Universidad Zaragoza
1542

# Trabajo Fin de Grado

## Diseño, implementación y operación de un servicio SOC basado en software libre

## Design, implementation and operation of a SOC service based on open source software


Autor

Francisco Javier Pizarro Martínez

Director

Miguel García Giménez



ESCUELA DE INGENIERÍA Y ARQUITECTURA

2023

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Francisco Javier Pizarro Martínez ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Grado en Ingeniería Informática (Título del Trabajo)

Diseño, implementación y operación de un servicio SOC basado en software libre.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 29 de noviembre de 2023

Fdo:

Dedicado a todos aquellos que han estado tanto en las buenas como en las malas,
así como a todos los que me han apoyado para llegar hasta aquí.

# Diseño, implementación y operación de un servicio SOC basado en software libre

## RESUMEN

La empresa NOLOGIN ofrece distintos servicios para cubrir los aspectos relacionados con la seguridad de los servicios informáticos de sus clientes. Para defender una infraestructura crítica se emplean soluciones Security Operations Center (SOC). Dado que la empresa solo dispone de soluciones comerciales, surge la necesidad de diseñar e implementar una nueva solución basada estrictamente en herramientas *open source*.

El problema radica en que existen muy pocas herramientas *open source* ya implementadas, las cuales solo aportan soluciones parciales a la problemática general del SOC. Es por ello que el primer gran reto de este proyecto consiste en diseñar e implementar una solución distribuida conformada por varios servicios distribuidos. El segundo gran reto de este proyecto es que, para demostrar la efectividad del mismo, se deben elegir diversos escenarios representativos de la situación actual que sirvan como prueba de concepto (PoC) para las capacidades defensivas de la solución propuesta. Estos escenarios han condicionado la toma de decisiones durante el proyecto.

El enfoque utilizado para resolver ambos retos entiende que una solución SOC es un conjunto de servicios correctamente desplegados y configurados, los cuales son operados por personas para defender de forma activa la infraestructura y los servicios correspondientes. Es por ello que tras estudiar el estado del arte, se ha decidido crear un SOC basado en tres pilares centrales: Security Information and Event Management (SIEM), Security Orchestration, Automation and Response (SOAR) y Endpoint detection and response (EDR). Cada elemento de los anteriores tiene su propio despliegue automatizado, distribuido, escalable (de forma horizontal y vertical) y tolerante a fallos (HA/DR). La solución cubre los aspectos de disponibilidad, autenticación, autorización y confidencialidad, adicionalmente también otorga unas mejores capacidades de respuesta al equipo humano encargado de operar el SOC.

Para obtener una solución de calidad, se han definido unas políticas de seguridad que esta debe cumplir. La arquitectura propuesta está conformada por Graylog como SIEM, Shuffle como SOAR y Wazuh como EDR. Los escenarios de prueba de concepto planteados representan de forma fidedigna la situación actual de muchas empresas, y todos ellos han sido resueltos de forma muy efectiva con la solución propuesta.

La solución final ha logrado sobrepasar los objetivos planteados inicialmente, desde los elementos que conforman la misma hasta los escenarios que demuestran el alcance de esta, así como en cuanto a los requerimientos especiales de la solución (escalado, tolerancia a fallos). Durante el proyecto se ha contribuido en el ámbito *open source* sobre el que se ha construido él mismo, más concretamente enriqueciendo la herramienta SOAR empleada en este TFG.

# Design, implementation and operation of a SOC service based on open source software

## ABSTRACT

The company NOLOGIN offers various services to cover aspects related to the security of its clients' IT services. In order to defend critical infrastructure, they use Security Operations Center (SOC) solutions. Since the company only has commercial solutions, there is a need to design and implement a new solution strictly based on open-source tools.

The problem is that there are very few pre-implemented open-source tools, and those available only provide partial solutions to the overall SOC problem. Therefore, the first major challenge of this project is to design and implement a distributed solution composed of several distributed services. The second major challenge is to choose various scenarios representative of the current situation to demonstrate the effectiveness of the solution as a proof of concept (PoC) for the defensive capabilities of the proposed solution. These scenarios have influenced decision-making throughout the project.

The approach used to address both challenges understands that a SOC solution is a set of properly deployed and configured services operated by individuals to actively defend the infrastructure and corresponding services. After studying the state of the art, it was decided to create a SOC based on three central pillars: Security Information and Event Management (SIEM), Security Orchestration, Automation and Response (SOAR), and Endpoint Detection and Response (EDR). Each of these elements has its own automated, distributed, scalable (both horizontally and vertically), and fault-tolerant deployment. The solution covers aspects of availability, authentication, authorization, and confidentiality. Additionally, it provides improved response capabilities to the human team responsible for operating the SOC.

To achieve a quality solution, security policies that it must comply with have been defined. The proposed architecture consists of Graylog as SIEM, Shuffle as SOAR, and Wazuh as EDR. The proof-of-concept scenarios presented faithfully represent the current situation of many companies, and all of them have been effectively resolved with the proposed solution.

The final solution has surpassed the initially set objectives, both in terms of its components and the scenarios demonstrating its scope, as well as meeting the special requirements of the solution (scaling, fault tolerance). Throughout the project, contributions have been made to the open-source community upon which it is built, specifically enhancing the SOAR tool used in this project.

# Índice

# Índice de figuras

# Índice de tablas

# Glosario

**Security Operations Center (SOC)**: Un centro de operaciones de seguridad es una instalación donde se monitorean, evalúan y defienden los sistemas de información empresariales.

**Casos de uso (CDU)**: En este contexto particular hace alusión a los tipos de ataque que están contemplados en el sistema y que cuentan con su correspondiente procedimiento de detección y tratamiento.

**High Availability (HA)**: Alta disponibilidad, es la cualidad que tiene un sistema informático si en caso de que un nodo falle, el sistema es capaz de seguir plenamente operativo.

**Disaster Recovery (DR)**: Recuperación de desastres, en caso de un fallo general del sistema, este debe poder ser recuperado en muy poco tiempo sin tener pérdidas significativas de información en el mismo.

**Ingeniería social**: Metodología de ataque, esta se basa en lograr engañar a una o más personas para conseguir que estas realicen una o más acciones potencialmente peligrosas.

**Phishing**: Ataque basado en ingeniería social, normalmente, este consiste en el envío de un correo electrónico haciéndose pasar por otra entidad/persona para conseguir los credenciales o información asociada al usuario atacado.

**Fuerza bruta**: Metodología de ataque basada en probar de forma exhaustiva credenciales para lograr obtener un acceso no autorizado, se puede realizar contra paneles de login o servicios directamente o contra el Hash de unos credenciales para a base de probar distintas opciones encontrar aquella, la cual una vez aplicado la función correspondiente de Hash se obtenga el mismo Hash. Para una mayor efectividad en este tipo de ataque se suelen emplear Wordlists que contienen las contraseñas más habituales, la más famosa de estas es RockYou.

**SSH (Secure Shell Protocol)**: Servicio que se encuentra habilitado en la gran mayoría de servidores Linux, este permite al personal informático autorizado obtener un acceso remoto de consola sobre dichos servidores para llevar a cabo las acciones pertinentes.

**DDoS (Distributed Denial of Service)**: Ataque de denegación de servicio distribuido, este consiste en que un gran número de equipos informáticos (usualmente botnets) bombardeen un servidor a peticiones con el objetivo de saturar este, atentando así contra la disponibilidad del mismo o en su defecto contra la calidad del servicio.

**Insider**: Forma de denominar a una persona que trabaja dentro de una empresa y trata de atacar esta desde dentro.

**Data Leaks**: Es la exfiltración de información de carácter confidencial sin permiso, pueden ser intencionales o accidentales dependiendo de cómo se originen.

**Rubber Ducky**: Herramienta empleada por los hackers, esta aparenta ser un USB normal no obstante en su interior cuenta con una SD que permite guardar código escrito en el lenguaje DuckyScript, dicho código será ejecutado en la máquina que se introduzca dicho USB de forma inmediata, para lograr esto el USB finge ser un teclado a ojos de la máquina a la que ha sido conectado.

**SQL Injection**: Ataque común contra servidores web, este se aprovecha de debilidades en el backend a la hora de generar queries para la base de datos correspondiente. Para explotar dicho ataque se introducen cadenas SQL mal formadas intencionalmente, de forma que se obtenga un acceso no autorizado a información.

**TOR (The Onion Router)**: Red especial de nodos empleada para anonimizar el tráfico de internet, funciona como una cebolla y sus capas. Dentro de esta red se encuentran páginas webs con contenidos ilegales, dichas webs no pueden ser accedidas desde fuera de esta red dado que tienen en sus dominios la extensión .onion. Para acceder a dicha red de forma completamente transparente se emplea el navegador TOR. También se puede usar dicha red de más formas, para el envío o recepción de información de forma anónima.

**Disponibilidad**: Garantizar que los sistemas, servicios y datos se encuentren funcionando en condiciones adecuadas, de forma que cuando un usuario los necesite estos provean un servicio óptimo y no fallen/se encuentren ralentizados.

**Confidencialidad**: Propiedad asociada normalmente a archivos, consiste en preservar las restricciones autorizadas de acceso y divulgación de la información.

**Integridad**: Protección contra la modificación o destrucción indebidas de la información.

**Autentificación**: El acto de probar la identidad declarada ante un sistema/servicio.

**Autorización**: Define lo que está permitido o no realizar para una identidad concreta para la cual previamente se ha realizado de forma satisfactoria un proceso de autentificación.

**CSIRT (Computer Security Incident Response Team):** Término empleado para aludir al equipo humano responsable de la respuesta a incidentes de seguridad.

# Capítulo 1

# Introducción

La ciberseguridad es una faceta muy importante dentro de la informática dado que sustenta el correcto funcionamiento de todos los sistemas informáticos y permite la detección en tiempo real de ataques o de potenciales amenazas, defendiendo el sistema atacado cuando se detectan. Esto permite asegurar un funcionamiento adecuado del sistema, así como garantizar los 3 pilares básicos de la ciberseguridad: confidencialidad, integridad y disponibilidad.

Otra parte importante de la ciberseguridad consiste en automatizar la respuesta a estos ataques y amenazas de forma que al recibir una alerta, se realicen de forma automática e inmediata las acciones necesarias para contener y subsanar el problema.

## 1.1 Contexto y motivación

Este proyecto se ha llevado a cabo en la empresa NOLOGIN, una consultora que se dedica a gestionar los centros de datos y las soluciones de ciberseguridad de otras empresas donde ésta juega un papel crítico. Algunos de sus clientes son administraciones públicas, como el Gobierno de Aragón (AST) o empresas multinacionales en las que la seguridad juega un papel crucial, como Ferrovial. NOLOGIN, a pesar de estar principalmente localizada en Zaragoza, tiene importantes y significativos clientes en varios países.

Mediante el desarrollo de este proyecto se pretende ampliar la oferta de soluciones ofrecidas por la empresa, de forma que esta nueva solución "lowcost" pueda ser empleada por clientes preocupados por su seguridad y con un bajo poder adquisitivo.

En este proyecto, el enfoque utilizado entiende que una solución SOC es un conjunto de servicios correctamente desplegados y configurados, los cuales son operados por personas para defender de forma activa la infraestructura y los servicios correspondientes.

## 1.2 Objetivos

El objetivo global de este proyecto es diseñar, implementar y operar un Security Operations Center (SOC)[4][5] basado en software libre que sirva como un nuevo producto dentro de las soluciones que ofrece la empresa aumentando así el público objetivo de la misma. El sistema debe ser capaz de monitorizar el correcto funcionamiento del sistema a proteger, detectando las anomalías o ataques que aparezcan, generando las correspondientes alertas y sus consiguientes respuestas automáticas y avisos.

Dado que este sistema va a ser la base de la solución SOC, este debe tener ciertas cualidades especiales. La primera de ellas es que, dado que es vital para garantizar la seguridad del cliente, este debe ofrecer *alta disponibilidad* (HA) y en su defecto una *recuperación de desastres* (DR).
Así mismo, como cada cliente es diferente y esta es una solución genérica, debe tener la capacidad de escalar de forma que, aunque la infraestructura del cliente sea excesivamente grande y compleja de monitorizar, la solución pueda crecer y ajustarse para aguantar dichas cargas de trabajo.

Otro aspecto a tener en cuenta es que, dado que este sistema se va a implantar en una gran cantidad de entornos, el proceso de despliegue de la solución que contiene las distintas herramientas debe ser automático y rápido. Un último aspecto derivado del último mencionado es que además de tener que automatizar el despliegue, las herramientas subyacentes deben permitir importar y exportar toda la operativa relativa a la implantación de los casos de uso (CDUs). El primer objetivo del proyecto es, entonces, diseñar e implementar una solución que cumpla con lo anteriormente planteado.

El segundo objetivo del proyecto consiste en la operación de la infraestructura del SOC una vez se encuentre implementado para crear los mecanismos de defensa pertinentes y proteger la organización ante los distintos escenarios de ataques representativos elegidos. Dichos escenarios servirán como PoC (Prueba de Concepto) de las capacidades del SOC. A su vez, cada escenario deberá tener un CDU asociado para su segurización, y estos deben comprender todo el ciclo defensivo. Es decir, desde la detección de la amenaza hasta su eliminación y hacer lo pertinente para que la misma no se repita.

## 1.3 Estructura

La memoria comienza con un glosario que recoge los términos técnicos más importantes empleados a lo largo de esta. En segundo lugar, en la introducción se centra el contexto del problema y los objetivos a alcanzar en la solución. Inmediatamente después en el estado del arte se explica el estado actual de los SOC y de los conceptos de soluciones que pueden ser utilizadas en estos. También estudia las herramientas comerciales y las alternativas *open source*.

Una vez se ha definido el contexto del problema y se han estudiado las herramientas que lo resuelven parcialmente, se ha realizado una fase de análisis en la cual se analizan de forma profunda los requisitos exigidos por cada una de las partes del proyecto, las herramientas analizadas como posibles soluciones, también se encuentran los escenarios de alto nivel a ser cubiertos y los CDUs concretos asociados.

Con todos los requisitos y escenarios de prueba definidos la siguiente fase es el diseño del sistema, en esta se detalla la arquitectura deseada del sistema, cómo interactúan los distintos componentes entre sí y los casos de usos contemplados en la solución.

Dentro de la sección de implementación se explica de forma detallada la implementación del sistema, así como el despliegue del mismo, cuenta también con la implementación de los CDUs. Una vez estuvo todo implementado se realizó la correspondiente fase de experimentación, en esta se encuentran todas las pruebas significativas sobre un despliegue lo más cercano posible a la realidad dentro de un entorno controlado de pruebas.

Para finalizar, en el apartado de conclusiones se reflexiona acerca del grado de finalización del proyecto, así como de su posible futuro.

Existen dos secciones adicionales que cuentan con las fuentes de información y con detalles adicionales, estas son la bibliografía y los anexos respectivamente.

# Capítulo 2

# Estado del arte

**“A security infrastructure will drive an attacker to the weakest link.”**
La evolución tecnológica ha hecho que la ciberseguridad sea vital en un mundo interconectado. Inicialmente, la seguridad en sistemas informáticos era deficiente y estaba ausente a nivel de diseño de aplicaciones y protocolos. Con la aparición de servicios más relevantes, la conciencia sobre la necesidad de sistemas informáticos seguros ha crecido gradualmente en la sociedad. Desde la aparición de los SOCs hace 18 años, su importancia se ha incrementado de forma significativa. Este incremento se debe principalmente a la necesidad primordial de prevenir incidentes de ciberseguridad, lo que ha llevado a la adopción de los centros de operaciones de seguridad en las distintas empresas. A pesar de su popularidad, la cantidad de material académico al respecto es escasa.

*Otro problema en este estado del arte es que carece de una buena definición a nivel global, dado que hay más de una forma de interpretar cómo debe funcionar un SOC. Para algunos investigadores, un SOC es únicamente una entidad responsable de supervisar la red. Para otros, es una unidad organizativa que abarca todas las operaciones de seguridad, como la gestión de incidentes y la inteligencia sobre amenazas. (Vielberth, 2020, Introduction)[1]*

En la actualidad la mayoría de soluciones de ciberseguridad han evolucionado hasta contar con una variedad de sistemas que conforman el SOC. Algunos de los sistemas más relevantes en el panorama actual son los siguientes: Los SIEM se encargan de centralizar la información y alertar en caso de ataque (alerta), los SOAR se encargan de automatizar los procesos de respuesta a los ataques para agilizar así el ciclo defensivo (respuesta), y por último para brindar una capa extra de protección a los Endpoint se emplean los denominados EDR, los cuales actúan como una versión más avanzada de lo comúnmente denominado antivirus (prevención). A continuación, en el apartado de conceptos, se detalla con precisión qué es cada uno de estos sistemas y su funcionamiento.

## 2.1 Conceptos

A continuación se presentan una serie de conceptos que explican las diferentes tipologías de las herramientas más comúnmente empleadas dentro de una solución SOC.

**Security Information and Event Management (SIEM)**: Dispositivo que aglomera los logs generados en las infraestructuras del cliente, los preprocesa, etiqueta, parsea y clasifica, y una vez hecho esto correla dichos logs, es decir busca si se cumplen ciertas características tanto intrínsecas a un log concreto como extrínsecas al mismo como puede ser la relación con otros logs. Posteriormente en caso de haber encontrado una correlación concreta genera una respuesta asociada que puede consistir en notificar dicho evento de cierta forma o en avisar de dicho evento a otros componentes de la solución para que estos tomen las medidas oportunas.

**Security Orchestration, Automation and Response (SOAR)**: Dispositivo que se encarga de realizar las respuestas automáticas asociadas a los eventos generados en otros elementos del sistema. Para generar dichas respuestas realiza los cambios necesarios en las máquinas correspondientes, ya sea mediante alguna herramienta de CM (Configuration Management) o mediante interacciones por medio de API/RPC con elementos especiales del sistema como pueden ser los cortafuegos.

