no es una tarea menor y distintas versiones generan distintas dependencias. Por ejemplo, Elasticsearch admite Java 8 y 9, pero el problema es que Logstash solo es compatible con Java 8, por ende es un prerrequisito para esta instalación. Una guía detallada para la instalación del stack ELK se puede consultar en [7].

1. Instalar y configurar Elasticsearch

   - `wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -`
   - `echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main"| sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list`
   - `sudo apt install elasticsearch`
   - `sudo vim /etc/elasticsearch/elasticsearch.yml`. Aquí se debe configurar:

```
. . .
network.host: localhost
. . .
```

   - Para ponerlo en marcha basta con: `sudo systemctl start elasticsearch`

2. Instalar y configurar Kibana

   - `sudo apt install kibana`
   - `sudo vim /etc/kibana/kibana.yml`. Para un ambiente local, aquí se debe configurar:

```
server.port: 5601
server.host: "localhost"
elasticsearch.url: "http://localhost:9200"
```

- Para ponerlo en marcha basta con: `sudo systemctl start kibana`
- Opcionalmente se puede instalar y configurar Nginx [36] como proxy reverso para Kibana.

3. Instalar y configurar Logstash
   - `sudo apt install logstash`
   - Las configuraciones necesarias para el funcionamiento de Logstash son la creación del pipeline. Este pipeline debe ubicarse en el directorio: `/etc/logstash/conf.d/`.
   - Para iniciar Logstash basta con: `sudo systemctl start logstash`. Esto utilizará los archivos de configuración ubicados en el paso anterior.

6.2. Configuraciones

En esta sección se presentan las configuraciones necesarias para llevar a cabo la prueba de concepto, independientemente del ambiente.

6.2.1. Elasticsearch

En este componente del stack se puede llevar a cabo la creación de un índice Elasticsearch. Esta tarea se puede realizar en la etapa de configuración o directamente se le puede delegar dicha tarea al componente Logstash, configurando un output Elasticsearch y utilizando las funcionalidades que este plugin de output brinda.

6.2.2. Logstash

Para el funcionamiento del pipeline de Logstash se puede generar un único archivo de configuración que establezca las configuraciones de input, filtrado y output. Esta solución es poco modular y dificulta el incremento del código. Por esta razón y debido a que Logstash en su inicio carga todos los archivos de configuración desde un directorio por defecto, se define la siguiente estructuración bajo el mismo:

- **input/**: este directorio contiene los archivos de configuración para la etapa de input. Cada input deberá tener su propio archivo y se nombran de la siguiente manera: `xyy_nombre`

[36] https://www.nginx.com

donde x es un número único para el input e yy denota un número único del archivo dentro del input x. *nombre* es un nombre que debe contener al menos el tipo de input.

- **filter/**: este directorio contiene los archivos de configuración para la etapa de filter. Cada etapa del filtrado se deberá identificar con un número y tendrá sus propios archivos bajo ese número. Los archivos en este directorio se nombran de la siguiente manera: `xyy_nombre` donde x es un número que identifica la etapa: 1 para filtros de inicialización, 2 para filtros de parseo, 3 para los de normalización, 4 para los de enriquecimiento, 5 para correlación, del 6 al 8 sin determinar y 9 para tareas finales antes de terminar con el filtrado, e.g. limpieza de campos temporales. Finalmente, yy denota un número único del archivo dentro del la categoría x y *nombre* un nombre dentro de la misma.
- **output/**: este directorio contiene los archivos de configuración para la etapa de output. Cada output deberá tener su propio archivo y se nombran de la siguiente manera: `xyy_nombre` donde x es un número único para el tipo de output e yy denota un número único del archivo dentro del output x. *nombre* es un nombre que debe contener al menos el tipo de output.

6.3. Funcionalidades

A continuación, se presentan las funcionalidades incluidas en la prueba de concepto desarrollada. En este sentido, se muestra como fueron implementadas las funcionalidades de filtrado (parseo, normalización y enriquecimiento) diseñadas en la sección anterior. A su vez, se detalla la implementación de lo requerimientos de sesión, tipo de atacante, resumen de actividad por sesión, alertas y visualizaciones.

6.3.1. Parseo

Para las tareas de parseo de los eventos se utiliza el plugin Grok. Este plugin cuenta con una gran variedad de expresiones regulares que facilitan el parseo de los eventos entrantes. También cuenta con expresiones regulares para alguno de los formatos de logs más utilizados, como Apache Access log y Syslog.

Para esta prueba de concepto se utilizó uno de los patrones disponibles para Apache access:

```
grok {
  match => [ "message", "%{COMBINEDAPACHELOG}" ]
}
```

Para entender un poco más estos patrones disponibles, se puede consultar el repositorio grok-

patterns[37].

La expresión regular `COMBINEDAPACHELOG` está compuesta por subexpresiones que identifican los campos de un evento presentes en un access log (en su configuración por defecto). Por ejemplo, dirección IP, timestamp, user agent, entre otros.

También, se desarrolló un patrón para los Apache error logs. Aunque en esta prueba de concepto no se consideran este tipo de logs, se desarrolló este patrón que incluye el parseo de la información añadida por *ModSecurity*. Esta expresión puede consultarse en el Anexo 2.

6.3.2. Normalización

La normalización debe llevar los campos creados por el plugin *Grok* a los establecidos en el modelo de datos. Para esto, se utiliza el plugin *Mutate*. Estas tareas se llevan a cabo de la siguiente manera:

```
mutate {
    rename => { "host" => "destino" }
}
```

En este ejemplo, se renombra el campo `host` que contiene la IP destino del evento por `destino`.

Otra tarea que se puede realizar en esta etapa, es la creación de los campos nuevos presentes en el modelo de datos. Por la particularidad de como funcionan los plugins utilizados para el enriquecimiento, esta tarea se puede realizar en dicha etapa.

6.3.3. Enriquecimiento

Algunos campos presentes en el modelo de datos surgen del enriquecimiento. Por ejemplo, los de país, useragent o franja horaria. A continuación, se detalla la construcción de estos campos:

- **país**: para cumplir con este campo se utiliza el plugin *geoip*, que entre otras cosas, deja disponible el país de la IP origen. Esta configuración se realiza de la siguiente manera:

  ```
  geoip {
    source => "IP"
  }
  ```

  Donde se le da como entrada al plugin el campo IP previamente parseado y normalizado. El resultado obtenido es una variedad de campos que deben ser normalizados para respetar el modelo de datos planteado.

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

- **useragent**: para obtener una información estructurada del campo agent parseado por *Grok* se utiliza la siguiente configuración:

```
useragent {
  source => "agent"
  target => "useragent"
}
```

  Esta configuración utiliza el plugin useragent para realizar la tarea. En el ejemplo, toma el campo `agent` y deja la salida en el campo `useragent`.

- **franja horaria**: este campo, a diferencia de los anteriores, es específico de esta solución, por lo que es natural que no exista un plugin pensado para ello. En este sentido se debe enriquecer mediante código Ruby utilizando el plugin para esta tarea:

```
ruby {
  path => "franja.rb"
}
```

  El código presente en el archivo `franja.rb` contiene tres variables configurables para setear los rangos de hora considerados para cada franja: mañana, medio día, tarde y noche. Este código en lenguaje Ruby, obtiene la hora de realizado el evento y lo clasifica en las franjas previamente definidas. Luego, enriquece el campo `franja horaria` con el resultado obtenido.

- **tags**: para cubrir este campo, similar al presentado en el ítem anterior, se utilizó el plugin ruby. En este sentido, se desarrolló un script que verifica si la dirección IP del origen pertenece a una base de datos de IPs asociadas a Tor[38].

- **alerta**: este campo es opcional y se crea en eventos que, al ser procesados por un script Ruby, determinan cierta anomalía. En esta prueba de concepto, se desarrolló un script que verifica si la cantidad de bytes presentes en el log del evento supera un umbral establecido. En caso positivo, crea el campo "alerta" y le coloca un mensaje relacionado a la anomalía detectada.

6.3.4. Sesión

El enriquecimiento de los eventos con un identificador de sesión consta de los siguientes pasos:

1. Se crea un identificador único del evento con el mensaje del mismo utilizando el plugin *fingerprint*. La configuración de esta tarea debe estar en el directorio de filtrado. Los campos más relevantes son:

[38]The Onion Router, web: https://www.torproject.org/

```
fingerprint {
  method => "MD5"
  target => "id_sesion"
}
```

En donde `method` indica el tipo de hash a utilizar y `target` el campo donde se almacena.

2. Se busca en el conjunto de los eventos ya centralizados, filtrando por eventos que cumplan los parámetros establecidos para la identificación de una sesión. Esta tarea se realiza configurando el plugin elasticsearch y agregando código Ruby extra que introduzca cierta lógica de correlación. Para la prueba de concepto, se utiliza el campo *IP origen*, *destino* y un tiempo de 120 segundos desde el inicio del evento, notar que no es posible con este plugin imponer la precondición de los 120 segundos. Un ejemplo de configuración para realizar esta tarea es la siguiente:

```
elasticsearch {
  hosts => ["localhost:9200"]
  query_template => "query.json"
  index => "carga_index"
  fields => { "id_sesion" => "temp_sessid"
              "@timestamp" => "started" }
}
```

Donde `hosts` indica la dirección IP y puerto donde corre la instancia Elasticsearch, `query_template` es un archivo en formato JSON[39] donde se encuentra la consulta a realizar. Esto respeta la decisión de diseño de mantener las consultas en un archivo a parte. Finalmente, `index` indica el nombre del índice donde realizar la consulta y `fields` los campos involucrados. De esta consulta se guarda el `id_sesion` del primer evento recuperado en la consulta y el tiempo de inicio del mismo.

3. Se ejecuta un código Ruby que verifica si el evento actual cumple con el rango de tiempo establecido anteriormente. Para esto, se utiliza el plugin ruby y un código separado que permite que el rango tiempo sea configurable.

4. Si la consulta a Elasticsearch no retorna ningún evento es porque se está ante el primer evento de una sesión. En este caso se setea el tiempo de inicio y el evento continua con el flujo del filtrado. En caso contrario, se sustituye el campo `id_sesión` del evento actual con el valor temporal (que es el identificador de la sesión a la que pertenece). Luego, se eliminan los campos temporales.

[39] JavaScript Object Notation: https://www.json.org/json-en.html

Time	clientip	sessid	franja_horaria
Aug 30, 2019 @ 21:34:08.000	12.8.0.6	54e5c97dafef851758b2ccc6ff2f52d8	noche
Aug 30, 2019 @ 21:34:08.000	12.8.0.8	af5cdfe19ef2fc74f24852f95e397705	noche
Aug 30, 2019 @ 21:34:07.000	12.8.0.6	54e5c97dafef851758b2ccc6ff2f52d8	noche
Aug 30, 2019 @ 21:34:07.000	12.8.0.8	af5cdfe19ef2fc74f24852f95e397705	noche
Aug 30, 2019 @ 21:30:02.000	12.8.0.5	ef4e79e426972334c92002a93883c55f	noche
Aug 30, 2019 @ 21:30:01.000	12.8.0.5	ef4e79e426972334c92002a93883c55f	noche
Aug 30, 2019 @ 21:29:51.000	12.8.0.4	7a1093f38eeff8659b90f9524c3904df	noche
Aug 30, 2019 @ 21:29:50.000	12.8.0.4	7a1093f38eeff8659b90f9524c3904df	noche

Figura 6: Visualización de eventos enriquecidos con el id de sesión y la franja horaria

6.3.5. Tipo de Atacante

El requerimiento de correlacionar la información para obtener el tipo de atacante se implementó en dos etapas. Primero, enriqueciendo el evento con información derivada de sus campos en Logstash. Luego, la tarea es completada analizando el conjunto de eventos en el que se encuentra él mismo (su sesión).

Cada tipo de atacante se implementó como se detalla a continuación:

- **Crawler**:
  1. En Logstash a un evento se lo clasifica como Crawler cuando el campo *useragent* se encuentra en una lista de crawlers conocidos.
  2. En la correlación de eventos pertenecientes a la misma sesión realizada en Elasticsearch, se agrega un script que busca si alguno de los eventos de la sesión tiene *tipo de atacante* seteado en "Crawler".
- **Tool**:
  1. En esta prueba de concepto, inicialmente a un evento se le coloca este tipo cuando el campo *solicitud* es un POST al recurso favicon. El código está modularizado para poder agregar otras posibles heurísticas para clasificar a un único evento como Tool.
  2. En la correlación de los eventos en Elasticsearch, se agregó un script que primero busca si alguno de los eventos de la sesión tiene *tipo de atacante* seteado en "Tool". Además el script calcula los request por segundo. Si estos superan un umbral configurable, se clasifica a la sesión como realizada por una herramienta de ataque Tool.

- **User**: en la correlación de los eventos se agrega un script que verifica que no se hayan detectado indicios para clasificar el ataque como Crawler o Tool. En tal caso, se clasifica al atacante como User.