**Endpoint detection and response (EDR)**: Herramienta compuesta por unos servidores y unos agentes que se instalan en los Endpoints o sistemas finales. Monitoriza la actividad en los Endpoints, pudiendo detectar ataques tanto por comportamientos sospechosos del sistema como por las firmas de comportamiento. Además, un EDR es capaz de responder de forma inmediata con acciones avanzadas ante un ataque. Ofrece capacidades para realizar una investigación forense de los incidentes.

**Intrusion detection system (IDS)**; Herramienta compuesta por un servidor central y unos sensores virtuales. Se emplea para detectar intrusiones, para ello el servidor central agrega y procesa la información recibida por los sensores. Esta herramienta **no** está preparada para tomar acciones defensivas por sí misma.

**Intrusion prevention systems (IPS)**: Herramienta muy similar al IDS, la diferencia principal radica en que esta **si** puede tomar acciones defensivas para prevenir el ataque, son especialmente útiles para mitigar problemas de red.

## 2.2 Herramientas

Se ha elegido emplear como piezas de la solución las herramientas SIEM, SOAR y EDR, es por ello que para poder posteriormente realizar un buen análisis de requisitos primero hay que conocer la situación actual de sus distintas implementaciones comerciales. Para elegir herramientas efectivas se ha recurrido al estudio de informes profesionales del sector[10][11][12].

Como implementaciones concretas de referencia para la comprensión del funcionamiento de cada tipo de herramienta se han elegido los siguientes fabricantes:

- SIEM: IBM
- SOAR : Palo Alto Networks
- EDR: Palo Alto Networks

Las justificaciones de las elecciones, como herramientas de referencia, se incluyen en el Anexo B.

Por otro lado, dentro de las herramientas *open source* encontramos las siguientes:

- SIEM: Graylog, ELK, Fluentd y Wazuh
- SOAR*:* Catalyst y Shuffle
- EDR*:* OpenEDR, Ossec y Wazuh

# Capítulo 3

# Análisis

Antes de poder comenzar a diseñar la solución, es necesario definir tanto los requisitos de cada una de las herramientas que la componen como seleccionar los distintos escenarios de ataques más relevantes de hoy en día.

Para la extracción de requisitos se ha analizado el funcionamiento de las herramientas comerciales tomadas como referencia. Adicionalmente, dado que existen una serie de requisitos transversales para todas las herramientas, se han definido unos requisitos generales.

## 3.1 Requisitos del sistema

Dado que el proyecto tiene varias herramientas claramente diferenciadas, en cada una de ellas se ha realizado el correspondiente estudio de las necesidades del sistema y a partir de este se ha elaborado una tabla de requisitos. Dentro de los requisitos, los que más se han valorado han sido aquellos que afectan de forma directa y severa a la funcionalidad, es decir, aquellos que no se pueden subsanar por medio de alguna alternativa o herramienta adicional.

Antes de definir los requisitos particulares de cada herramienta es necesario definir los requisitos generales que todas ellas deben cumplir, por ello a continuación se adjuntan algunos puntos clave para elaborar dichos requisitos generales.

Un factor clave a la hora de la elección ha sido la calidad de la documentación de las herramientas, ya que si se pretenden instalar en clientes finales, se necesita una información completa y detallada del fabricante para garantizar la operación y el correcto mantenimiento. De la misma manera, deben de ser herramientas en continuo desarrollo para ofrecer continuidad al cliente.

Puesto que a priori se desconoce el entorno a securizar, la solución debe ser tan adaptable y escalable como sea posible, además dado que el entorno es desconocido, este puede ser muy homogéneo o completamente heterogéneo. Teniendo en cuenta esto último, la solución debe funcionar correctamente con sistemas Unix, Linux y Windows.

Otro factor muy importante de cara a los organismos encargados de auditar que efectivamente un sistema ha sido o no vulnerado, es conservar todos los datos relevantes de este sin pérdida alguna de información a pesar de que en alguna máquina se produzcan fallos incluso a nivel de hardware. Por ello surge la necesidad de que el sistema cuente al menos con HA, o en su defecto DR.

| REQUISITOS GENERALES | |
|---|---|
| Identificador | Requisito |
| RG1 | El sistema debe ser capaz de funcionar tanto en entornos Linux/UNIX como en entornos Windows. |
| RG2 | El sistema debe ser capaz de escalar de forma horizontal o vertical. |
| RG3 | El sistema en su versión distribuida, debe ser tolerante a fallos (HA/DR). |
| RG4 | El sistema debe contar con una buena documentación. |
| RG5 | El proyecto *open source* del sistema debe seguir en desarrollo activo. |

*Tabla 3.1.1 Listado de requisitos generales*

En los requisitos anteriores, dado que son generales, no se ha querido distinguir entre funcionales y no funcionales. Para extraer los requisitos particulares de cada herramienta se han extraído de las capacidades esenciales de las soluciones comerciales estudiadas. El detalle del funcionamiento de las herramientas comerciales tomadas como referencia se encuentra en el Anexo B.

### 3.1.1 SIEM

A partir de los objetivos planteados y de la solución comercial de *Security Information and Event Management* estudiada se han extraído los siguientes requisitos, que deben ser cumplidos por la herramienta *open source* elegida:

| REQUISITOS SIEM | |
|---|---|
| Identificador | Requisito |
| RF1 | El sistema debe ser capaz de admitir tantos formatos de logs como sea posible. |
| RF2 | El sistema debe ser capaz de realizar tagging/categorización sobre los logs preprocesados. |
| RF3 | El sistema debe ser capaz de parsear la información de los logs taggeados mediante expresiones regulares. |
| RF4 | El sistema debe ser capaz de enriquecer los logs parseados. |
| RF5 | El sistema debe ser capaz de tener reglas de búsqueda complejas. |
| RF6 | El sistema debe ser capaz de correlar distintos logs entre sí. |
| RF7 | El sistema debe ofrecer la posibilidad de customizar dashboards. |
| RF8 | El sistema debe ser capaz de notificar tanto al analista como de generar un evento para notificar al SOAR. |
| RF9 | El sistema debe ser capaz de acceder a máquinas que guardan logs de forma local y extraer dichos logs. |

*Tabla 3.1.1.1 Listado de requisitos del SIEM*

### 3.1.2 SOAR

Del informe profesional, que explica cuáles deberían ser las cualidades y funcionamiento de un SOAR[11], en conjunto, con lo analizado en la alternativa comercial de *Security Orchestration, Automation and Response* y teniendo en cuenta los objetivos planteados, se extraen los siguientes requisitos:

| **REQUISITOS SOAR** | |
|---|---|
| Identificador | Requisito |
| RF1 | El sistema debe permitir crear llamadas a API custom para interaccionar con otros elementos. |
| RF2 | El sistema debe ser capaz de notificar al SOC mediante correo electrónico. |
| RF3 | El sistema debe tener un registro del histórico de workflows ejecutados. |
| RF4 | El sistema debe permitir crear workflows con estructuras condicionales en su interior. |
| RF5 | Los workflows deben permitir realizar paradas en su interior para una revisión manual de casos importantes por analistas. |
| RNF1 | El sistema debe tener las llamadas a APIs de los sistemas más comunes ya implementadas. |

*Tabla 3.1.2.1 Listado de requisitos del SOAR*

### 3.1.3 EDR

Tomando como referencia el funcionamiento de la herramienta comercial elegida como *Endpoint detection and response*, así como los objetivos planteados, se ha elaborado la siguiente tabla de requisitos:

| **REQUISITOS EDR** | |
|---|---|
| Identificador | Requisito |
| RF1 | El sistema debe ofrecer capacidad de integración con el resto de la solución. |
| RF2 | El sistema debe ser capaz de monitorizar el contenido de ficheros. |
| RF3 | El sistema debe ser capaz de detectar la conexión de dispositivos externos. |
| RF4 | El sistema debe permitir un análisis granular de los logs, para seguir la traza de una ejecución maliciosa. |
| RF5 | El sistema debe ser capaz de aislar el host en el que se encuentre instalado. |

*Tabla 3.1.3.1 Listado de requisitos del EDR*

## 3.2 Escenarios de prueba de concepto

Se establecen también una serie de escenarios que orientan los requisitos para la solución propuesta. Durante la selección se ha tenido en cuenta que estos deben ser una muestra lo suficientemente representativa de diversos tipos de ataques para demostrar la efectividad y el alcance real de la solución propuesta. Cada uno de los escenarios propuestos tendrá asociado el correspondiente caso de uso dentro del SOC.

Para elegir los escenarios más representativos de la situación actual se ha estudiado el panorama de la misma mediante el análisis de varios informes profesionales del sector[2][3]. Adicionalmente se han tenido en cuenta políticas internas de la empresa. Aquí se encuentra un listado de los distintos escenarios (a alto nivel) que deberán ser cubiertos por la solución, así como la justificación de la razón de su elección:

- Ataques de ingeniería social

  Según el reporte de 2022 de Verizon Data Breach Investigations Report (DBIR)[2], el 74% de las brechas de seguridad involucran un factor humano y más del 50% incluyen un factor de ingeniería social.
- Ataques de fuerza bruta

  Uno de los ataques más sencillos, consiste en el acceso no autorizado a recursos mediante el uso de un gran número de credenciales hasta lograr acceder al sistema.
- Accesos a potenciales amenazas

  El eslabón más débil en un contexto de ciberseguridad siempre va a ser el factor humano, es por ello que para brindar una capa extra de protección, se ha decidido cortar todo acceso a amenazas conocidas.
- Ataques contra la disponibilidad

  A diferencia de otros tipos de ataque que pretenden vulnerar la confidencialidad de la información, esté atenta contra la disponibilidad de los servicios, por lo que afecta directamente a la imagen empresarial, al coste y a su facturación. Es un tipo de ataque relevante que debe ser contemplado como tal.
- Ataques de insiders

  Otro punto débil de las organizaciones es ante los denominados “insiders”, empleados que por desconocimiento o por intenciones propias realizan ataques o robos de información desde dentro.
- Ataques contra servidores Web

  La mayoría de empresas hoy en día cuentan con distintos servidores webs, por lo que es extremadamente común tratar de explotar vulnerabilidades en estos.
- Accesos a contenidos restringidos

  Existen contenidos confidenciales a los que un empleado normal no debería acceder, es por ello que es necesario restringir dichos accesos.
- Ataques de intrusión física

  Otro punto débil en muchas entidades es la prevención de ataques basados en privilegios obtenidos a partir de una intrusión física en las instalaciones.
- Utilidades CSIRT (equipo humano que emplea el SOC como respuesta)

  Un SOC también debe mejorar la capacidad de respuesta del CSIRT.

Dado que estos escenarios son abstractos y el caso de uso concreto elegido para representar cada uno va a condicionar el diseño del sistema, aquí se adjunta la tabla de correspondencias entre los escenarios planteados y los casos de uso que los representan. Adicionalmente inmediatamente después de la tabla se adjunta la explicación concreta asociada a los 2 CDUs de utilidad del CSIRT.

| Escenario | Caso de uso |
|---|---|
| Ataques de ingeniería social | Análisis de Phishing |
| Ataques de fuerza bruta | Fuerza bruta a SSH |
| Accesos a potenciales amenazas | Accesos a webs con URLs maliciosas conocidas |
| Ataques contra la disponibilidad | Ataque DDoS |
| Ataques de insiders | Data Leaks USB + Rubber Ducky Attack |
| Ataques contra servidores Web | Ataque de SQL Injection |
| Accesos a contenidos restringidos | Accesos a la red TOR |
| Ataques de intrusión física | Análisis y detección de anomalías en red |
| Utilidades CSIRT | Rescue-Terminal |
| Utilidades CSIRT | Cuarentena completa de Hosts comprometidos |

*Tabla 3.2.1 Listado de correspondencias entre escenarios y casos de uso*

Una de las características más útiles de algunas de las soluciones comerciales de EDR son las denominadas “Live Terminal”. Esta funcionalidad consiste en poder acceder a una terminal con privilegios elevados en cualquiera de los dispositivos donde está instalada la solución. Esto permite al CSIRT responder de forma rápida ante las amenazas, además de poder ejecutar acciones de forense de forma remota antes de que la evidencia se pierda. Gracias a todas las ventajas que ofrece, se ha decidido crear un CDU que ofrezca una terminal similar (denominada Rescue-Terminal).

Otro CDU utilizado como utilidad del CSIRT es la posibilidad de contener una amenaza que ya ha ocurrido, mediante una cuarentena completa del equipo que ha sido comprometido, para evitar el denominado desplazamiento lateral.

# Capítulo 4

# Diseño del sistema

Es pertinente elegir algunas de las herramientas concretas a utilizar, porque estas condicionan el diseño de la solución. Teniendo en cuenta las tablas de requisitos previas y los escenarios elegidos se han realizado las siguientes elecciones:

- SIEM: Graylog
- SOAR: Shuffle
- EDR:Wazuh

El detalle completo de las elecciones previas se encuentra disponible en el Anexo C.
Los conceptos de las herramientas seleccionadas se encuentran en el Anexo D.

Antes de plantear ningún diseño se han definido las políticas de seguridad que estos deben cumplir. Con las políticas de seguridad ya definidas, se ha diseñado una arquitectura global, y a su vez también se han definido las arquitecturas internas de sus elementos subyacentes, así como, los flujos de información del sistema en conjunto. Ya con el sistema completo, se han diseñado de forma abstracta los casos de uso que posteriormente serán implementados con la arquitectura propuesta. Así, se ofrece hacia el final de este apartado un marco teórico de arquitectura y servicio que sirve como base para la posterior implementación.

## 4.1 Políticas de seguridad

El sistema debe ser capaz de proteger de distintos vectores de ataque. El denominador común de estos es que, para poder lanzar el correspondiente mecanismo de defensa, primero es necesario que el sistema se alimente de los logs necesarios para detectar los ataques.

Teniendo en cuenta los aspectos mencionados anteriormente y estos últimos se extrae la siguiente lista de políticas de seguridad:

- El sistema debe ser capaz de agregar información de todos los elementos a defender sin incurrir en fugas de información.
- El sistema debe tener un diseño basado en bastionado, es decir, contar con segmentación de red y un control firme del tráfico autorizado tanto de entrada como de salida dada su gran importancia.
- La información interna del sistema debe estar tan compartimentada como sea posible entre los distintos componentes.
- En la fase de implementación se debe tener en cuenta el principio de mínimo privilegio.
- El sistema debe ser tolerante a situaciones de fallo o de alta carga.
- El sistema debe tener capacidad de respuesta a todos los niveles posibles, es decir desde respuestas individuales en los distintos hosts de la red, hasta control del tráfico general de la red o del sistema centralizado de gestión de cuentas del mismo.
- Se debe poder brindar una respuesta tan rápida como sea posible en función de la amenaza.

Estas políticas se van a ver reflejadas a lo largo de toda la etapa de diseño desde la arquitectura hasta el flujo de información del sistema.

## 4.2 Arquitectura global del sistema

El sistema está conformado por varios componentes interconectados entre sí, donde cada uno de ellos tiene su propia arquitectura interna. En este apartado se detalla la arquitectura general de la red y los elementos significativos con los que interactúa la solución.

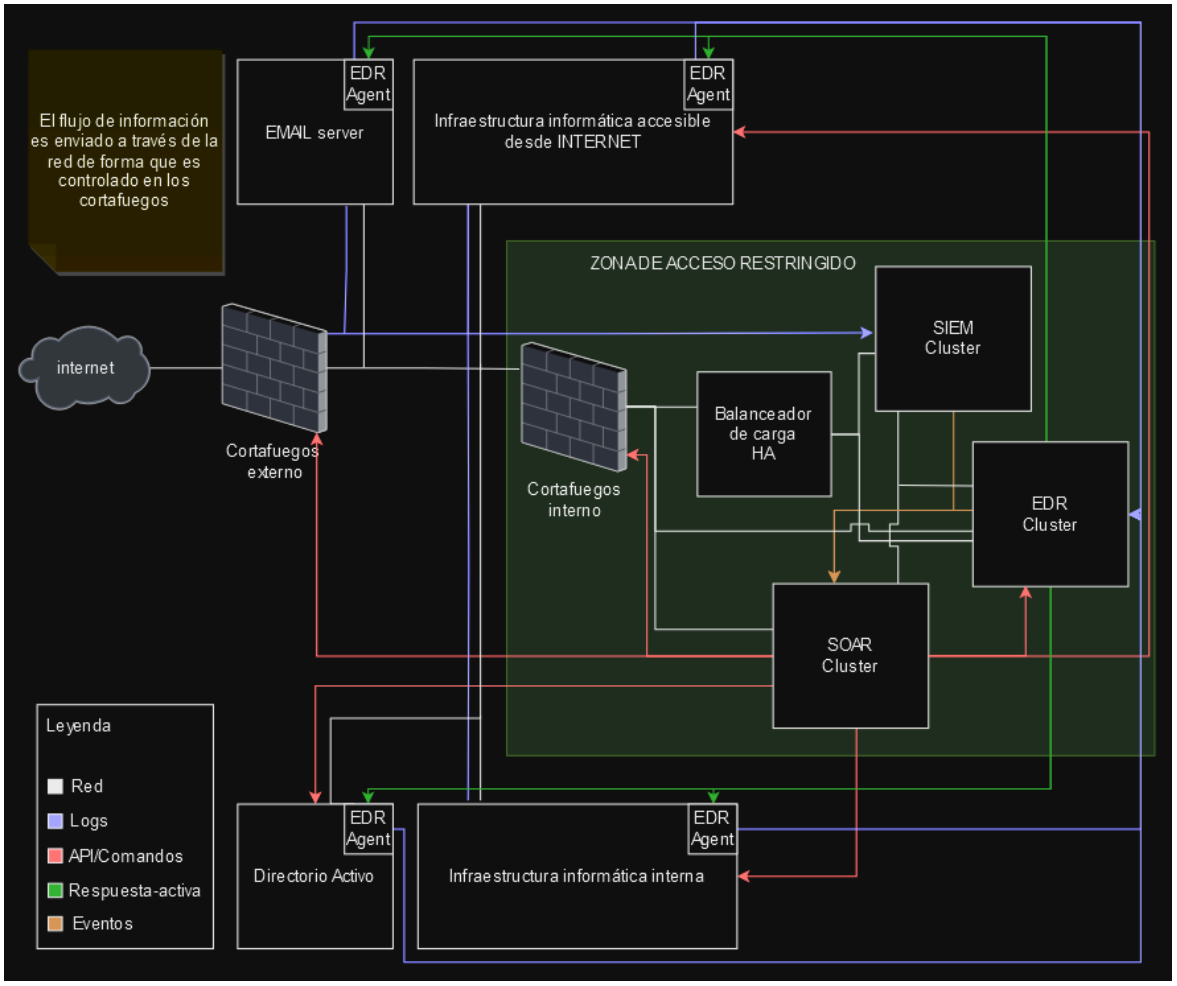


*Figura 4.2.1 Arquitectura del sistema completo*

Lo primero que debemos destacar de la arquitectura del sistema propuesto es que dado que el SOC es una pieza fundamental para garantizar la seguridad de toda la red, este no puede ser comprometido de ninguna manera. Es por ello que se encuentra en una zona de acceso restringido y extremadamente controlado mediante un cortafuegos adicional dentro de la propia red.