Cuando los eventos son individualmente procesados en Logstash y no se logra definir un tipo de atacante, se coloca al tipo de atacante como "UNDEFINED YET".

La implementación de la correlación y los scripts mencionados, siguiendo las decisiones de diseño, se llevaron a cabo mediante la utilización de una agregación del tipo *Scripted Metric* anidada a otra agregación de tipo *Bucketing* que agrupa los eventos por sesión. La implementación realizada en el alcance de esta prueba de concepto se encuentra disponible en el Anexo 1.

6.3.6. Resumen de actividad por sesión

El resumen de la actividad por sesión se implementó siguiendo el diseño presentado en 5.7. Utilizando la agregación *Term* y asignando como clave el campo *id_sesion* se obtienen los siguientes resultados sobre un conjunto de eventos de prueba:

```
"sesiones":
    {
      "key" : "fbd2eb97b77ff9cd643ffcf7d492b08f",
      "doc_count" : 9
    },
    {
      "key" : "b0eeaaf57d9f4030bcfc3f0bafd5c17b",
      "doc_count" : 4
    },
    {
      "key" : "09eb6b66c80ee6c92f347e130d0398d1",
      "doc_count" : 2
    },
    {
      "key" : "02cea1adf8871ba47dc06b40b3d90e9f",
      "doc_count" : 1
    }
```

El resumen de la actividad utilizando solamente el *Bucketing* se limita a brindar la cantidad de documentos presentes en cada agrupación (sesión). Luego de esto, se deben usar las *Metrics* para obtener el resumen completo. De esta manera, agregando los *Scripted Metric* a la agregación anterior se pueden obtener resultados de este estilo:

```
"key" : "fbd2eb97b77ff9cd643ffcf7d492b08f",
```

```
"doc_count" : 9,
"Request por segundo" : {
   "value" : 1.0
},
"Tiempo entre solicitudes" : {
   "value" : 1.0
},
"Tipo de atacante" : {
   "value" : "Crawler"
}
```

Donde se correlaciona la información de cada agrupación de eventos por sesión, obteniendo resultados como el tipo de atacante, la cantidad de solicitudes por segundo y el tiempo aproximado entre solicitudes.

6.3.7. Alertas

Para validar el modelado de las alertas, se implementó uno de los requerimientos especificados en la sección 4.5.1. Mediante el enriquecimiento de los eventos individuales y agregando un parámetro configurable en Logstash, se pueden detectar anomalías en estos eventos. La figura 7 muestra un conjunto de eventos que despliegan una alerta al superar un umbral de 8000 bytes. Esta alerta se presenta como un parámetro, por la cual se puede filtrar y consultar.

idsesion	ipcliente	alerta	bytes
54ad415cb0afb82d5c3600eacb350f28	32.106.232.12	SUPERA UMBRAL DE BYTES	8,500
b9ade614500f36571f1924793244a1e3	34.100.4.2	SUPERA UMBRAL DE BYTES	8,511
85097b9963e88746d6837c2897d47d54	12.8.0.2	SUPERA UMBRAL DE BYTES	8,511
168e546bcd5f4528f84375a2f0c479af	193.0.2.19	SUPERA UMBRAL DE BYTES	9,623
b4cb5757c9e5dcceeb9d0597fc48f4b8	193.0.2.15	SUPERA UMBRAL DE BYTES	9,477

Figura 7: Alerta por anomalía en cantidad de bytes

Al filtrar por este tipo de alertas, se puede realizar, por ejemplo, un análisis de los eventos de la sesión con el fin de determinar si la actividad es maliciosa o no.

6.3.8. Visualizaciones

Las visualizaciones modeladas en la prueba de concepto cubren algunas de las métricas relevadas. Además, generan un conjunto de vistas usuales en el contexto de análisis de logs. Estas

visualizaciones complementan la presentación de eventos que brinda Kibana.

Con el fin de mostrar el tamaño de las solicitudes por sesión, se creó la visualización presentada en la figura 8. Un tamaño de solicitud anómalo puede ser indicio de encontrarse frente a un ataque. En este sentido, la figura a continuación muestra el máximo valor en bytes de las solicitudes por cada sesión.

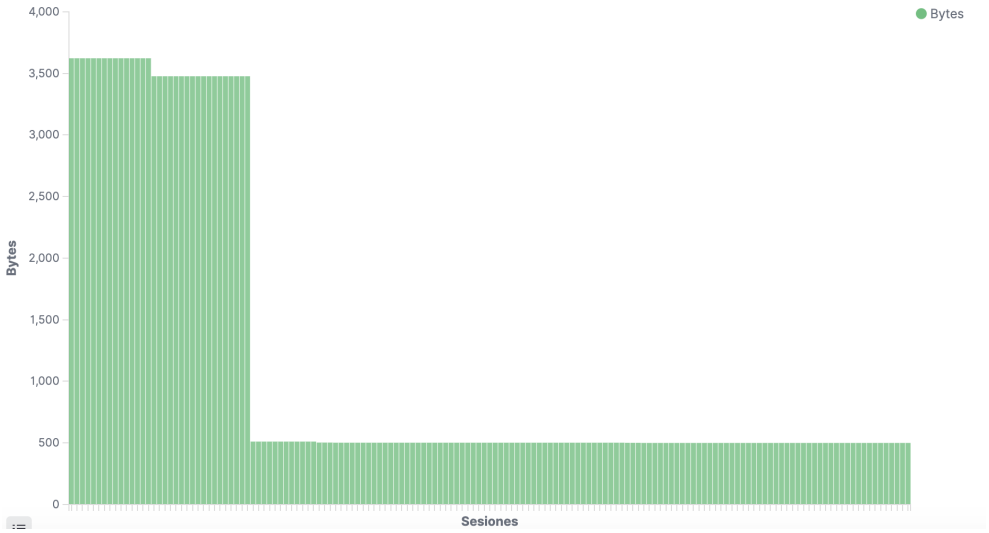

Figura 8: Visualización con la máxima cantidad de bytes por sesión

Otra visualización creada representa la evolución de la cantidad de eventos por sesión, tal como lo muestra la figura 9. Esta visualización permite, por ejemplo, observar sesiones con cantidad de eventos anómalos.