Los elementos más interesantes con los que el sistema va a interactuar de forma activa en las respuestas a incidentes son el cortafuegos externo y el Directorio Activo (sistema de gestión centralizada de usuarios), dado que con ellos se van a efectuar las acciones más relevantes. Otro elemento muy importante para efectuar las acciones van a ser los agentes del EDR que permite ejecutar acciones de respuesta rápida.

## 4.3 Arquitectura interna de las soluciones

Dado que los despliegues en HA/DR no han sido algo trivial, aquí se adjunta el diseño de los mismos así como una muy breve explicación de cada uno, no obstante el detalle completo de estos se encuentra en el Anexo E. Tanto en el SIEM como en el EDR ya existía una documentación indicando la tipología de nodos que debería tener un despliegue distribuido, no obstante en el SOAR no existía dicha documentación. A pesar de que dos de las herramientas contaban con información relativa al despliegue, ambas dejaban completamente de lado el despliegue de una arquitectura con balanceadores de carga a prueba de fallos, la cual es necesaria en ambos casos.

El principal reto ha sido automatizar el despliegue de las siguientes infraestructuras de forma que estas puedan escalar sin necesidad de modificar los distintos elementos encargados de realizar el despliegue, necesitando simplemente modificar únicamente una línea de código para reescalar la solución completa.

### 4.3.1 SIEM

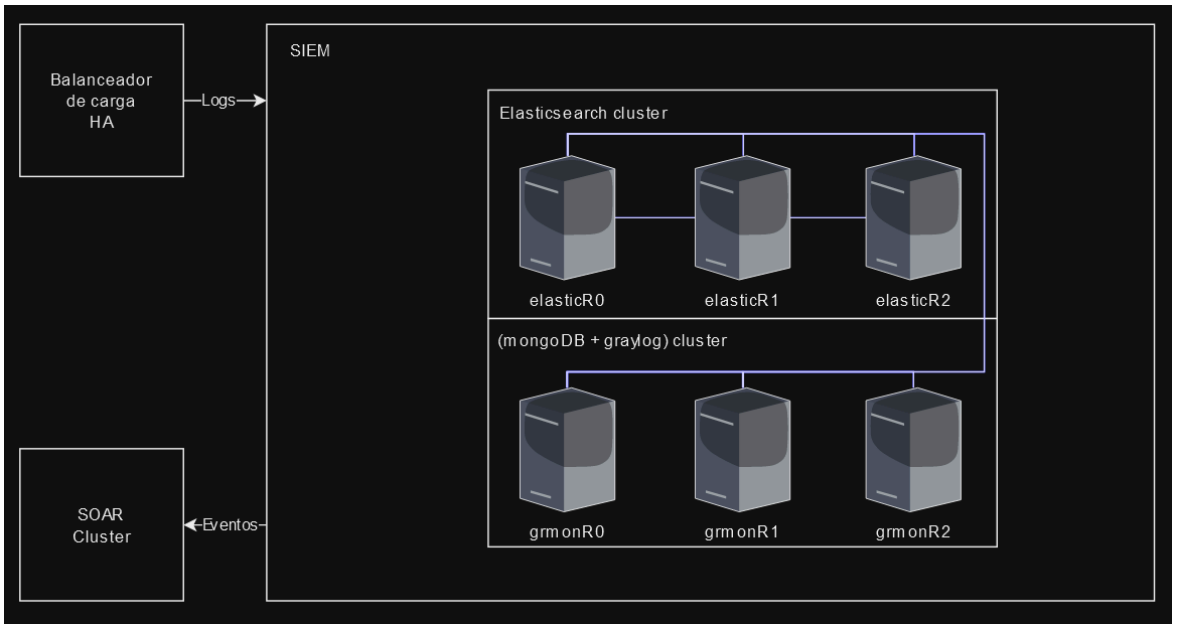


*Figura 4.3.1.1 Arquitectura interna del SIEM*

El SIEM se ha diseñado de forma que contiene una base de datos distribuida, concretamente ElasticSearch. También contiene un cluster de nodos que ejecutan la base de datos que contiene las configuraciones de la herramienta principal, concretamente MongoDB (en modo replicaset), y la herramienta principal Graylog.

El SIEM recibe logs desde el balanceador de carga HA y envía notificaciones de eventos al SOAR.

### 4.3.2 SOAR

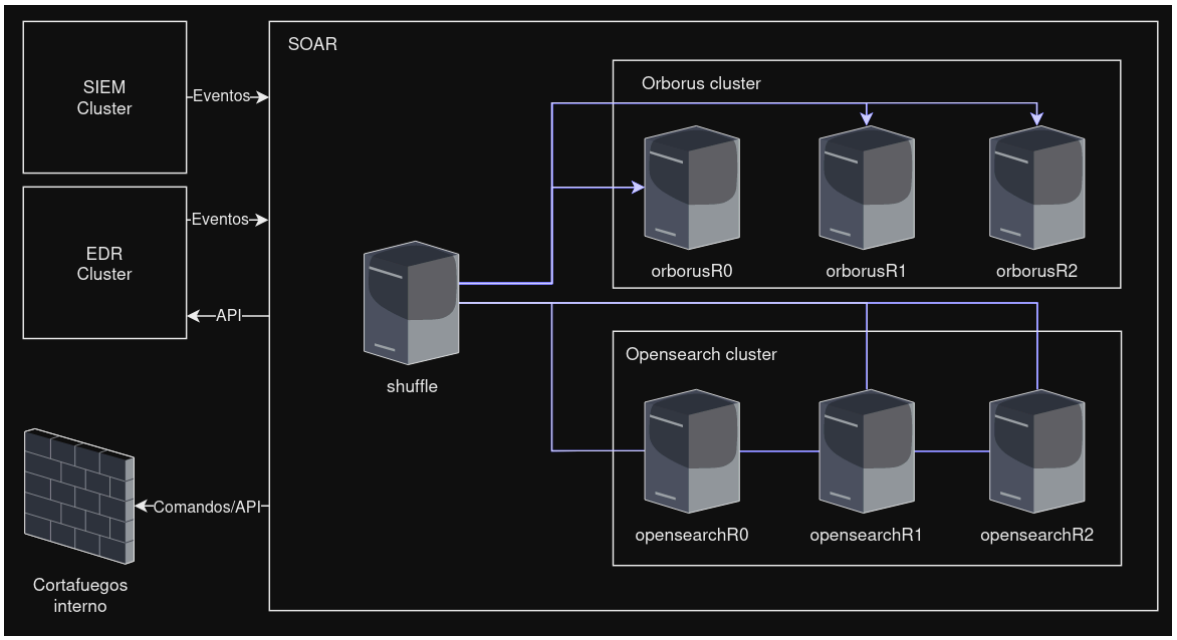


*Figura 4.3.2.1 Arquitectura interna del SOAR*

De forma similar a la arquitectura anterior, el diseño planeado cuenta con una base de datos clusterizada (Opensearch). Asimismo esta cuenta con un cluster de nodos Orborus, los cuales actúan como workers del SOAR. Por último el nodo Shuffle contiene la Web y el Backend. En este diseño por limitaciones de la herramienta solo se alcanza la propiedad DR y no la de HA, otra limitación de la herramienta es que esta debe ser desplegada mediante contenedores.

El SOAR recibe notificaciones de eventos del SIEM y del EDR. También envía llamadas a API al EDR y llamadas API o comandos al cortafuegos interno (que los puede reenviar a otros puntos).

Cabe destacar que a diferencia de las otras herramientas, el proyecto Open Source del SOAR no contaba con documentación relativa a cómo realizar un despliegue distribuido (a pesar de que la herramienta sí estaba preparada para este). Es por ello que tras comprobar exhaustivamente el correcto funcionamiento de este despliegue, se contribuyó al proyecto Open Source añadiendo dicho diseño. Para ello se realizó un fork del repositorio oficial de Github, se creó una nueva rama en dicho fork y posteriormente se abrió una Pull Request para incluir los nuevos contenidos en el repositorio oficial de la documentación de la herramienta.

### 4.3.3 EDR

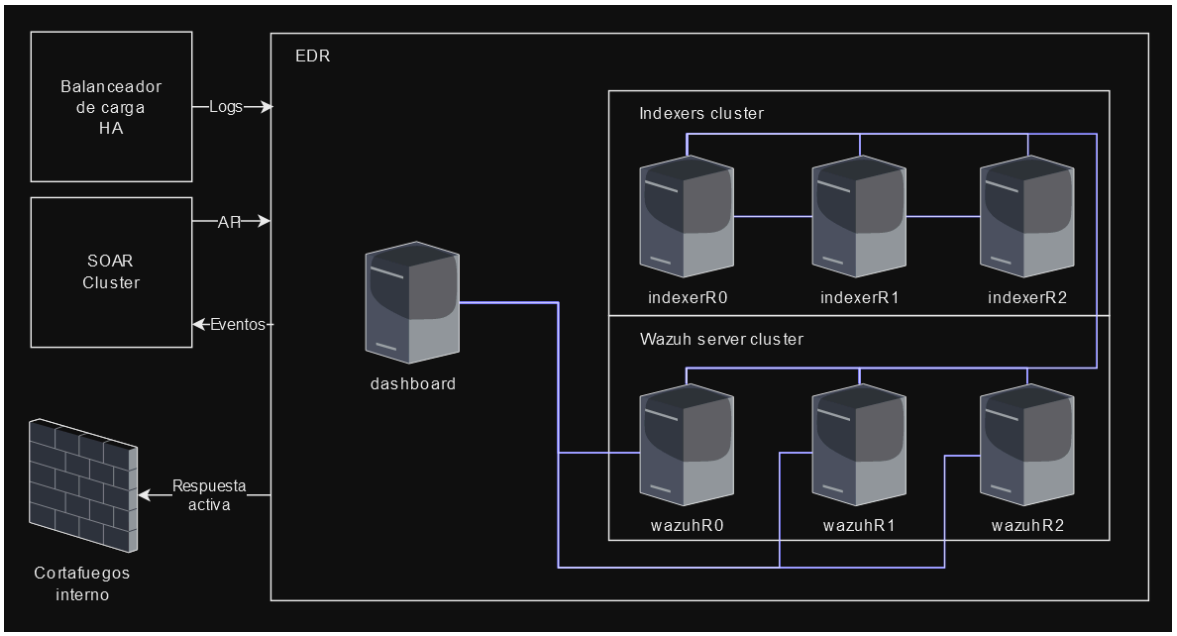


*Figura 4.3.3.1 Arquitectura interna del EDR*

Al igual que en las arquitecturas previas, la solución propuesta cuenta con una base de datos distribuida (un Fork modificado de Opensearch). Otro elemento significativo es el cluster de servidores que ejecutan toda la lógica y backend. Por último existe un único nodo encargado del Frontend Web. A pesar de la existencia del punto de fallo único en el Frontend, dado que la herramienta no necesita este para seguir en pleno funcionamiento, sí se ha conseguido un despliegue en HA (a excepción de dicho nodo que es DR). No se incluye en el diagrama pero también utiliza los balanceadores en HA.

El EDR recibe los logs desde el balanceador de carga HA, también puede recibir llamadas a API desde el SOAR. Por otro lado, este puede notificar eventos al SOAR y enviar respuestas activas a sus endpoints a través del cortafuegos interno.

### 4.3.4 Balanceador de carga en HA

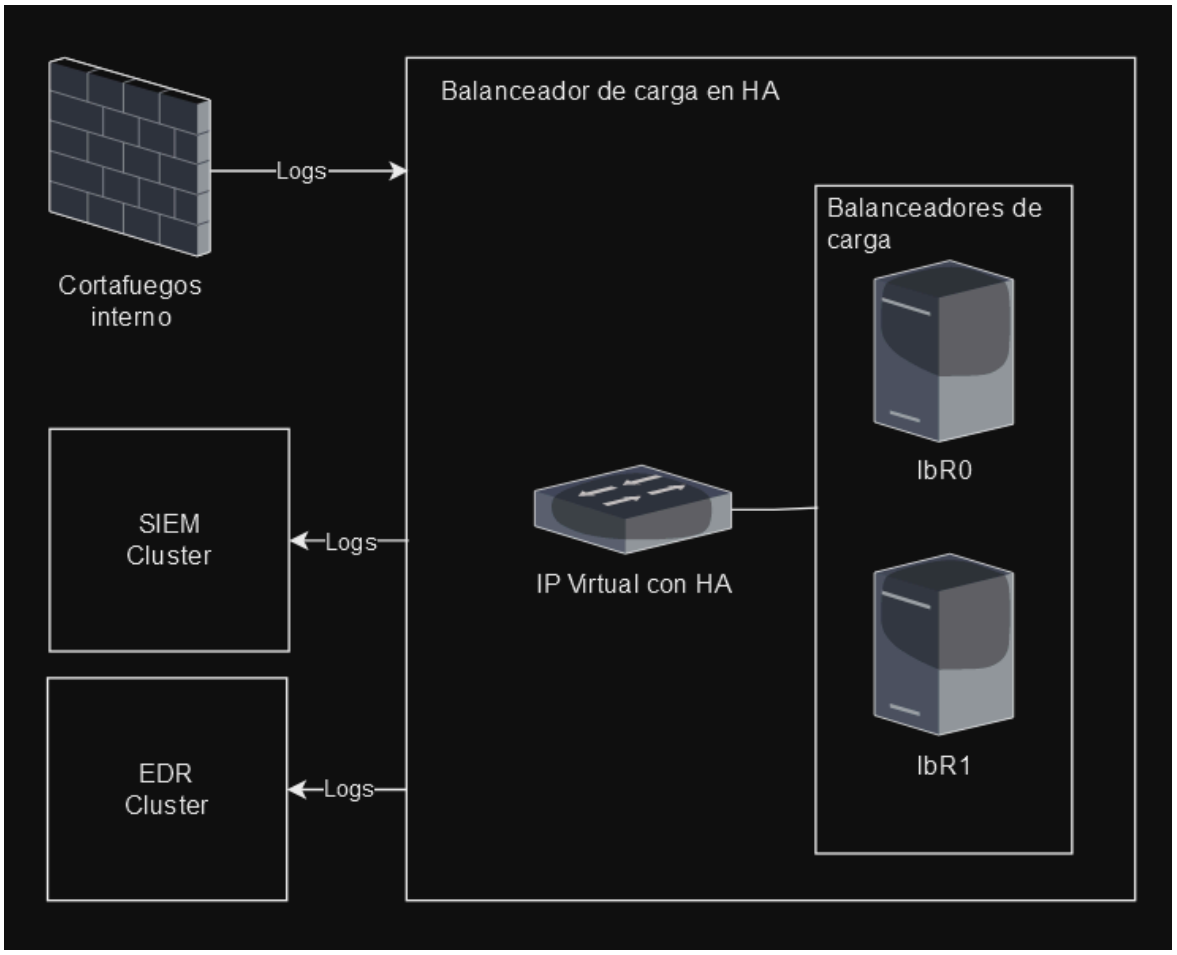


*Figura 4.3.4.1 Arquitectura interna de los balanceadores de carga en HA*

Dado que se agrega mucha información de entrada tanto para el SIEM como para el EDR y esta necesita ser distribuida de forma equivalente entre los servidores que conforman cada uno de dichos clusters, surge la necesidad de crear un balanceador de carga, no obstante dado el contexto y el requerimiento de la HA, no es suficiente con un balanceador.

Para crear una estructura de balanceador de carga en HA, primero se definen una serie de réplicas de balanceadores de carga que comparten una misma configuración. Para tener siempre una IP disponible aunque alguno de los balanceadores de carga falle se emplea una IP virtual, la cual en caso de que un balanceador caiga, se autoasigna a otra de las réplicas de los balanceadores de forma que se mantiene el servicio.

El balanceador HA recibe los logs provenientes del cortafuegos interno y los envía al EDR o al SIEM según corresponda.

## 4.4 Funcionamiento del sistema

Para poder realizar un esbozo a alto nivel de los casos de uso contemplados por el sistema, primero es necesario definir el funcionamiento básico del mismo y cuáles van a ser los flujos de información principales dentro de este.

Además dado que la solución cuenta con componentes claramente diferenciados, estos deben ser capaces de interaccionar de forma correcta entre sí, creando de esta forma una secuencia de pasos ordenados que cubra todo el ciclo defensivo.

### 4.4.1 Generación de alertas en el SIEM

El SIEM va a ser el punto neurálgico a la hora de manejar información del estado global del sistema. Va a ser el encargado de agregar la información de los distintos elementos (recibida en forma de logs) para lanzar las alertas correspondientes en caso de ataque.

Los logs entran a Graylog a través de los denominados *Inputs*, nada más entrar estos son *parseados* por los denominados *Extractores*, posteriormente son almacenados en los *Stream*, estos últimos pueden aplicar filtros sobre los logs para determinar si pueden ser almacenados en un *Stream* concreto o no.

Tras ser almacenados por primera vez, los logs pueden ser enriquecidos mediante el uso de *Pipelines*. Estas se conforman de *Stages*, y cada uno de ellos tiene una o más reglas que deben cumplirse para avanzar al siguiente *Stage* o *Pipeline*. Estas reglas permiten manipular los distintos logs o almacenarlos en otros *Stream*.