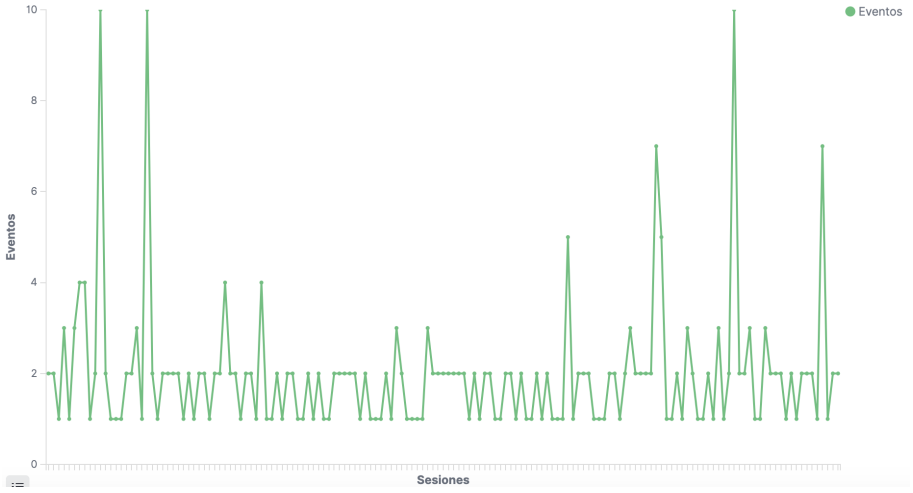

Figura 9: Visualización eventos por sesión

El uso de tipos de datos que permiten la geolocalización (en el pasaje de los eventos a Elasticsearch), permite construir mapas representativos para los eventos. Por ejemplo, el mapa pre-

sentado en la figura 10 representa la ubicación perteneciente a las direcciones IPs origen de los eventos. Este mapa permite agregar cierta información respecto a los eventos localizados en determinado lugar, por ejemplo, se pueden configurar que despliegue el destino (para saber a que WAF pertenece) o el identificador de sesión, entre otros.

Figura 10: Mapa con geolocalización del origen de los eventos

Otras visualizaciones desarrolladas, de las presentadas en 4.5.2, se pueden observar en las figuras 11 y 12. La primera, muestra la cantidad de eventos por WAF en un determinado tiempo, mientras que la segunda despliega un gráfico con los códigos de respuesta http reportados por todos los nodos.

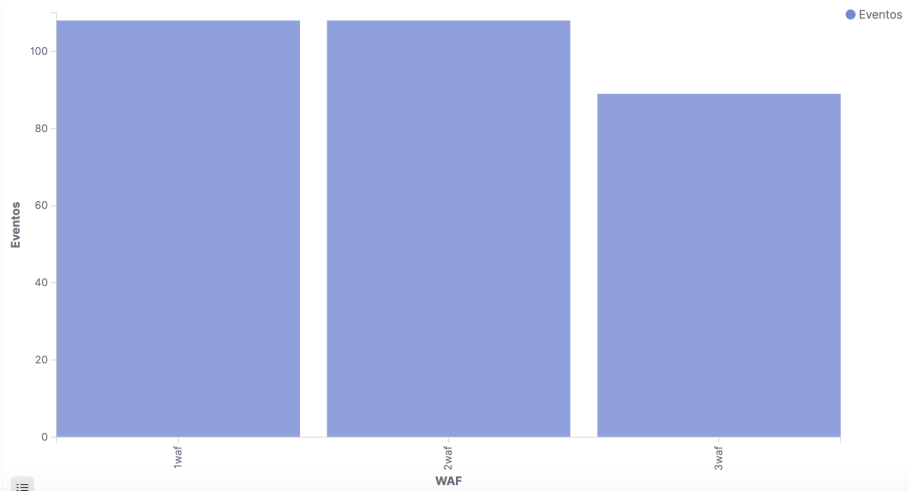

Figura 11: Cantidad de eventos por WAF

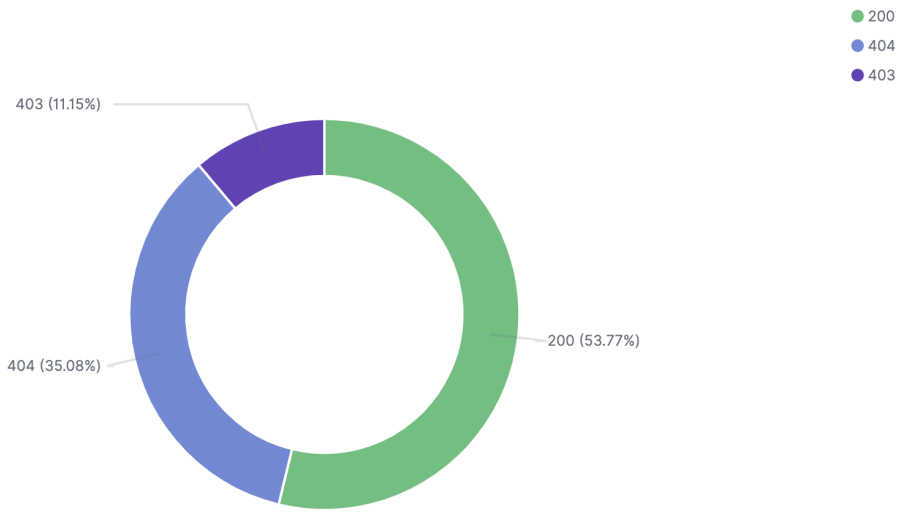

Figura 12: Proporción eventos por códigos de respuesta HTTP

En cuanto a la visualización de la información correlacionada para, por ejemplo, obtener el tipo de atacante o la cantidad de solicitudes por segundo dentro de una sesión, se evaluó el uso de *Kibana Extended Metric Plugin*[40]. Para esto, fue necesario instalar una nueva versión del stack ELK (7.4.2), de manera que esta herramienta sea compatible. Luego de varias pruebas realizadas, se descartó su uso debido a que permite modelar las agregaciones propuestas.

La manera de visualizar la información correlacionada en esta prueba de concepto, es mediante el ingreso de las agregaciones en la funcionalidad "Dev Tool". La figura 13 muestra un ejemplo de consulta mediante esta herramienta.

Figura 13: Visualización de los resultados de correlación

Es posible guardar las consultas con el fin de facilitar el testing a realizar en la prueba de concepto.

[40]https://github.com/ommsolutions/kibana_ext_metrics_vis

Por último, el uso de Kibana facilita la creación de visualizaciones respecto a métricas usuales para los eventos individuales. Estas visualizaciones se pueden realizar dinámicamente según las necesidades del analista.

6.4. Pruebas realizadas

Esta sección presenta las pruebas realizadas sobre la prueba de concepto desarrollada. Se muestran algunos resultados relacionados con las funcionalidades presentadas, con el objetivo de validar las mismas.

6.4.1. Carga de eventos

Se realizó una carga estática de eventos desde tres archivos diferentes, que simulaban tener distintos destinos, es decir, simular que los eventos provienen de tres WAFs diferentes.