Para enriquecer logs con información externa se emplean los *LookUp Tables*. Estos se componen de una *Memoria Caché* y de un *Data Adapter*, donde el último de estos es el encargado de extraer la información ya sea mediante un CSV o una llamada a API.

Para generar eventos y alertas se ha empleado el plugin de correlaciones desarrollado por Airbus-Cyber. Este plugin proporciona una forma muy sencilla para, en base a una serie de *reglas*, agrupar ciertos tipos de logs y en caso de que se den una serie de condiciones generar la *alerta* correspondiente. El uso del plugin agiliza la creación de nuevos eventos y alertas, lo que implica una mejora significativa en el tiempo de respuesta ante incidentes. Al crear una *Alerta* en el plugin este crea un *Stream* que contiene las *reglas* necesarias para filtrar solo los logs que pueden disparar el *evento*. Además, crea el *evento*, el cual realiza de forma periódica la búsqueda en este *Stream* generando la correspondiente *notificación*.

Las alertas generadas son notificadas a través de una comunicación HTTPs a un Webhook configurado en el SOAR.

### 4.4.2 Generar respuestas automáticas en el SOAR

Los *workflows* son disparados por los *triggers*. Estos tres son los *triggers* más interesantes:

- Mediante un *webhook*, donde el SIEM o el EDR se encargan de lanzar una petición HTTPs POST que contiene la información del evento.
- El segundo de ellos es mediante una ejecución programada de forma temporal. Esto permite realizar comprobaciones periódicas con las que realimentar al SIEM.
- La tercera opción nos permite abstraer como si de funciones se tratase, mediante el uso de llamadas a *Subworkflows*.

Cada CDU cuenta con su *workflow* particular según el procedimiento de actuación a seguir.

Aquellas partes repetidas en varios *workflows* se deben encapsular en *subworkflows*.

Una vez activado el *trigger* de un *workflow*, se ejecutan las acciones definidas en este. Dentro de estas ejecuciones podemos encontrar variables de *workflow*, es decir, constantes predefinidas antes de que este fuese ejecutado, y *execution arguments,* o variables que son generadas en runtime.

Es posible realizar ejecuciones condicionales o bucles. Siempre se puede acceder a los resultados de las ejecución de tareas previas. Estos datos son accesibles en formato JSON y se pueden modificar mediante el uso de Python embebido o con sintaxis de Liquidity.

### 4.4.3 Funcionamiento del EDR

El EDR se encarga de monitorizar de forma proactiva todos los Endpoint detectando cambios en los ficheros relevantes del sistema, así como virus conocidos o patrones de comportamiento. Un agente instalado en un Endpoint también se encarga de todo lo relacionado con Configuration Management (CM), tal que un administrador puede ir realizando los parches de seguridad necesarios sobre el sistema.

Ofrece capacidades como herramienta de investigación forense para analizar de forma detallada una traza temporal de lo ocurrido, así como la posibilidad de responder de forma completamente automática cuando se activan determinadas alertas, como por ejemplo la detección de un ataque de SQL injection. Se pueden crear respuestas automáticas creando los scripts correspondientes en Python.

El EDR cuenta con una API la cual permite interactuar con este, de forma que otros componentes de la solución puedan lanzar acciones del mismo en uno o varios endpoints. Si a lo anterior le añadimos la posibilidad de emplear respuestas activas propias programadas en Python, las posibilidades son infinitas.

## 4.5 Casos de uso

Con toda la arquitectura y los flujos de información ya definidos, es necesario también diseñar el funcionamiento de los CDUs, concretados previamente en la etapa de análisis, los diseños de estos vienen condicionados por los tipos de herramientas seleccionadas para la solución.

A continuación se adjunta el diseño del funcionamiento, de cada uno de ellos, donde estos se explican a un nivel independiente de las herramientas concretas elegidas.

Los diseños de funcionamiento a alto nivel han sido ordenados y agrupados según el aspecto de seguridad que cubren. Cada vez que aparece una nueva mecánica operativa, dentro de estos diseños, se explica el funcionamiento de la misma.

### 4.5.1 Disponibilidad

**Ataque DDoS**

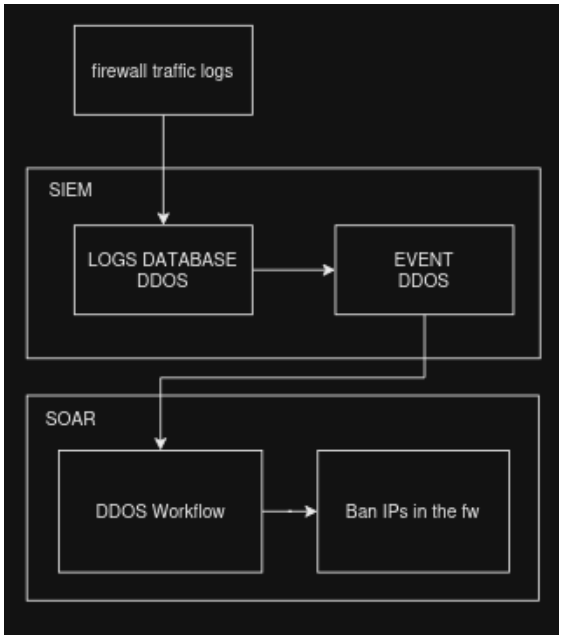


*Figura 4.5.1.1 Diseño alto nivel CDU ataque DDoS.*

Para detectar un ataque DDoS se deben emplear los logs del cortafuegos, ya que de estos se puede extraer la información de todo el tráfico entrante de la red, más concretamente el tamaño en Bytes de las comunicaciones (también se puede emplear la cantidad de peticiones). El SIEM almacena dichos logs en la BBDD correspondiente, correla estos y en caso de detectar un posible ataque lanza el correspondiente Workflow en el SOAR. El workflow a su vez bloquea las IPs correspondientes en el cortafuegos.

### 4.5.2 Autenticación

**Fuerza bruta a SSH**

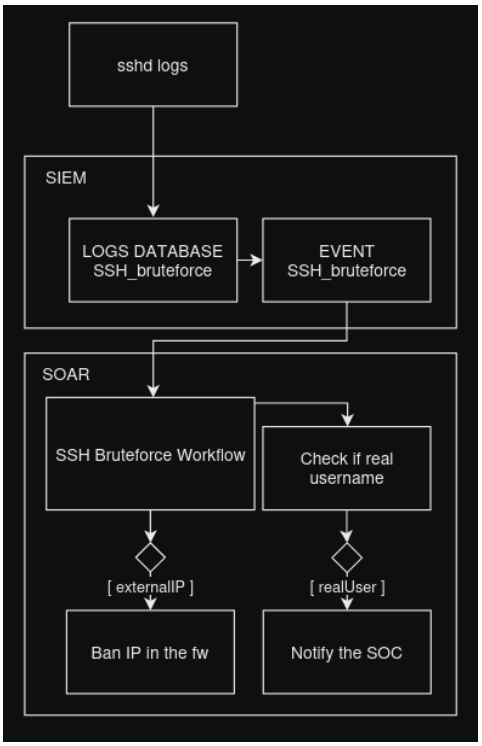


*Figura 4.5.2.1 Diseño alto nivel CDU ataque fuerza bruta a SSH.*

La detección de un ataque de fuerza bruta comienza con el envío y procesamiento en el SIEM de los logs de acceso SSH de un servidor. Este diseño incluye la ejecución condicional, dentro del Workflow del SOAR.

**Ataque de SQL Injection**

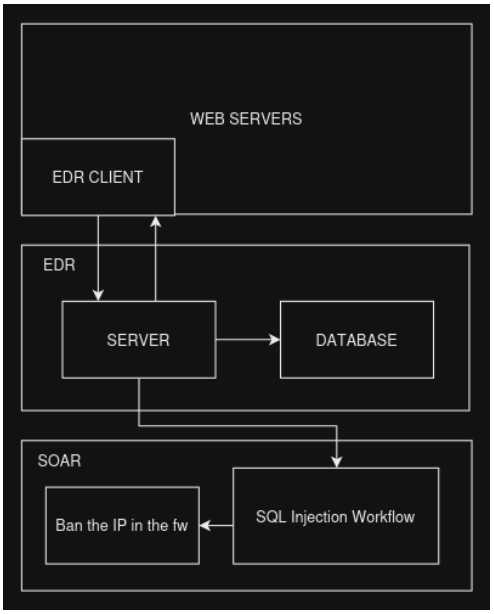


*Figura 4.5.2.2 Diseño alto nivel CDU ataque SQLi*

Para proteger a los servicios de este tipo de ataque, se van a emplear los logs del servidor web que registran las peticiones contra la base de datos. Dado que estos no necesitan ni ser correlados, ni ser enriquecidos, se puede emplear directamente el EDR para detectar el ataque sin necesidad de pasar por el SIEM, posteriormente se actúa empleando para ello el SOAR.

**Detección de phishing**

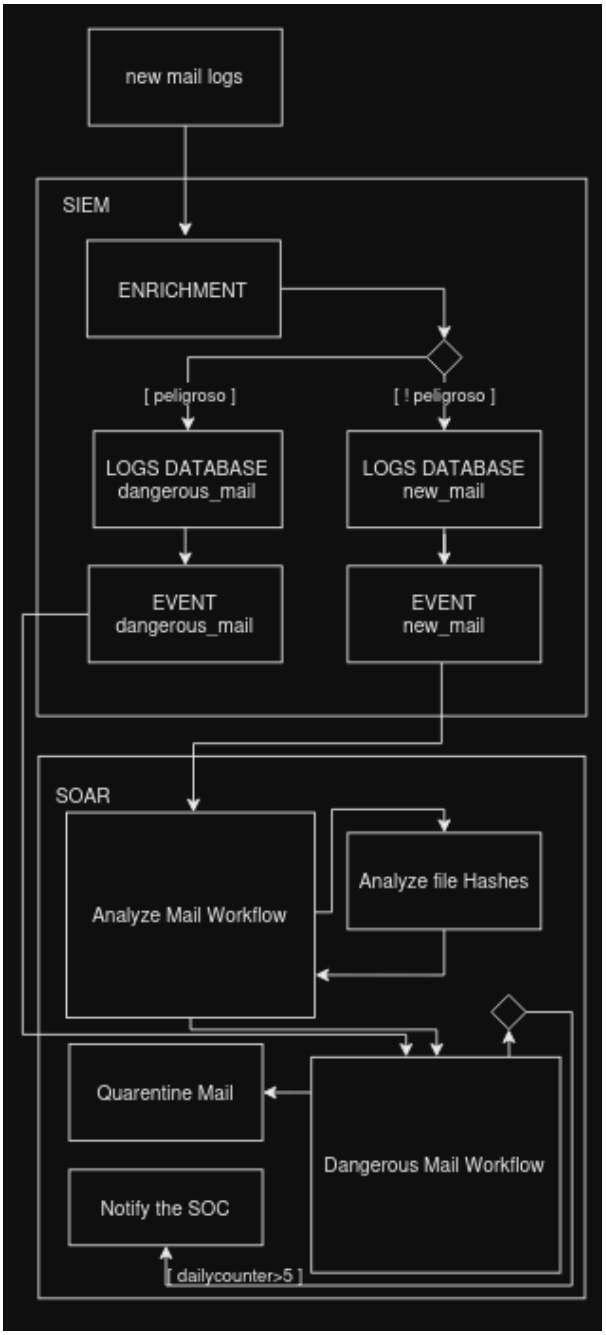


*Figura 4.5.2.3 Diseño alto nivel CDU Detección de phishing.*

Concretamente este CDU trata el phishing realizado a través del correo electrónico, es por ello que el primer paso es detectar la recepción de este. Para ello, los logs del correspondiente servidor de correo electrónico son enviados al SIEM y procesados. Además de las etapas comunes de procesamiento, estos logs tienen una fase adicional de enriquecimiento. Posteriormente estos son almacenados en un lugar específico de la BBDD o en otro dependiendo del contenido que ha sido añadido mediante el enriquecimiento. También destaca que en este caso dentro del SOAR, un workflow lanza otro workflow.

### 4.5.3 Autorización

**Accesos a webs con URLs maliciosas conocidas**

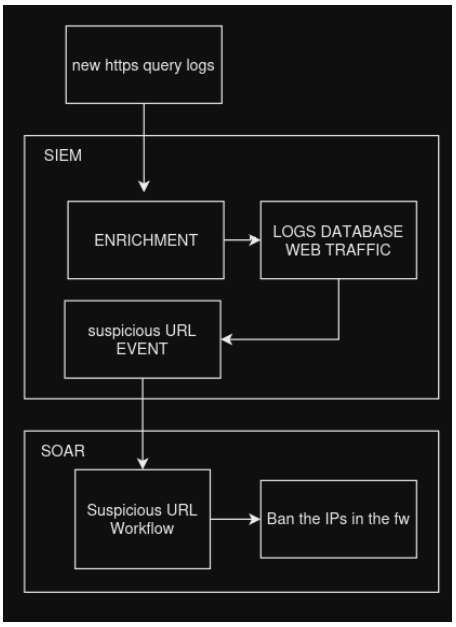


*Figura 4.5.3.1 Diseño alto nivel CDU acceso a webs maliciosas*

El primer paso para la detección de un acceso a una web maliciosa es conseguir extraer las IPs destino de los accesos a páginas web. Para ello se deben emplear los logs de tráfico del cortafuegos al ser este el elemento central de las comunicaciones.

**Accesos a la red TOR**

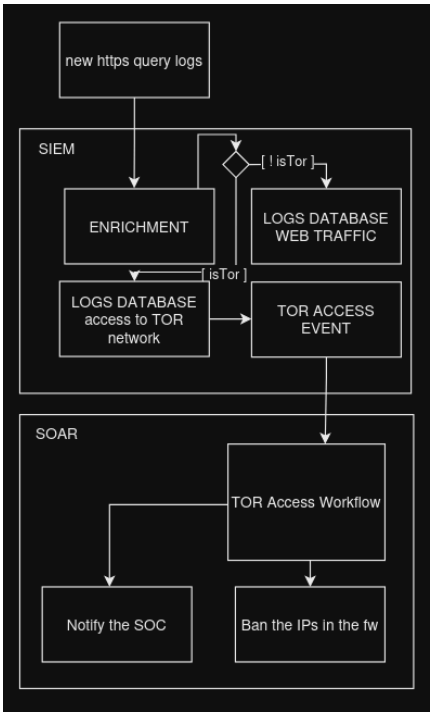


*Figura 4.5.3.2 Diseño alto nivel CDU accesos a la red TOR.*

La red TOR (*The Onion Router*) es una red que encapsula el tráfico mediante capas de cifrado para tratar de ocultar información, como por ejemplo los datos enviados o la IP tanto de destino como de origen de quien la utiliza. Además, visto desde fuera, parece tráfico HTTPs común, por ello para poder detectarlo se deben emplear logs del cortafuegos.

**Análisis y detección de anomalías en red**

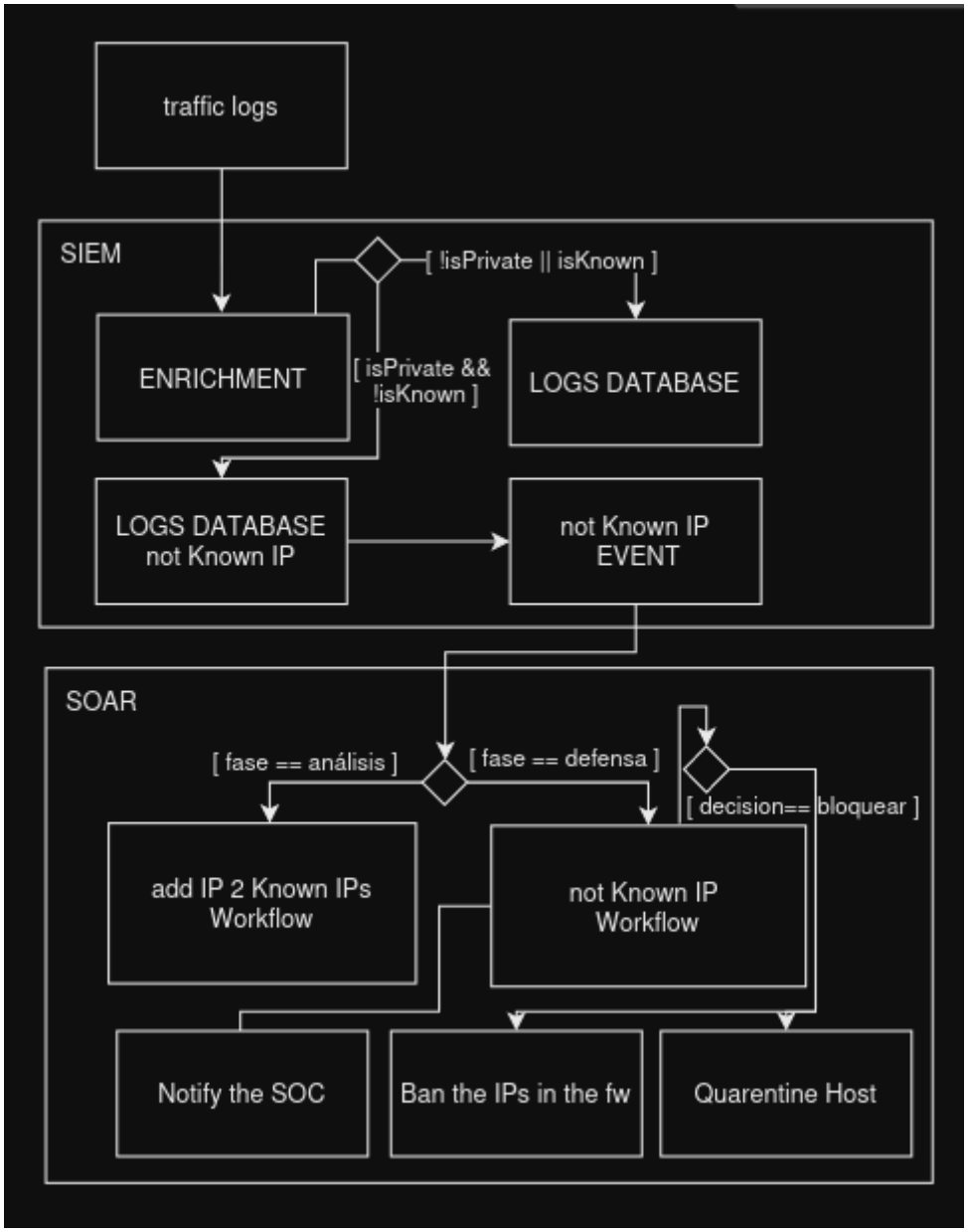


*Figura 4.5.3.3 Diseño alto nivel CDU análisis y detección de anomalías en red*

Este CDU pretende cubrir un caso de intrusión física en el que el atacante se conecta sin autorización a la red. El CDU consiste en dos fases, una etapa inicial de aprendizaje de la red y una posterior de defensa, en la cual se puede decidir si bloquear o no las IPs sospechosas..