Para verificar que la carga fue correcta se puede bien utilizar la API de Elasticsearch, o observarlo directamente desde Kibana. La figura 14 muestra el estado para el índice "logstash-demo" que es el nombre del índice utilizado para la carga de las pruebas. Allí se puede observar el estado, la cantidad de documentos, entre otros.

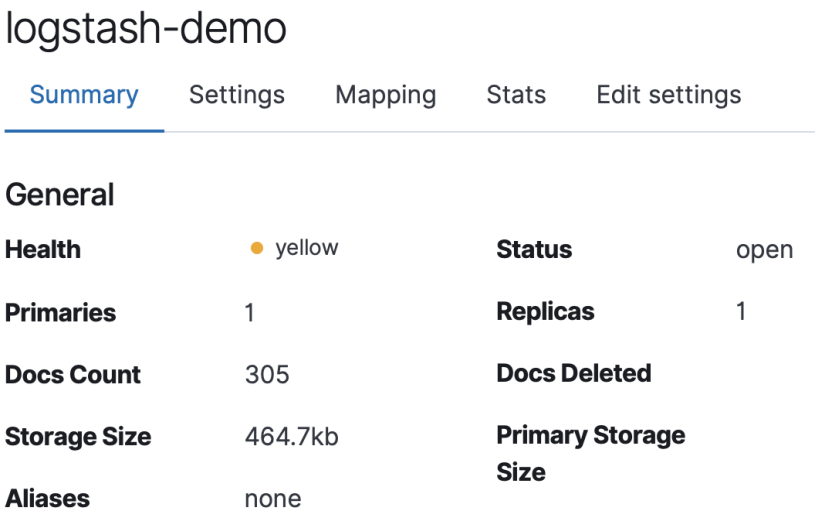

Figura 14: Estado del índice con la carga realizada

Para verificar la correcta normalización de los eventos y la ausencia de errores en la carga, se pueden analizar las salidas en formato JSON de los mismos. La figura 15 muestra una visualización de un evento, donde se pueden observar algunos de los campos del modelo de datos propuesto.

franja_horaria: noche response: 200 timesec: 77,687 timestamp: 30/Aug/2019:21:34:47 -0300 ipcliente: 12.7.0.6 tipo_atacante: TOOL useragent.major: 68 useragent.os: Ubuntu useragent.os_name: Ubuntu useragent.build: useragent.device: Other useragent.minor: 0 useragent.name: Firefox date: Aug 29, 2019 @ 21:00:00.000 destino: 1waf path: /Users/leoalberro/ownCloud/Proyecto de grado/prototipo/eventos-waf1/otros.log httpversion: 1.1 @version: 1 info_adicional: metodo: POST @timestamp: Aug 30, 2019 @

Figura 15: Evento

6.4.2. Etiquetado de sesión

Una sesión está correctamente etiquetada si todos los eventos pertenecientes a la misma cumplen las siguientes condiciones:

1. La dirección IP origen es la misma.
2. El nodo destino es el mismo.
3. Están en un rango de 120 segundos.

El punto 2 surge debido a que estamos modelando una solución donde se centralizan eventos de múltiples WAFs. La figura 16 muestra un resumen de una sesión, donde el rango de tiempo, la IP origen y el destino cumplen con las condiciones establecidas.

Time	idsesion	ipcliente	destino
Aug 30, 2019 @ 21:34:47.000	432aa655a15ed514b2cd6367a3dafb25	12.7.0.6	2waf
Aug 30, 2019 @ 21:34:37.000	432aa655a15ed514b2cd6367a3dafb25	12.7.0.6	2waf
Aug 30, 2019 @ 21:34:17.000	432aa655a15ed514b2cd6367a3dafb25	12.7.0.6	2waf
Aug 30, 2019 @ 21:34:16.000	432aa655a15ed514b2cd6367a3dafb25	12.7.0.6	2waf

Figura 16: Etiquetado de eventos por sesión

6.4.3. Eventos por sesión

El requerimiento de actividad 4.5.3 se puede ver como el "resumen" de una sesión. En esta prueba se filtran los eventos por un identificador de sesión. Estos eventos pueden resumirse en una visualización como la mostrada en la figura 17. En esta visualización se muestran algunos campos seleccionados entre los disponibles y se cuenta con la funcionalidad de Kibana de examinar cada uno de ellos mediante el despliegue del evento que se desee.

Time	ipcliente	metodo	idsesion	start_timestamp	pais
Aug 30, 2019 @ 21:24:49.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:29.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:48.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:42.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:59.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:46.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:45.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:39.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:34.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States
Aug 30, 2019 @ 21:24:31.000	12.5.0.1	GET	fbd2eb97b77ff9cd643ffcf7d492b08f	Aug 30, 2019 @ 21:24:29.000	United States

Figura 17: Visualización de eventos pertenecientes a una misma sesión

6.4.4. Tipo de atacante por evento

Luego de la carga de los eventos desde Logstash, se puede observar en Kibana como cada evento individual fue etiquetado con un tipo de atacante.

	Time	idsesion	tipo_atacante
>	Aug 30, 2019 @ 21:29:51.000	81296704f68ec53af0d34d12563e7ec0	UNDEFINED YET
>	Aug 30, 2019 @ 21:29:38.000	e788078d005831e1a290a8aa9135efcd	UNDEFINED YET
>	Aug 30, 2019 @ 21:29:39.000	7e925f40864f45f95b46f21ff296ac73	UNDEFINED YET
>	Aug 30, 2019 @ 21:24:42.000	fbd2eb97b77ff9cd643ffcf7d492b08f	UNDEFINED YET
>	Aug 30, 2019 @ 21:34:37.000	b0eeaaf57d9f4030bcfc3f0bafd5c17b	TOOL
>	Aug 30, 2019 @ 21:34:17.000	f6f859b6bee629e14a7c027a77f56feb	CRAWLER

Figura 18: Listado de eventos individuales con el campo *tipo_atacante*

.

6.4.5. Tipo de atacante

El *tipo de atacante* final con el que se clasifica una sesión, se prueba mediante la consulta a Elasticsearch con las agregaciones definidas anteriormente.

```
"key" : "fbd2eb97b77ff9cd643ffcf7d492b08f",
"doc_count" : 9,
"Tipo de atacante" : {
```

```
 "value" : "User"
 }

"key" : "b0eeaaf57d9f4030bcfc3f0bafd5c17b",
"doc_count" : 4
"Tipo de atacante" : {
 "value" : "Tool"
}

"key" : "f6f859b6bee629e14a7c027a77f56feb",
"doc_count" : 2,
"Tipo de atacante" : {
 "value" : "Crawler"
}
```


En el resultado obtenido para una consulta, como se muestra arriba, se puede observar la clasificación de las sesiones en *tipo de atacante*. Como era de esperar, la sesión mostrada en la figura 21 es clasificada como *Crawler* y es debido a que los eventos individuales estaban clasificados de esta manera. Similar a esto, en la figura 20 se puede observar como algunos de los eventos de esta sesión estaban clasificados como *Tool* y por esta razón, la sesión también. Finalmente, la figura 19 muestra una sesión donde los eventos no contienen información acerca del tipo de atacante. Esta información, sumada a que las solicitudes por segundo no superaron el umbral establecido, determinaron que el tipo de atacante para esta sesión sea *User*.

	Time	tipo_atacante	idsesion
>	Aug 30, 2019 @ 21:24:42.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:45.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:34.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:48.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:39.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:59.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:31.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:46.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f
>	Aug 30, 2019 @ 21:24:29.000	UNDEFINED YET	fbd2eb97b77ff9cd643ffcf7d492b08f

Figura 19: Listado de eventos filtrado por sesión con el campo *tipo_atacante*

	Time	tipo_atacante	idsesion
›	Aug 30, 2019 @ 21:34:37.000	TOOL	b0eeaaf57d9f4030bcfc3f0bafd5c17b
›	Aug 30, 2019 @ 21:34:47.000	TOOL	b0eeaaf57d9f4030bcfc3f0bafd5c17b
›	Aug 30, 2019 @ 21:34:17.000	TOOL	b0eeaaf57d9f4030bcfc3f0bafd5c17b
›	Aug 30, 2019 @ 21:34:07.000	UNDEFINED YET	b0eeaaf57d9f4030bcfc3f0bafd5c17b

Figura 20: Listado de eventos filtrado por sesión con el campo *tipo_atacante*

	Time	tipo_atacante	idsesion
›	Aug 30, 2019 @ 21:34:17.000	CRAWLER	f6f859b6bee629e14a7c027a77f56feb
›	Aug 30, 2019 @ 21:34:27.000	CRAWLER	f6f859b6bee629e14a7c027a77f56feb
›	Aug 30, 2019 @ 21:34:37.000	CRAWLER	f6f859b6bee629e14a7c027a77f56feb
›	Aug 30, 2019 @ 21:34:07.000	CRAWLER	f6f859b6bee629e14a7c027a77f56feb

Figura 21: Listado de eventos filtrado por sesión con el campo *tipo_atacante*

7. Conclusiones

Este proyecto fue creado con el fin de brindar un análisis a una solución que contempla la recepción, análisis y centralización de *logs* de seguridad provenientes de múltiples WAFs ModSecurity. El objetivo principal es el de validar su posible construcción e incorporación a las herramientas actualmente disponibles para estas tareas. Estas herramientas resultan de gran importancia en el área de seguridad y análisis de logs, debido a que de no contar con las mismas, el trabajo de los analistas de seguridad se vería desbordado rápidamente.

En este trabajo entonces, se elaboró una solución que contempla la recepción, análisis y centralización de *logs* de seguridad provenientes de múltiples WAFs. Esta solución fue validada en una prueba de concepto que resultó en una herramienta open source, altamente configurable y por lo tanto customizable, que deja una base y un precedente para construir soluciones de este estilo, sobre una plataforma ELK. Estas características permiten automatizar rápidamente muchas de las tareas a las que los analistas de seguridad se enfrentan en el día a día. Además, el hecho de que este tipo de herramienta sea altamente customizable, permite desplegar la misma en un ambiente dinámico como lo es el análisis de logs de seguridad. Este dinamismo se debe a que los atacantes día a día crean formas cada vez más sofisticadas de realizar acciones maliciosas sobre los sistemas informáticos. Por lo tanto, contar con una herramienta con esta características resulta menester para poder adaptar la misma a cada caso particular y a nuevos ataques que se presenten.

Para esto, el proyectó fue conformado por diferentes etapas, teniendo en primer lugar el estado del arte, luego el análisis, seguido del diseño, y por último, el desarrollo de la prueba de concepto.

La prueba de concepto, valida en primer lugar, el uso de la plataforma ELK como centralizador y analizador de eventos de seguridad provenientes de WAFs ModSecurity distribuidos. En particular, se pudo comprobar que es posible el modelado de los requerimientos claves vinculados a la correlación distribuida de estos eventos. En cuanto a la implementación de requerimientos asociados a esta correlación, es posible concluir que:

- mediante el enriquecimiento de los eventos en *Logstash* es posible llevar a cabo tareas de correlación para eventos individuales. Un ejemplo de esto es la asociación de un evento a una sesión.
- mediante la construcción de consultas adecuadas para *Elasticsearch*, se puede construir un análisis apropiado para la correlación de eventos centralizados y provenientes de múltiples fuentes. Realizar este análisis fue posible aún cuando el soporte de Kibana para desplegar estos resultados no es el adecuado.

En cuanto a las dificultades afrontadas a lo largo del proyecto, resulta de gran interés analizar las más relevantes, con el objetivo de aportar una visión más general sobre el trabajo realizado y los desafíos del mismo.

- El principal desafío dentro de la etapa de estado del arte, se centró en el estudio y entendimiento a fondo de ciertos temas, entre ellos, diferentes técnicas para el manejo de eventos de seguridad (indicadores de compromiso), y técnicas para la correlación de eventos, que en principio resultaban ser desconocidos para el autor.
- Al inicio de la etapa de análisis, se enfrentaron problemas relacionados al relevamiento de requisitos. A pesar de poder identificar claramente varios requisitos, fue necesario acudir a un relevamiento externo de manera de poder validar y recabar más información al respecto. De esta manera, fue posible superar la dificultad mencionada, obteniendo requisitos completos, consistentes y precisos. Esto es de gran importancia para ahorrar problemas y retrabajo en el resto de las etapas.
- Al momento de tomar decisiones sobre el diseño de la arquitectura, fue necesario consultar a un arquitecto experto en la tecnología *Elasticsearch*, con el fin de obtener argumentos sólidos para la toma de estas decisiones. En particular, la tarea que llevó más tiempo de abordar y resolver, fue determinar la distribución de la lógica correspondiente a la correlación de eventos sobre los distintos componentes del stack ELK.
- Por último, en la etapa de desarrollo de la prueba de concepto se encontraron varias dificultades, principalmente relacionadas al modelado de la correlación. La carencia de documentación adecuada para el lenguaje de *scripting* y las agregaciones utilizadas, resultó en la extensión del tiempo inicial estimado para la tarea, elevándolo a casi el doble. Por otro lado, *Kibana* no permite la construcción de cualquier tipo de visualización personalizada, a menos que se desarrolle un *plugin* a medida. Esto imposibilitó la representación de los resultados provenientes de las agregaciones, como visualizaciones estándar de *Kibana*.