### 4.5.4 Confidencialidad

**Data Leaks mediante USB y Ataque mediante Rubber Ducky**

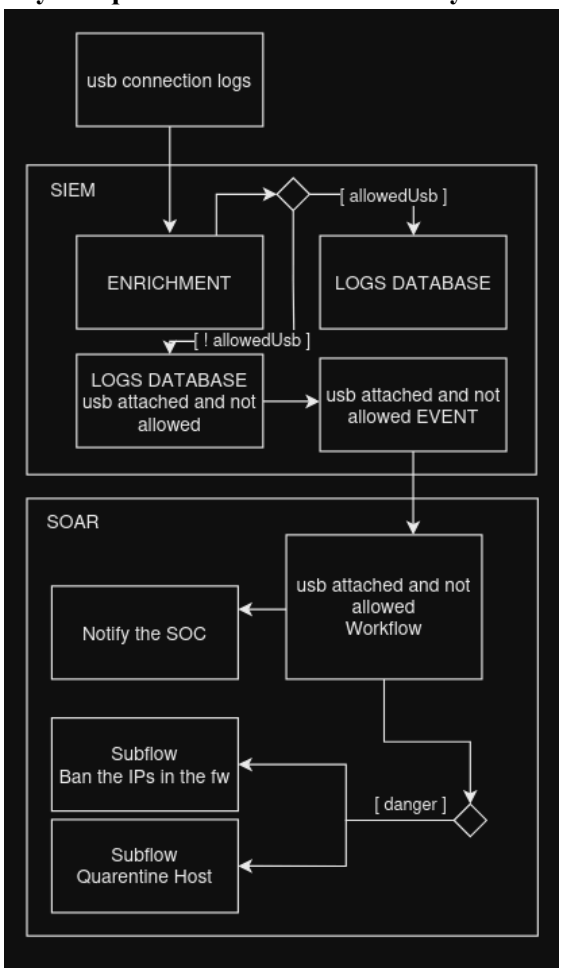


*Figura 4.5.4.1 Diseño alto nivel CDU Data Leaks mediante USB y ataque mediante Rubber Ducky*

Para ambas casuísticas de ataque son necesarios los logs de conexión de los USBs a los distintos dispositivos, los cuales deben ser enriquecidos en el SIEM.

### 4.5.6 Utilidades para CSIRT

Dado que ambos CDUs son lanzados de forma manual, estos comienzan en el SOAR y posteriormente aprovechan las capacidades del EDR para actuar sobre los endpoints como corresponde.

**Rescue-Terminal**

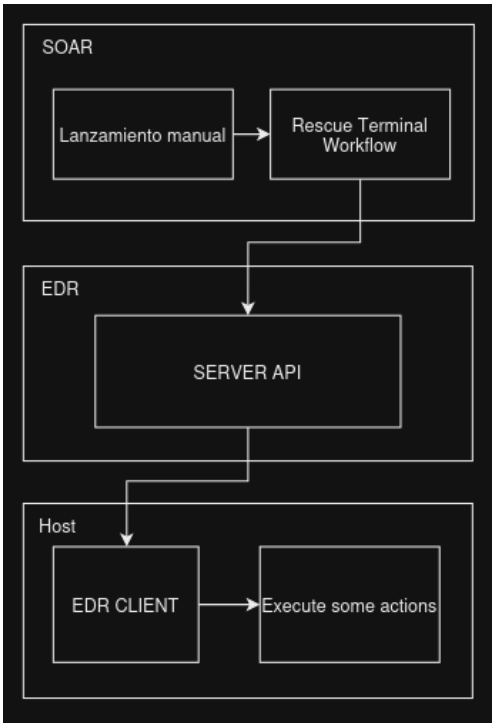


*Figura 4.5.6.1 Diseño alto nivel CDU Rescue-Terminal*

**Cuarentena completa de Hosts comprometidos**

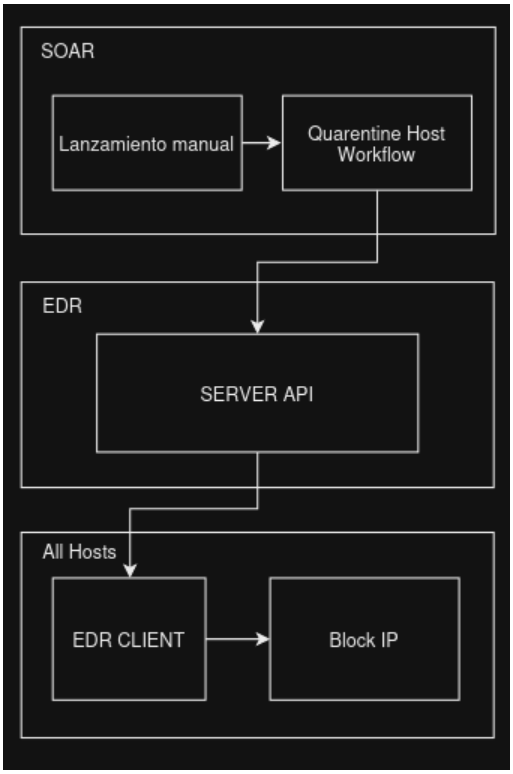


*Figura 4.5.6.2 Diseño alto nivel CDU Cuarentena completa de Hosts comprometidos*

## 4.6 Despliegue

En la figura 4.6.1 se muestra el diseño del diagrama de despliegue de la zona de acceso restringido de la solución, mostrada en el diagrama de arquitectura global previo:

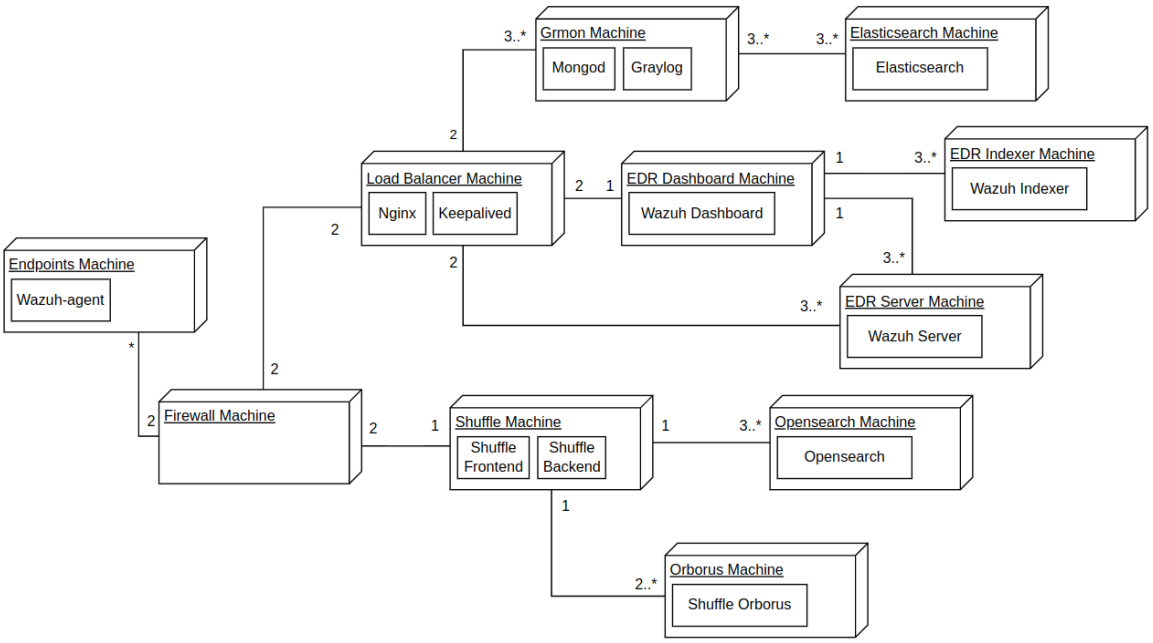


*Figura 4.6.1 Diagrama de despliegue de la solución completa*

El despliegue base, el cual ha sido diseñado para poder ser escalado, cuenta con un total de 23 máquinas, este garantiza el HA de todos los componentes (exceptuando el SOAR que solo alcanza la DR).

Todo el tráfico entrante y saliente debe pasar por el cortafuegos, de forma que sea imposible realizar comunicaciones no autorizadas con la zona de red de la solución.

Para ahorrar recursos se reutiliza la misma estructura de balanceadores de carga en HA, para el SIEM y el EDR.

Este despliegue concreto, arrastra una serie de implicaciones posteriormente a la hora de implementar la solución y de resolver los CDUs con la misma.

# Capítulo 5

# Implementación

La implementación del sistema cuenta con la puesta en marcha de cada uno de los elementos significativos del sistema y una breve explicación de cómo se emplean, adicionalmente se contemplan en un último apartado algunas de las configuraciones necesarias de dichos elementos para funcionar de forma correcta en los casos de uso planteados.

## 5.1 Despliegue de la infraestructura

### 5.1.1 Hardware empleado y SO

Inicialmente para los entornos de prueba tempranos se emplearon despliegues sobre máquinas virtuales (VM, de sus siglas en inglés) temporales. Una vez se alcanzó un despliegue completamente automático que contaba con las características deseadas, este se llevó a cabo en un entorno real de preproducción, el cual es completamente idéntico al de producción dado que la metodología de despliegue y de red siguen el mismo funcionamiento. Además, para garantizar su comportamiento real sin impactar en producción, los sistemas base de estas VMs estables fueron desplegados por el departamento de servicios internos de Nologin acorde a los procedimientos de producción que tienen definidos. Todas las VMs tienen como S.O. Ubuntu Focal dado que este es compatible con todas las herramientas empleadas y es un S.O. muy curado y estable. Como modelo de contenedores para el SOAR se ha optado por emplear Docker, dado que todavía no existe una documentación adecuada para una implementación sobre Kubernetes (K8s).

Cada VM cuenta con unas especificaciones de hardware diferentes dependiendo del tipo de tarea que va a realizar, estas se recogen en la siguiente tabla:

| Tipo de máquina | RAM | Cores de CPU | Disco |
|---|---|---|---|
| BBDD | 2-4 GB | 2 | 15 GB |
| Procesado | 3-6 GB | 2-4 | 10 GB |
| Balanceador de carga | 1 GB | 1 | 8 GB |

*Tabla 5.1.1.1 Especificaciones Hardware de cada tipo de máquina en la implementación*

De cualquier manera, se han seguido las especificaciones de los desarrolladores en cuanto a capacidades para garantizar despliegues exitosos en clientes. Dependiendo de los requisitos del cliente se debería aumentar el tamaño del almacenamiento en disco de las bases de datos.
El cortafuegos empleado ha sido el de Palo Alto Networks, dada la relación comercial existente entre ambas empresas.

### 5.1.2 Despliegue automático

En cuanto a la automatización del despliegue, teniendo en cuenta que este se puede reescalar simplemente añadiendo o eliminando una línea de código, en primera instancia se emplea un script programado en Ruby que se encarga de una primera etapa de aprovisionamiento a través de SSH, en la cual se realiza una copia de todos los ficheros necesarios a todas las VMs.

Este script además genera de forma automática unas variables auxiliares que son necesarias para automatizar este escalado. Posteriormente dicho script ejecuta distintos playbooks (ficheros que modelan las configuraciones con las que debe contar el sistema objetivo) de Ansible (herramienta de modelado de sistemas) inyectando en cada uno las variables mencionadas anteriormente. Cada tipo de nodo cuenta con un playbook asociado. Todos los playbooks han sido creados específicamente para este despliegue. Aproximadamente suman un total de 550 líneas, solo contando los propios playbooks.

Dichos playbooks realizan todas las configuraciones en las VMs poniendo en funcionamiento todos los elementos del sistema mediante una sola llamada a este script. Adicionalmente para facilitar las configuraciones de algunos ficheros concretos se ha recurrido al uso de templates hechos con la sintaxis de jinja2.

El resultado que subyace a todo lo anterior es realizar un despliegue automático y transparente en el cliente. De esta manera, se puede comenzar rápidamente a monitorizar la ciberseguridad de los clientes sin que tenga que verse afectado su negocio. Al contar con una configuración base, solamente serán necesarias pequeñas configuraciones tras el despliegue.

### 5.1.3 Software y configuraciones de cada tipo de nodo

En lo relativo a la puesta en marcha de los servidores se utiliza la nomenclatura RX para denominar aquellos elementos que se encuentran replicados, siendo la X el número de la réplica en cuestión.

- elasticRX: Se ha instalado la base de datos ElasticSearch y se ha manipulado el fichero /etc/elasticsearch/elasticsearch.yml.
- grmonRX: Se han instalado la base de datos MongoDB y el SIEM Graylog, también se han modificado los ficheros /etc/mongod.conf , /etc/graylog/server/server.conf y /etc/default/graylog-server. Adicionalmente se ha instalado el plugin de graylog-plugin-alert-wizard desarrollado por Airbus-Cyber.
- opensearchRX: Se ha instalado la base de datos OpenSearch y se ha modificado el fichero /etc/opensearch/opensearch.yml.
- orborusRX: Se ha clonado el repositorio de Shuffle, se ha modificado la variable de entorno BASE_URL y se ha lanzado el correspondiente Docker-Compose.
- shuffle: Se ha clonado el repositorio de Shuffle, se ha modificado la variable de entorno SHUFFLE_OPENSEARCH_URL y se ha puesto en ejecución el Docker-Compose.
- indexerRX, wazuhRX, dashboard: Se ha descargado y lanzado el instalador wazuh-install.sh. En el último nodo de tipo indexerRX se ha iniciado el cluster empleando el flag --start-cluster.
- lbRX: Se han instalado los servicios Nginx y Keepalived, se han modificado los ficheros /etc/keepalived/keepalived.conf y /etc/nginx/nginx.conf.

## 5.2 Desarrollo casos de uso

En este apartado se encuentran detallados los desarrollos de los CDUs definidos previamente, partiendo desde el razonamiento sobre diagramas de alto nivel específicos de las herramientas empleadas hasta la implementación en cada una de las herramientas de los distintos CDUs. Los elementos auxiliares empleados, como subworkflows, pipelines (y sus respectivos stages), configuraciones del EDR y lookup tables se encuentran al ser menos relevantes en el Anexo F.

### 5.2.1 Detección de phishing

Para la etapa de enriquecimiento se han empleado las APIs de OTX (implementado con Data Adapters, LookUp Tables y Pipelines disponibles en Anexo F.1 y F.4) para revisar en una primera instancia la legitimidad del origen del correo. Los logs son correlados y se genera un evento por cada mail recibido, que es notificado al SOAR mediante una llamada HTTPs, disparando el workflow de **Analyze Mail**.

En el caso de que inicialmente se haya detectado que este correo electrónico es potencialmente peligroso, desde este workflow se ejecuta directamente el workflow **Dangerous Mail**. Si no es así, el workflow inicial **Analyze Mail** obtiene una copia del contenido del e-mail y lo analiza en busca de 2 elementos: enlaces, los cuales son verificados mediante la API de VirusTotal, y los Hashes de ficheros, que también son verificados con esta API. En caso de detectar algo sospechoso se ejecuta el workflow **Dangerous Mail**.

Dentro del workflow **Dangerous Mail** se realiza una cuarentena preventiva del correo electrónico para proteger a la organización y al usuario. Adicionalmente, se aumenta un contador diario con el número de phishings recibidos por ese usuario y, si este contador supera un umbral predefinido (por ejemplo, 5 correos electrónicos no legítimos) se notifica al SOC dado que la dirección de correo puede haber sido comprometida.

La pipeline del SIEM cuenta con 2 etapas. La primera de ellas realiza el enriquecimiento, y la segunda en caso de que el correo venga de una fuente maliciosa lo elimina del stream de **new_mail** y lo añade al stream de **dangerous_mail**.