Uno de los puntos a agregar en referencia a la dificultad del desarrollo es que, si bien, centralizar los eventos dentro de una herramienta como *Elasticsearch*, puede resultar ser una tarea rápidamente abordable, cuando se pretenden aplicar técnicas para la normalización y correlación de eventos, la dificultad aumenta significativamente. Esto es debido a que las funcionalidades "a medida" que se requieran añadir a este *stack open source* para realizar estas tareas, en muchas ocasiones no cuentan con soporte, o directamente no es posible modelarlas con *plugins* ya incorporados. Esto último, puede derivar en la necesidad de desarrollar estos complementos íntegramente o utilizar los existentes de manera no convencional.

En lo que respecta a la distribución del tiempo, las tareas de este proyecto se realizaron con algunas modificaciones respecto al plan inicial. De los doce meses de trabajo, dos fueron utilizados para la investigación del estado del arte, tanto de los temas a abordar, como posibles técnicas a aplicar. Luego, para el análisis de la solución, se tomaron cuatro meses. Esta etapa excedió el tiempo estimado inicialmente, debido a la relevancia que tiene en este proyecto y los problemas que surgieron durante su desarrollo. Finalmente, el diseño y la prueba de concepto construida utilizaron el tiempo restante hasta la finalización del proyecto.

Por otra parte, la construcción de documentación acompañó todo el desarrollo del proyecto. Además del reporte principal, se generaron documentos individuales para las etapas de estado del arte y análisis de la solución: [**Estado del Arte**] y [**Análisis**] respectivamente.

Como trabajo a futuro, este proyecto deja varias líneas. En cuanto al modelado de los requerimientos analizados, se podría trabajar en la incorporación de aquellos que involucran a los indicadores de compromiso. Esta tecnología y los requerimientos asociados, fueron investigados en el estado del arte y analizados en la especificación de requerimientos, pero no se ahondó en la integración con la herramienta.

Otra posible línea de trabajo podría ser abordar la incorporación de más y mejores alertas. En este sentido, se puede evaluar integración de la herramienta con algún framework como ElastAlert[41] en Kibana.

También es clara la necesidad de desarrollar un proyecto front-end que cree las visualizaciones adecuadas para la lógica de correlación incorporada. Esta tarea fue evaluada, pero el alcance de la misma supera al de este proyecto. La falta de este desarrollo se ve reflejada en las salidas en formato JSON mostradas para la correlación que obtiene el resumen de una sesión.

Además, resulta evidente la necesidad de montar pruebas en un ambiente más afín a uno de producción. Esto se debe principalmente a que esta solución fue evaluada funcionalmente, sobre escenarios montados en una infraestructura reducida. Para cumplir con los objetivos reales del proyecto, y que esta herramienta tenga un valor real en el área de análisis de eventos de seguridad, es necesario medir indicadores de performance en un ambiente como el mencionado anteriormente. Con ese objetivo, también se debe considerar la incorporación de otros tipos de logs, con los que se trabaja en la operativa real, como lo son los Apache error logs.

Como reflexión final, es posible concluir que se logró cumplir con los objetivos del proyecto. Los resultados obtenidos, pueden ser de gran interés para la comunidad de seguridad, y, debido al componente ingenieril del proyecto, puede tener una rápida aplicabilidad en la operativa habitual de los analistas de seguridad que se enfrentan a problemas de análisis de *logs* de seguridad en su día a día.

En último lugar, es importante destacar que durante el desarrollo de este proyecto, se incorporó una gran cantidad de aprendizajes, tanto técnicos como de crecimiento personal. Esta combinación, en definitiva, son las que deben acompañar a un ingeniero a lo largo de su formación.

[41] https://github.com/Yelp/elastalert

Referencias

[1] Jugal Kumar Bhattacharyya, Dhruba Kumar; Kalita. *DDoS attacks: evolution, detection, prevention, reaction, and tolerance*. CRC Press, 2016.

[2] Onur Catakoglu. *Using Web Honeypots to Study the Attackers Behavior*. PhD thesis, TELECOM ParisTech, 2017.

[3] Anton Chuvakin, Kevin Schmidt, and Chris Phillips. *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*. Syngress Publishing, 2013.

[4] Justin Clarke. *SQL Injection Attacks and Defense*. Syngress, 2 edition, 2012.

[5] DavidJBianco. Enterprise Detection & Response - The Pyramid of Pain. `www.detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html`. Último acceso: 2019.

[6] Instituto Nacional de Seguridad de España. La «otra manera» de identificar malware. `www.incibe-cert.es/blog/indicadores-de-compromiso`. Último acceso: 2019.

[7] DigitalOcean. How To Install Elasticsearch, Logstash, and Kibana. `https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-18-04`. Último acceso: 2019.

[8] J.E. Doak, J.B. Ingram, S.A. Mulder, J.H. Naegle, J.A. Cox, J.B. Aimone, K.R. DIxon, C.D. James, and D.R. Follett. Tracking cyber adversaries with adaptive indicators of compromise. In *Proceedings - 2017 International Conference on Computational Science and Computational Intelligence, CSCI 2017*, pages 7–12, 2018.

[9] Allan Liska; Timothy Gallo. *Ransomware*. O'Reilly Media, Inc., 2016.

[10] Dieter Gollmann. *Computer Security*. Wiley, 3 edition, 2011.

[11] HostAdvice.com. ModSecurity - How to Setup ModSecurity for Apache on Ubuntu 18.04. `www.hostadvice.com/how-to/how-to-setup-modsecurity-for-apache-on-ubuntu-18-04/`. Último acceso: 2019.

[12] K. Kent and M. P. Souppaya. *Sp 800-92. guide to computer security log management*. U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 2006.

[13] Z. Martinasek, P. Blazek, P. Silhavy, and D. Smekal. Methodology for correlations discovery in security logs. In *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 294–298, Nov 2017.

[14] ModSecurity. Open Source Web Application Firewall. `www.modsecurity.org`. Último acceso: 2019.

[15] Andreas Müller. Event Correlation Engine. Master's thesis, Swiss Federal Institute of Technology, Zurich, 2009.

[16] Kanchanmala Bharamu Naukudkar, Dayanand D. Ambawade, and J. W. Bakal. Enhancing performance of security log analysis using correlation-prediction technique. In Suresh Chandra Satapathy, Amit Joshi, Nilesh Modi, and Nisarg Pathak, editors, *Proceedings of International Conference on ICT for Sustainable Development*, pages 635–643, Singapore, 2016. Springer Singapore.

[17] U. Noor, Z. Anwar, A.W. Malik, S. Khan, and S. Saleem. A machine learning framework for investigating data breaches based on semantic analysis of adversary's attack patterns in threat intelligence repositories. *Future Generation Computer Systems*, 95:467–487, 2019.

[18] WAF-FLE Project. An OpenSource ModSecurity Console. `http://www.waf-fle.org`. Último acceso: 2019.

[19] Byron Alfonso Carrión Ramírez. Diseño e Implementación de una solución de gestion centralizada de logs de aplicaciones, sistemas y dispositivos basada en Logstash que permita la creación de cuadros de mando para explorar, analizar y monitorear eventos de seguridad. Master's thesis, UOC, UAB, URV, UIV, 2015.

[20] Chad Russell. *Web Application Firewalls*. O'Reilly Media, Inc., United States of America, 2018.

[21] Emitzá Guzmán Tobias Röhm Benoit Gaudin Newres Al Haider Sergio Zamarripa López, María del Carmen Calle Villanueva. Monitoring Control for Remote Software Maintenance. Technical report, ICT, 2010.

[22] M. Verma, D.P. Kumarguru, S.B. Deb, and A. Gupta. Analysing indicator of compromises for ransomware: Leveraging iocs with machine learning techniques. In *2018 IEEE International Conference on Intelligence and Security Informatics, ISI 2018*, pages 154–159, 2018.

8. Anexo 1

En este anexo se detallan las configuraciones necesarias para poner en marcha el sistema propuesto. Se toma como base el siguiente directorio:

```
POC
  |___correlaciones
  |___pipeline___ _input
  |               |_filter
  |               |_output
  |___ruby-scripts
  |___thor
  |___web-crawlers
  |___pipeline-demo.conf
  |___README.md
```

Debido a que las rutas de configuración de Logstash pueden variar dependiendo del sistema operativo, el funcionamiento se explicará con rutas relativas o ejemplos instanciados a un sistema en particular. Esta herramienta además, se puede utilizar de dos maneras: como servicio y como programa (pasándole las configuraciones como parámetros).

Para poner en marcha el stack ELK se debe entonces:

- Levantar el servicio de Elasticsearch.
- Levantar el servicio de Kibana.
- Comprobar que los servicios estén levantados, ya sea mediante comandos del sistema operativo o intentando acceder mediante el navegador. Por defecto: Kibana `localhost:5601`, Elasticsearch `localhost:9200`.
- Configurar los parámetros del archivo `ruby-scripts/constans.rb`. Esto incluye, por ejemplo, las rutas a la base de datos de los crawlers o el valor de constantes, como el tiempo máximo de una sesión.
- Si usamos un único archivo de configuración podemos iniciar Logstash directamente con el ejecutable: `sudo bin/logstash -f pipeline-demo.conf`
- Si se usa un pipeline modularizado, se debe usar Logstash como un servicio. Para esto se debe copiar el directorio `pipeline` en el path adecuado, por ejemplo en Ubuntu `/etc/logstash/conf.d`. Luego de esto iniciar el servicio de Logstash.

Algunas consideraciones respecto a la carga de los eventos:

- Si se utiliza el archivo único `pipeline-demo.conf`, se debe editar el archivo indicando la ruta al archivo de los eventos.

- En ambos casos se debe tener cuidado con la configuración del índice Elasticsearch. Tanto en la sección de output (donde se indica el índice Elasticsearch al que se volcarán los eventos), como en el plugin de filtrado elasticsearch hacen referencia al índice. El índice debe ser configurado y no dejarlo a la creación por defecto debido a que la correlación necesita saber el nombre del índice que debe utilizar para identificar una sesión.

Para ejecutar y visualizar las correlaciones que se encuentran en el directorio con este nombre, se deben copiar las mismas en Kibana. Para esto la Kibana brinda una herramienta llamada "Dev Tools" donde permite la interacción con Elasticsearch. Una vez copiadas estas agregaciones quedan guardadas en la herramienta.

9. Anexo 2

9.1. Métrica para el tipo de atacante

```
"atacante": {
    "scripted_metric": {
        "params": {"umbral":1},
        "init_script" : """
            state.tipo_por_evento=[];
            state.count=0;
            state.tiempos=[];
                        """,
        "map_script" : """
            state.tipo_por_evento.add(doc['tipo_atacante.keyword'].value);
            double time;
            if(doc['timesec'].size()==0){time = 0}else{time = doc['timesec'].value }
            state.tiempos.add(time);
            state.count+=1
                        """,
        "combine_script" : """
            double max = state.tiempos[0];
            double min = state.tiempos[0];
            for (x in state.tipo_por_evento) { if(x == 'CRAWLER'|| x == 'TOOL')
            {return x} }
            for (t in state.tiempos){if(t>max){max=t} if(t<min){min=t}}
            if(state.count/(max-min) < params.umbral){return 'TOOL'}
            return 'USER'
                                """,
        "reduce_script" : """
            String res = '';
            for (a in states) { res = a }
            return res
                                """
    }
}
```

9.2. Métrica para la cantidad de solicitudes por segundo

```
"request por seg": {
    "scripted_metric": {
        "init_script" : """
            state.count = 0;
            state.tiempos = []
                        """,
        "map_script" : """
            double time;
            if(doc['timesec'].size()==0){time = 0}else{time = doc['timesec'].value}
            state.tiempos.add(time);
            state.count+=1
                        """,
        "combine_script" : """
            double max = state.tiempos[0];
            double min = state.tiempos[0];
            for (t in state.tiempos){if(t>max){max=t} if(t<min){min=t}}
            return state.count/(max-min)
                            """,
        "reduce_script" : """
            double res = 0;
            for (a in states) { res = a } return res
                        """
    }
}
```

9.3. Patrón para parsear Apache Error logs

```
APACHE_ERROR_TIME %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}
APACHE_ERROR_LOG \[%{APACHE_ERROR_TIME:timestamp}\] \[:%{LOGLEVEL:loglevel}\]
(?:%{PID:pidinfo}) (?:\[client %{IP:clientip}:%{POSINT:port}\]){0,1}
%{GREEDYDATA:errormsg}
PID \[pid %{NUMBER}:tid %{NUMBER}\]
```