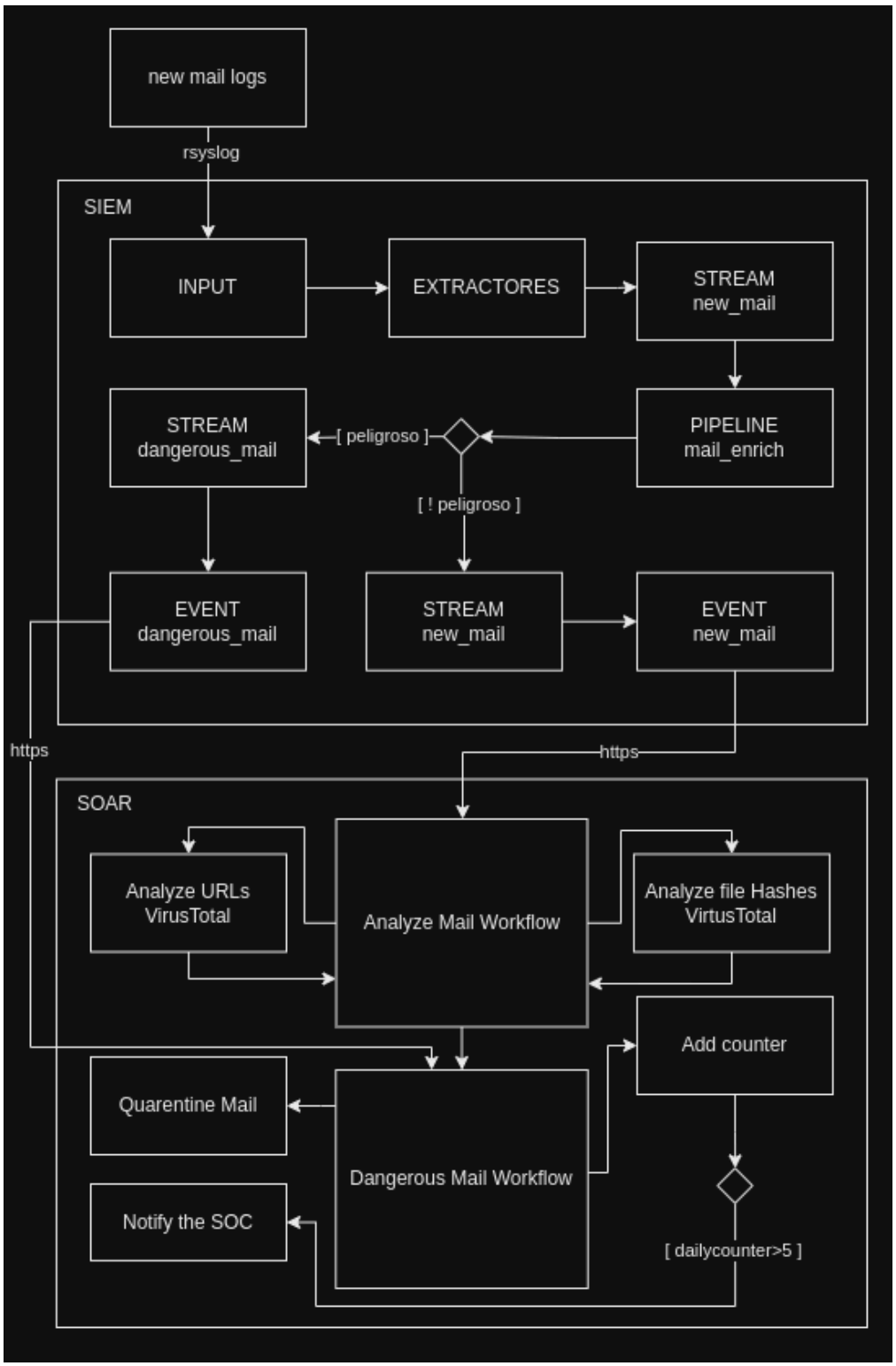


*Figura 5.2.1.1 Diagrama de implementación CDU Detección de phishing*

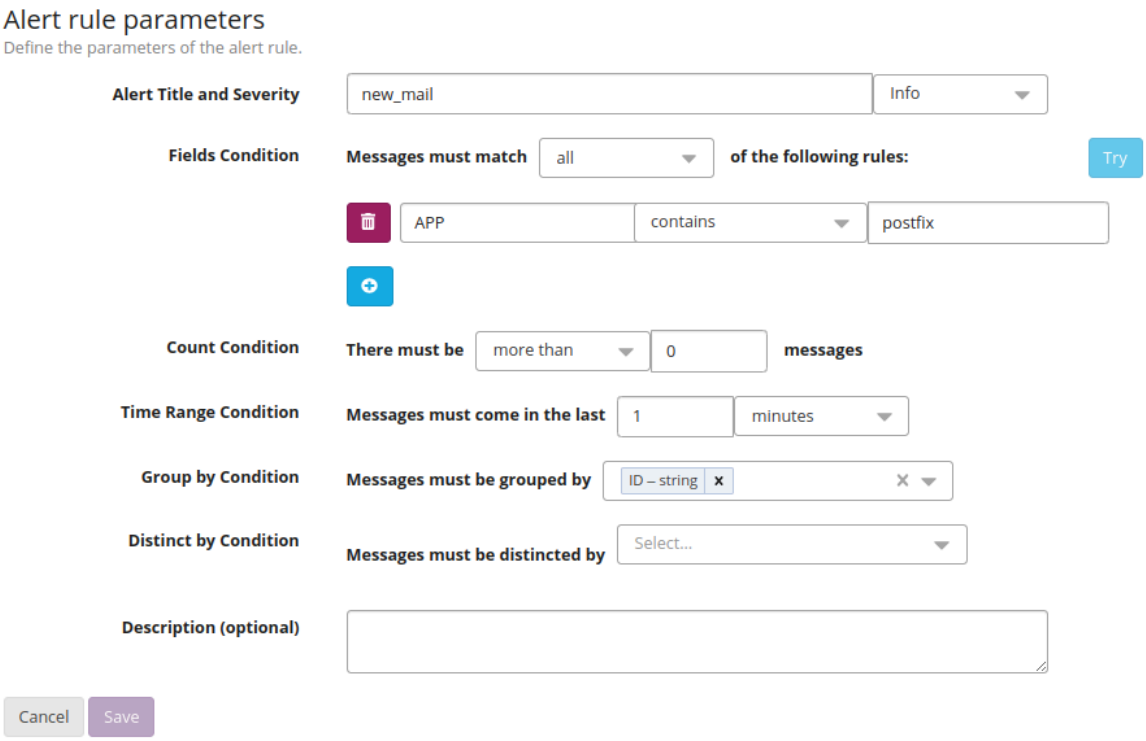


*Figura 5.2.1.2 Implementación alerta SIEM new_mail*

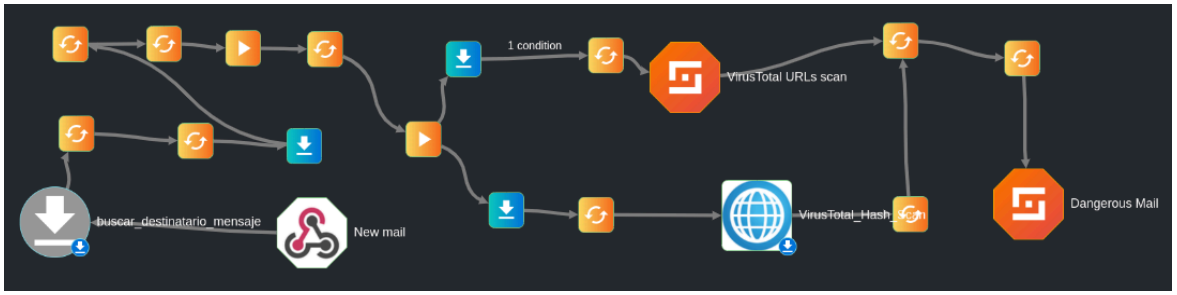


*Figura 5.2.1.3 Implementación workflow SOAR Analizar Mail*

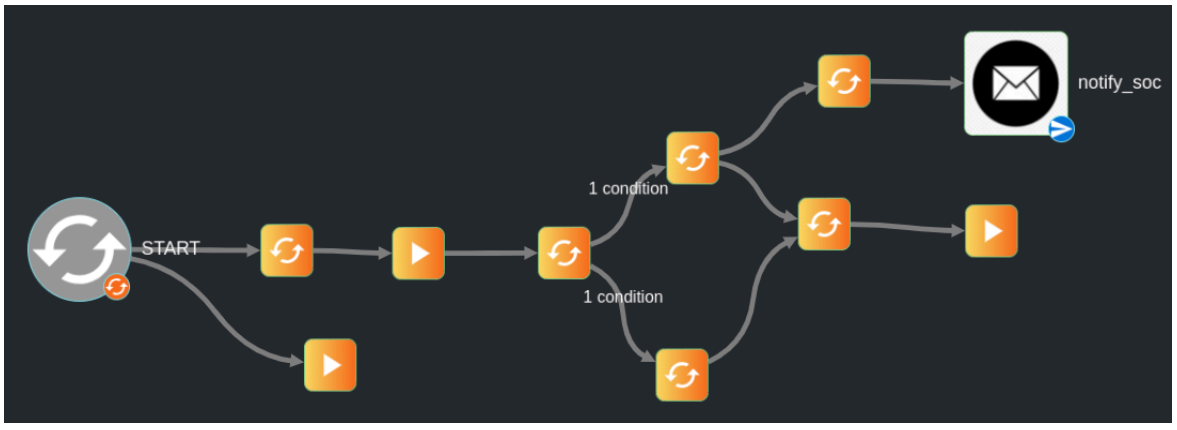


*Figura 5.2.1.4 Implementación workflow SOAR Dangerous Mail*

### 5.2.2 Fuerza bruta a SSH

Los logs son correlados, en el caso de que se detecten más de 10 intentos de login fallidos con un mismo usuario o desde una misma IP en menos de 1 minuto se genera un evento. Este evento es notificado por HTTPs al SOAR activando el **SSH Bruteforce** workflow.

El workflow del SOAR analiza la información y extrae la IP del atacante, así como las cuentas afectadas. Dentro del ataque cabe distinguir dos tipos. El primero de ellos es mediante el uso de un diccionario de cuentas comunes tipo root, admin, pi, etc. Este caso es el más común y no debe generar preocupación en el SOC. No obstante, un caso que sí que debe ser analizado en profundidad es el intento de ataque de una cuenta real de un empleado de la empresa, especialmente si esta es de alguien con muchos privilegios. En el primer caso se opta por bloquear la IP mediante una llamada API al cortafuegos empleando para ello el subworkflow asociado (disponible en el Anexo F.2). El segundo caso adicionalmente debe notificar al equipo de ciberseguridad para una investigación en mayor profundidad de forma manual. Con esta investigación se determinará por qué dicha cuenta es objeto del ataque, lo que puede proporcionar información muy útil sobre posibles fugas de información, *insiders*, etc.

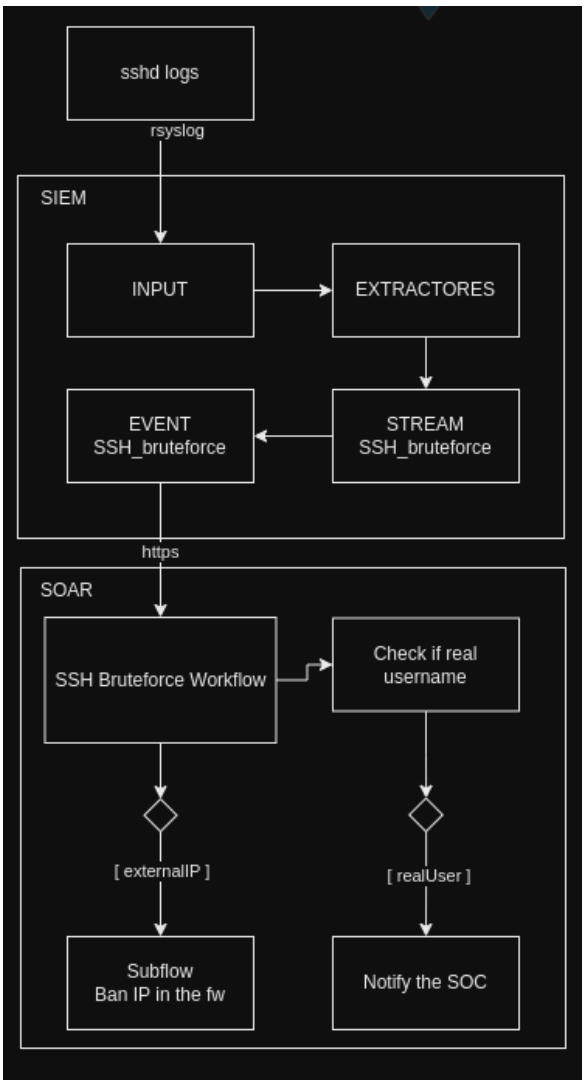


*Figura 5.2.2.1 Diagrama de implementación CDU Fuerza bruta a SSH*

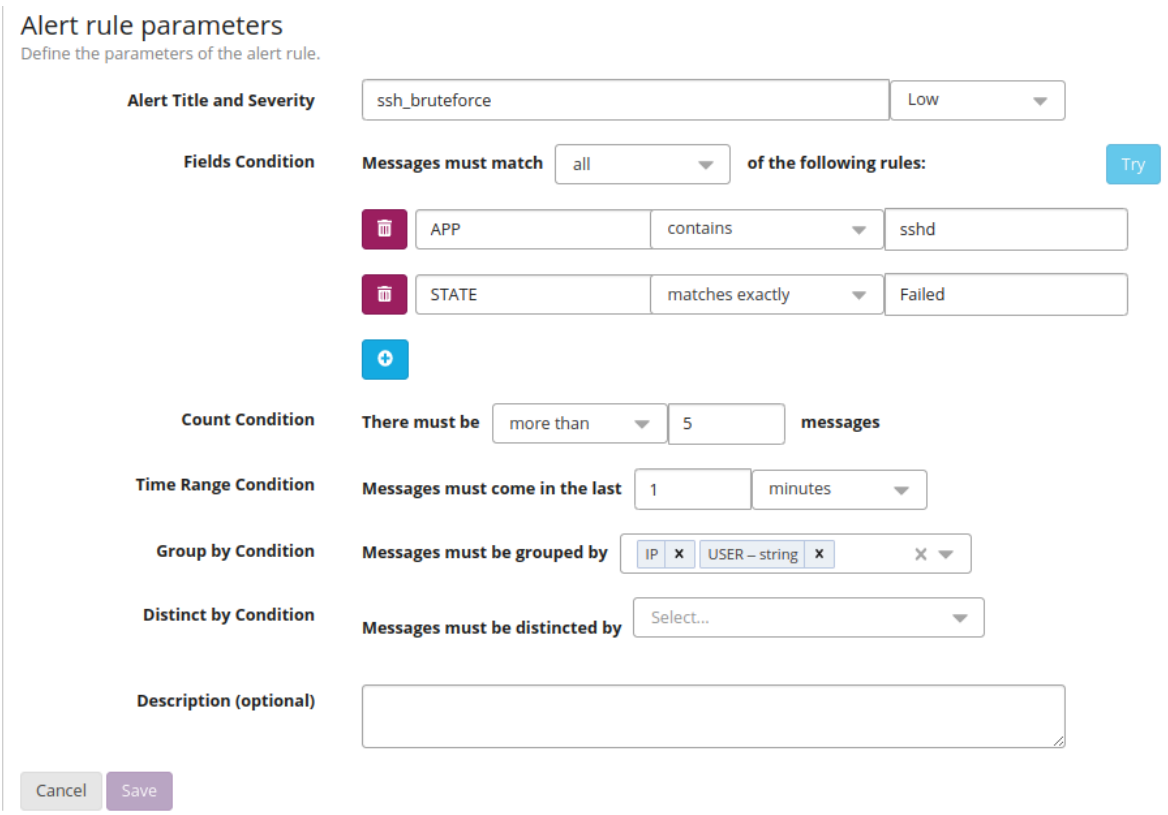


*Figura 5.2.2.2 Implementación alerta SIEM ssh_bruteforce*

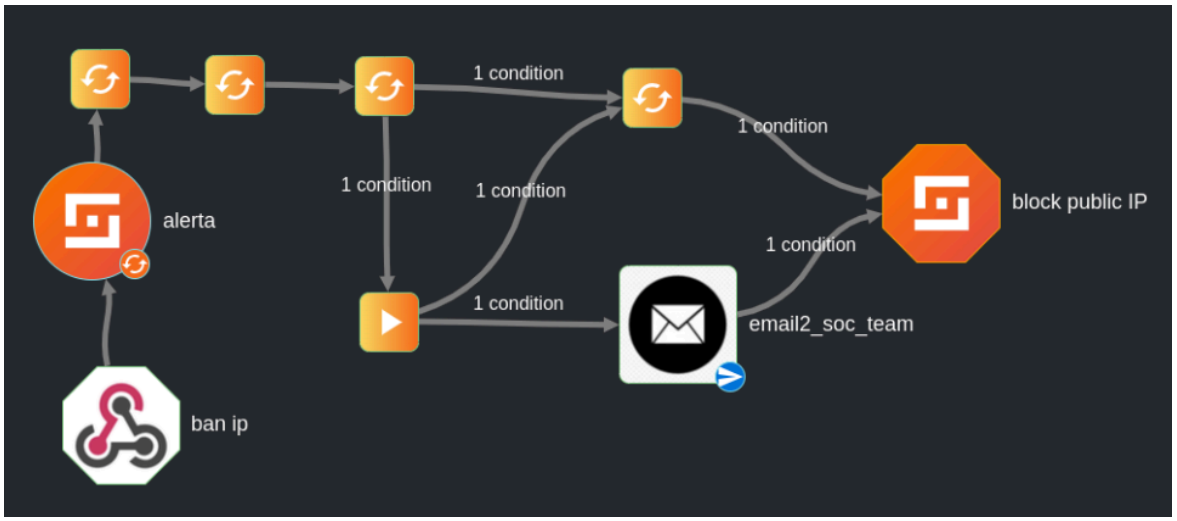


*Figura 5.2.2.3 Implementación workflow SOAR Fuerza bruta SSH*

### 5.2.3 Accesos a webs con URLs maliciosas conocidas

En el caso particular del entorno donde se ha implementado el CDU, los logs del cortafuegos son bastante completos y permiten encontrar todos los logs relativos a peticiones web aplicando los siguientes criterios:

El puerto destino es el 80 o el 443
La tecnología empleada para lanzar la petición es browser-based
Contiene el campo URLFilename

Una vez se han analizado y filtrado los logs, necesitan ser enriquecidos mediante Feeds (fuentes de información) externos a través de llamadas API. Este enriquecimiento se lleva a cabo con un Pipeline que añade a cada log el campo **suspiciousUrl**, el cual contiene un booleano que indica si el acceso es sospechoso o no. Si se detecta que es sospechoso se genera un evento que notifica al SOAR, el cual ejecuta el workflow correspondiente y bloquea la IP sospechosa.

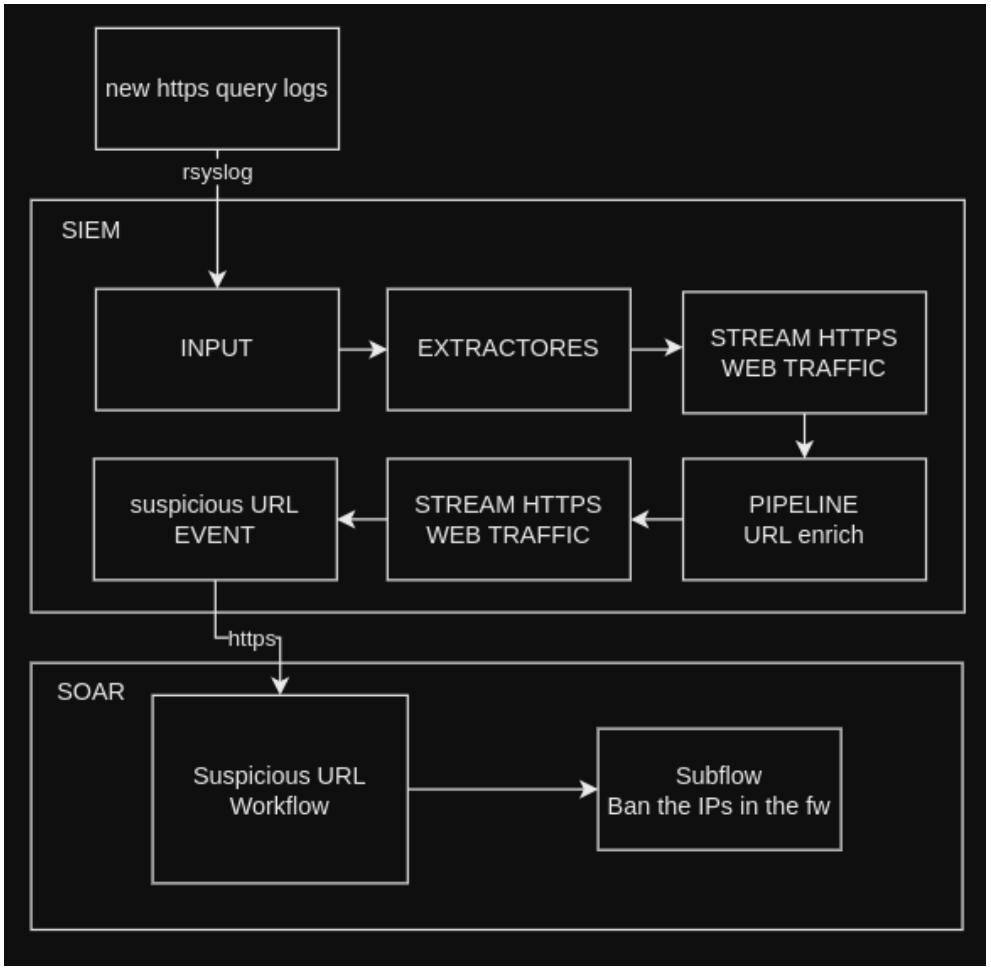


*Figura 5.2.3.1 Diagrama de implementación CDU Accesos a webs con URLs maliciosas conocidas*

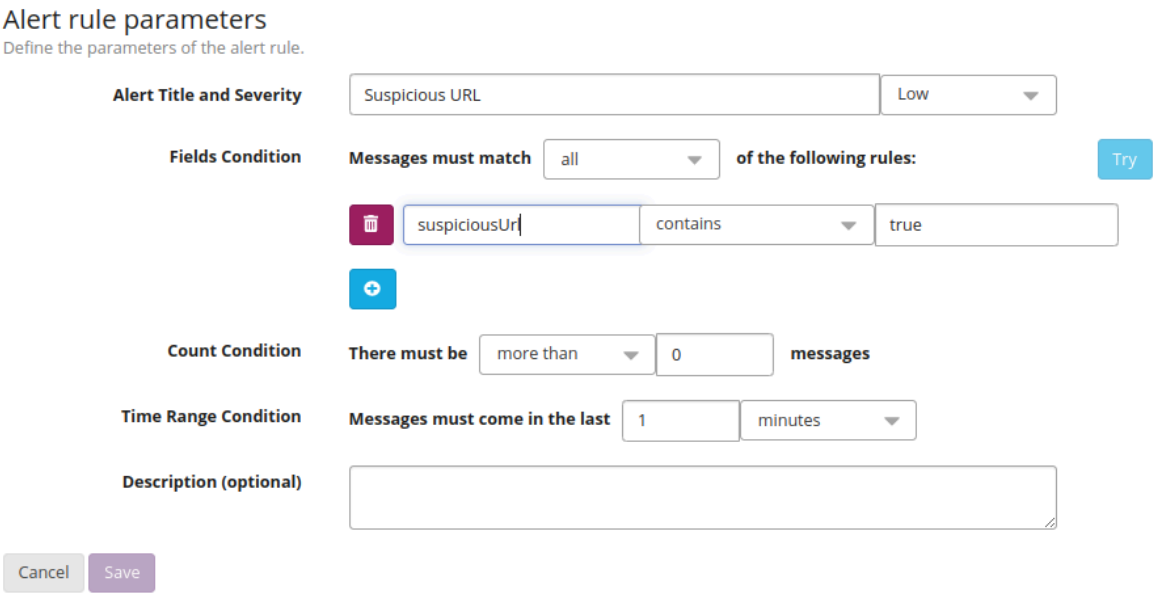


*Figura 5.2.3.2 Implementación alerta SIEM Suspicious URL*

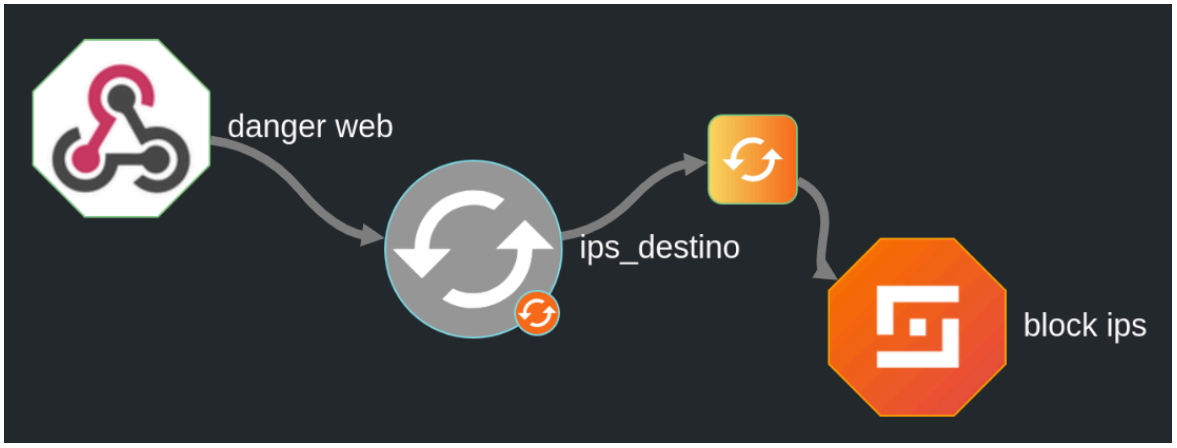


*Figura 5.2.3.3 Implementación workflow SOAR Web peligrosa*

### 5.2.4 Ataque DDoS

Con los logs ya en el SIEM, se clasifican según el tipo de log y se realiza el sumatorio de los Bytes que tienen los logs de los últimos 5 minutos. Si este valor supera un umbral predefinido, puede tratarse de un ataque DDoS.

Para fijar el umbral, se realizó un estudio de la carga de red del entorno de desarrollo y posteriormente se multiplicó por 10 para tener un margen y filtrar falsos positivos. Una vez detectado el ataque se notifica por HTTPs al SOAR y se dispara la ejecución del workflow **DDOS** que ejecuta las acciones necesarias para bloquear ese tráfico en el cortafuegos.

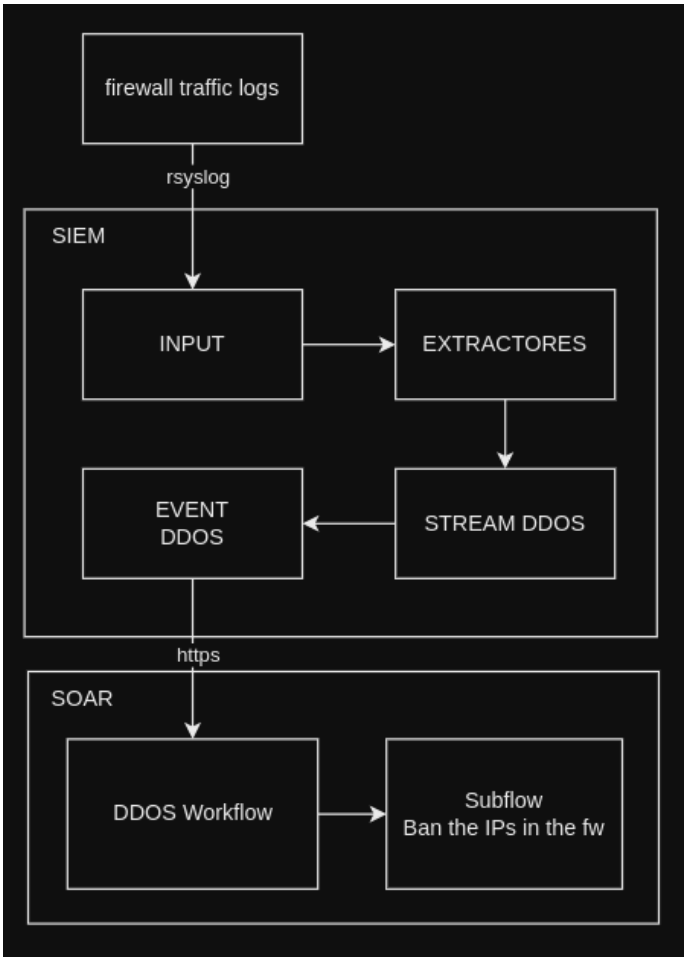


*Figura 5.2.4.1 Diagrama de implementación CDU Ataque DDOS*

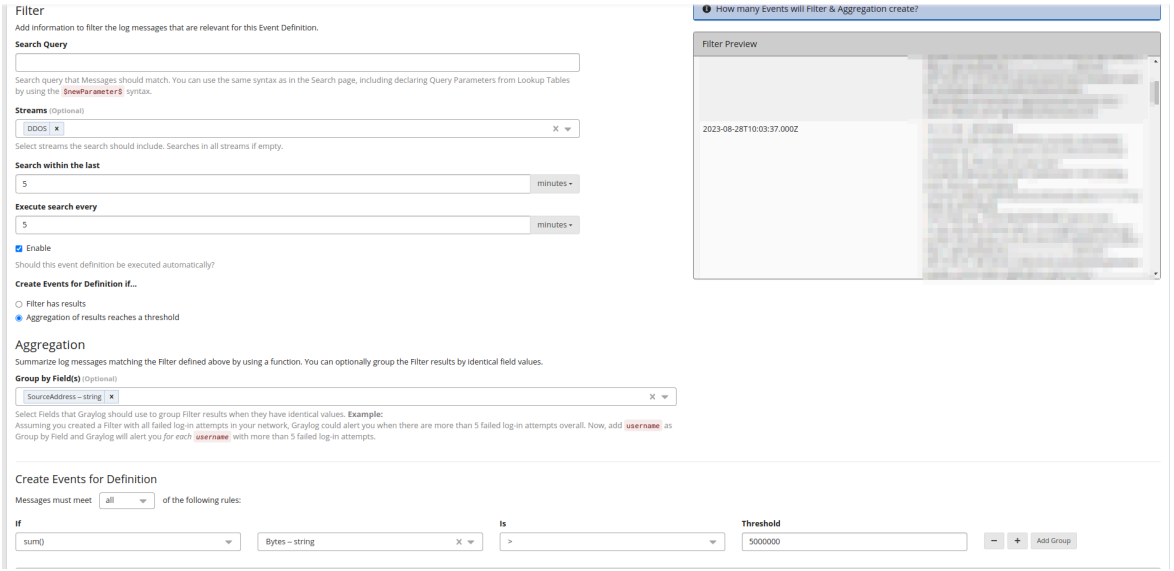


*Figura 5.2.4.2 Implementación alerta SIEM DDoS*

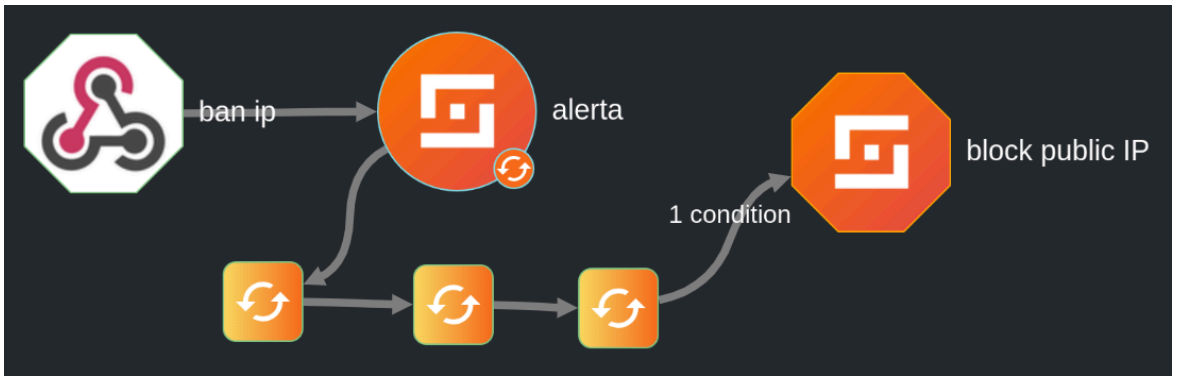


*Figura 5.2.4.3 Implementación workflow SOAR DDoS*

### 5.2.5 Data Leaks mediante USB y Ataque mediante Rubber Ducky

Para realizar el enriquecimiento en el SIEM se emplea la pipeline **not-allowed-usb-flag** (disponible en Anexo F.4.3), la cual consulta la LookUp Table de **allowed-usbs** (disponible en Anexo F.1) y consulta mediante un Data Adapter un CSV que contiene la lista de USBs autorizados. En caso de que sea un USB no autorizado se mueve el log al Stream **usb attached and not allowed** y genera el evento que notifica al SOAR y ejecuta el correspondiente workflow.

En una primera instancia se notifica al SOC para evitar un potencial Data Leak, y adicionalmente se emplea la API del EDR para escanear los ficheros y las vulnerabilidades de la máquina en la que se ha conectado el USB. Si se detecta algún fichero peligroso, la máquina se pone en cuarentena completa y se evitan los movimientos laterales.

La pipeline de enriquecimiento del SIEM cuenta con 2 etapas. La primera de ellas comprueba si el dispositivo está autorizado, y la segunda en caso de que no lo esté lo añade al stream de **usb attached and not allowed**.

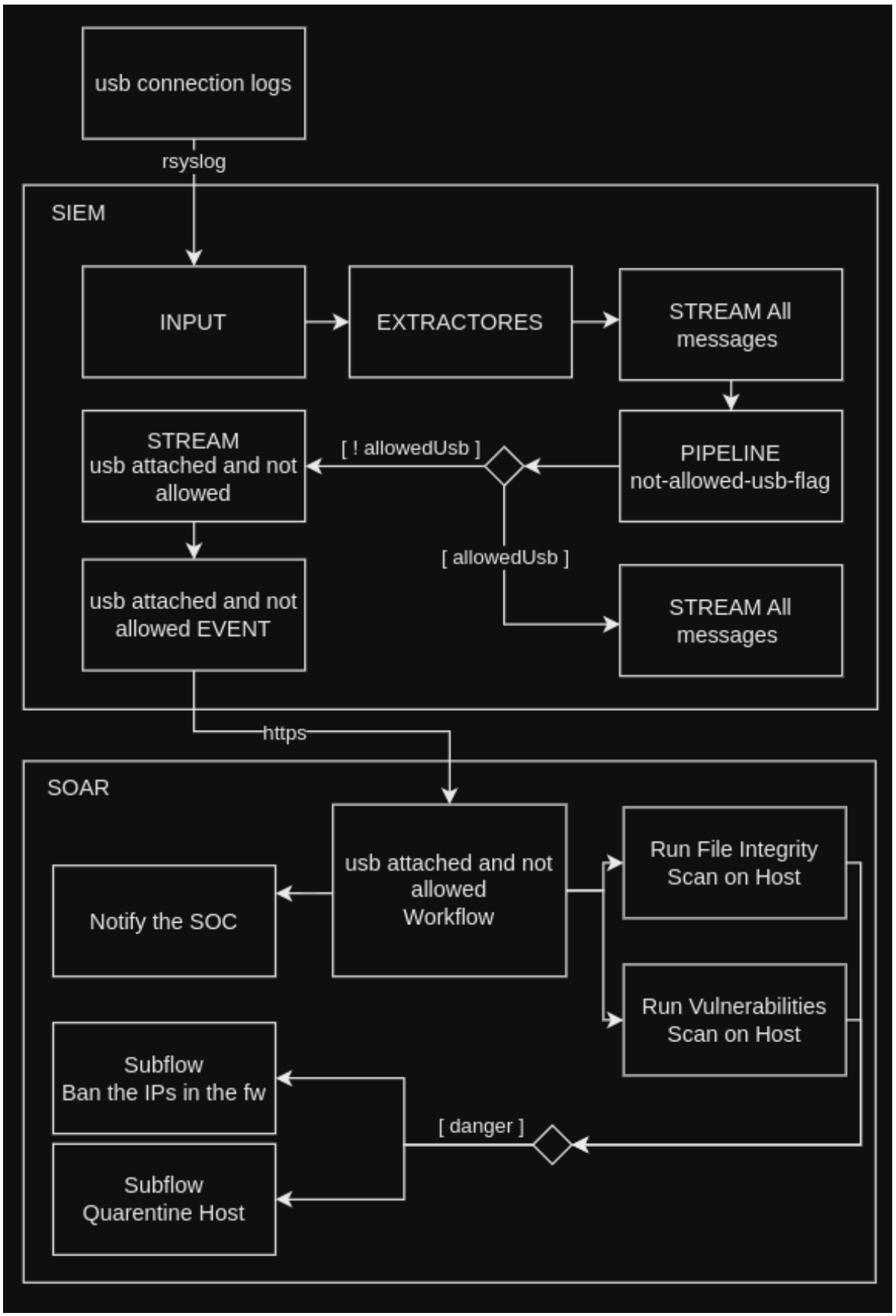


*Figura 5.2.5.1 Diagrama de implementación CDU USB Data Leaks y Ataque mediante Rubber Ducky*

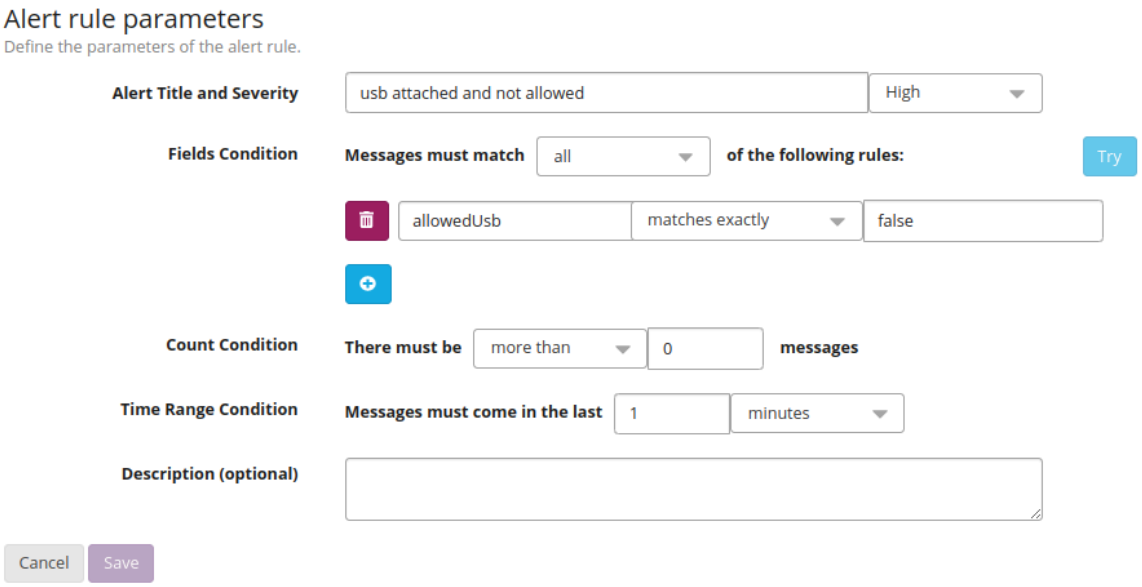


*Figura 5.2.5.2 Implementación alerta SIEM USB Attached and not allowed*

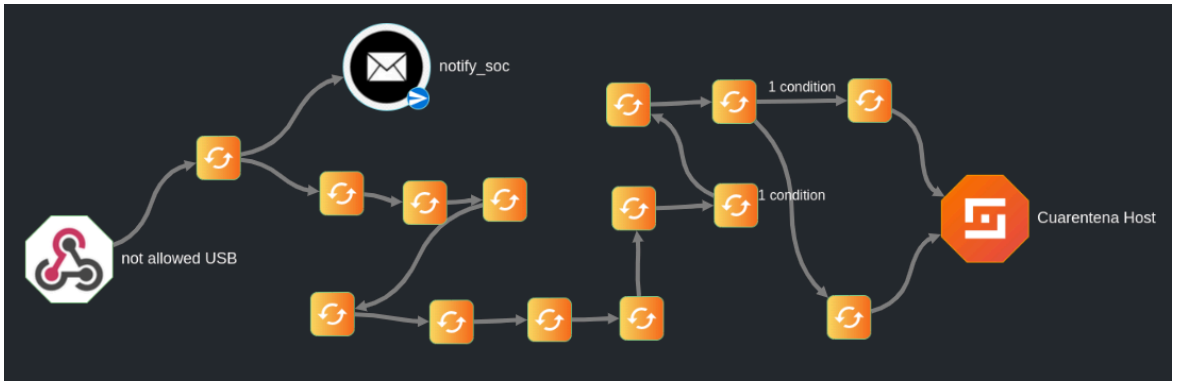


*Figura 5.2.5.3 Implementación workflow SOAR Conexión USB no autorizado*

### 5.2.6 Ataque de SQL Injection

Para que el EDR detecte el ataque, es necesario configurar el este para tal propósito, dicha configuración se encuentra disponible en el Anexo F.5.1

Una vez se detecte el ataque en el EDR, este notificará por medio de una integración al SOAR y se ejecutará el workflow **SQL Injection**, el cual emplea el subflow para bloquear la IP en el cortafuegos.

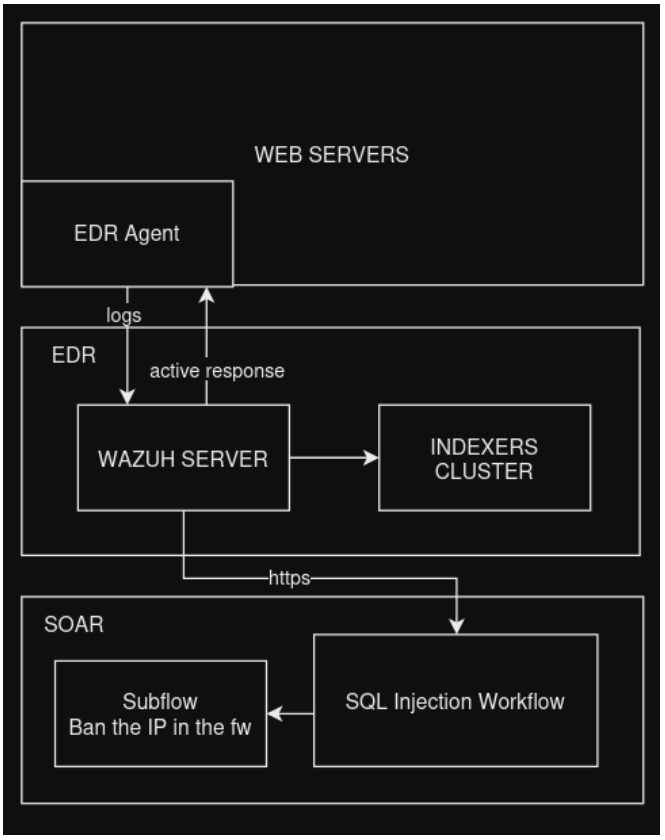


*Figura 5.2.6.1 Diagrama de implementación CDU Ataque de SQL Injection*

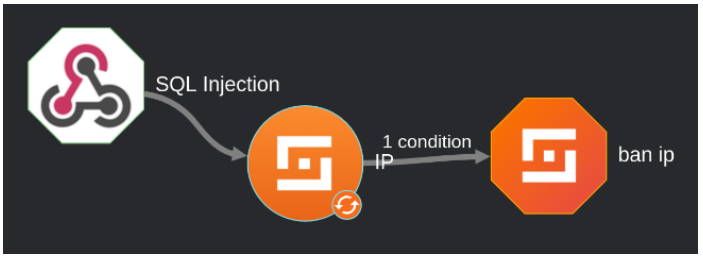


*Figura 5.2.6.2 Implementación workflow SOAR SQL Injection*

### 5.2.7 Accesos a la red TOR

Dado que desde fuera parece tráfico HTTPs común, para poder detectarlo se debe localizar el primer punto de entrada a la red TOR que utiliza el usuario (TOR tiene la estructura de nodos interconectados, cuyas entradas y salidas a dicha red se denominan Nodos Frontera). .

El SIEM los recibe y en una fase de enriquecimiento añade un campo a todas las conexiones de tipo HTTPs donde se indica con un booleano si la IP destino es o no un nodo conocido de la red TOR. La información de los nodos frontera se obtiene mediante la LookUp Table **Tor Exit Nodes** (disponible en Anexo F.1), la cual coteja cada IP de forma local contra un fichero que se renueva de forma periódica.

Tras detectarse una conexión con un nodo TOR, se lanza el evento correspondiente **access to TOR network** y se alerta al SOAR por medio de una notificación webhook que ejecuta el correspondiente workflow para alertar al equipo del SOC del acceso a la vez que se bloquea la IP del nodo TOR mediante la API del cortafuegos. De esta forma, toda comunicación con el nodo de entrada queda bloqueada.

La pipeline de este caso de uso cuenta con 2 etapas. La primera de ellas comprueba si una IP pública forma o no parte de los nodos conocidos de la red TOR La segunda pipeline, en el caso de que la IP forme parte, lo elimina del stream de **https web traffic** y lo añade al stream de **access to TOR network traffic**. La pipeline se encuentra disponible en F.4.4

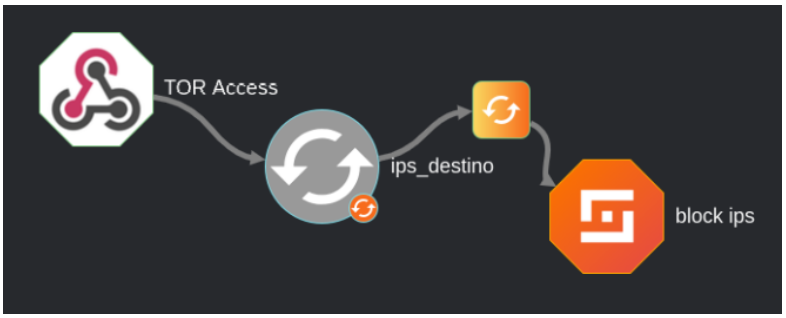


*Figura 5.2.9.1 Implementación workflow SOAR Acceso a la red TOR*

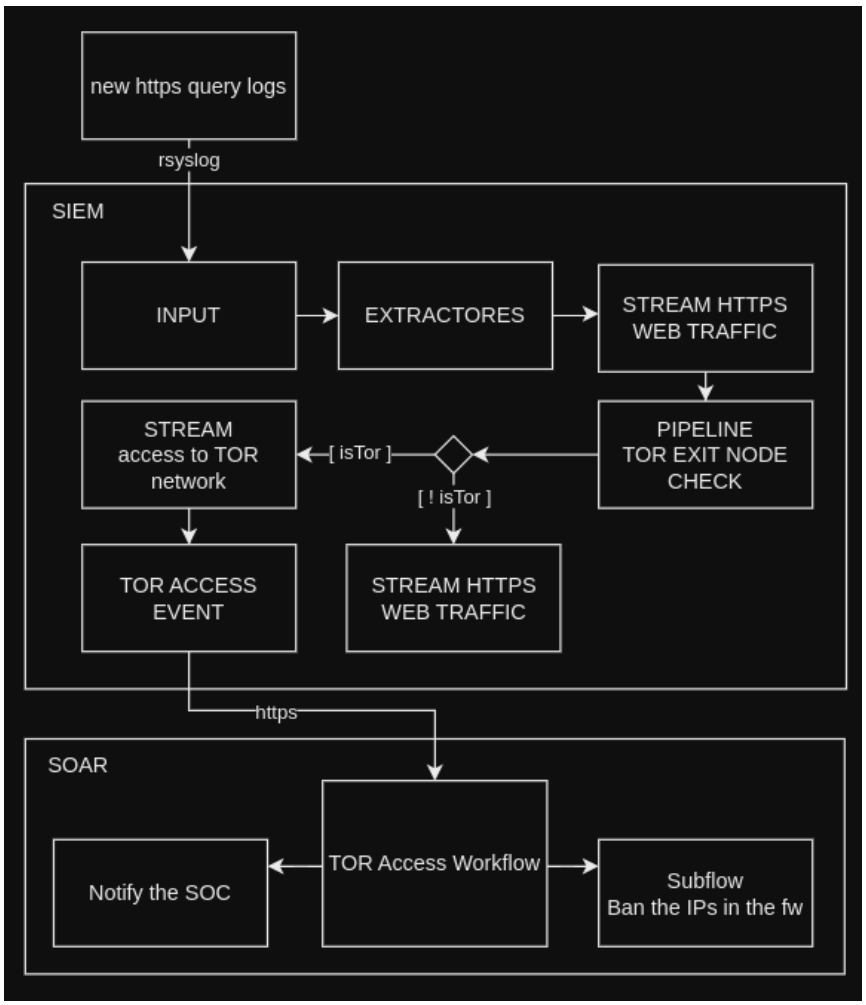


*Figura 5.2.7.2. Diagrama de implementación CDU Acceso a la red TOR*

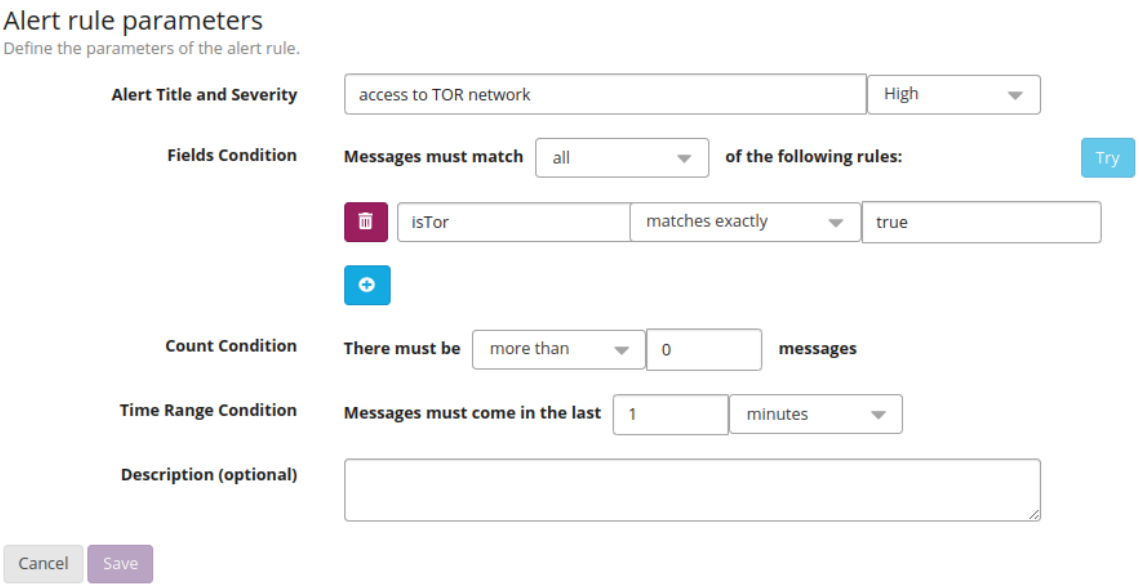


*Figura 5.2.7.3 Implementación alerta SIEM Access to TOR network*

### 5.2.8 Análisis y detección de anomalías en red

La implementación de este caso de uso requiere dos fases de funcionamiento. La primera de ellas se basa en el aprendizaje, en la cual el SIEM recibe los logs del cortafuegos y mediante el SOAR se añaden las IPs privadas a un fichero CSV. Tras un periodo de aprendizaje, esta lista es revisada y aprobada por un responsable de los servicios internos para confirmar las IPs vistas.

La segunda fase trata de detectar IPs anómalas, ya que mediante los logs del cortafuegos analiza si alguna de las IPs privadas es desconocida. Si es así y una IP privada no aparece en el CSV aprobado, el log analizado se mueve al stream de **not known IP** y se manda un evento al SOAR para que alerte al equipo SOC. El bloqueo de la IP no se ha configurado de forma automática para evitar pérdidas de servicio (análisis de riesgo y relación entre seguridad y servicio).

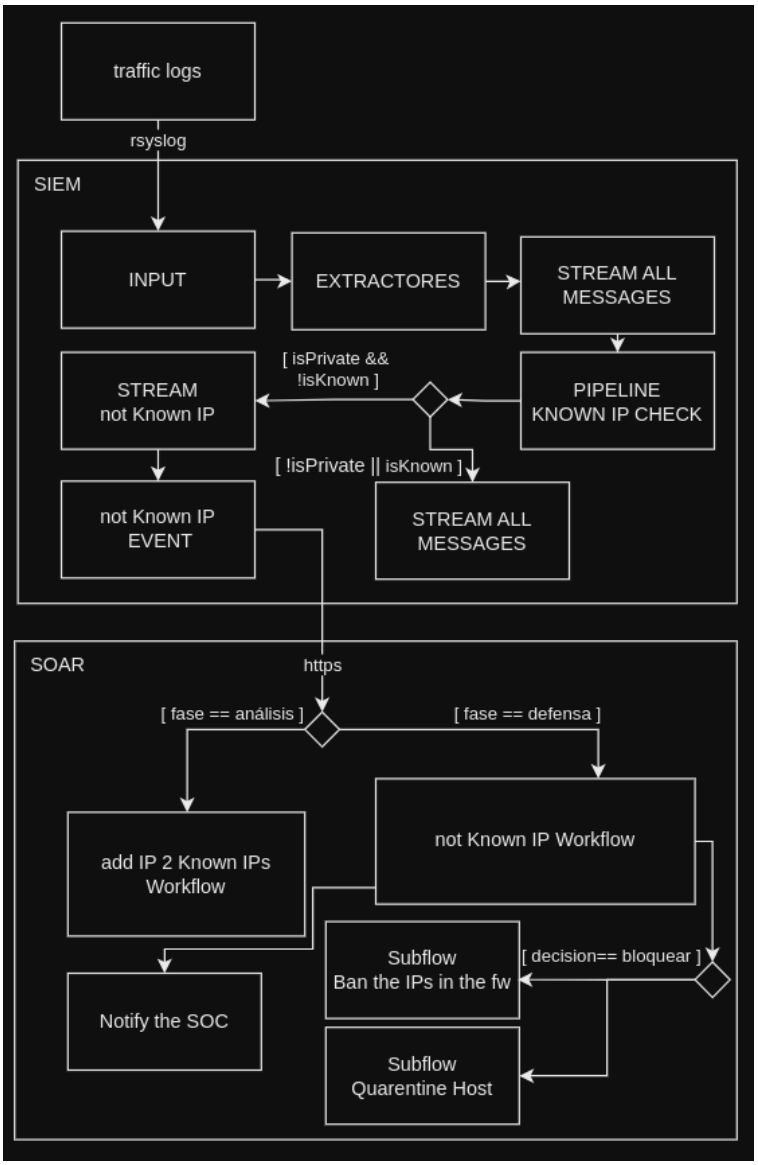


*Figura 5.2.8.1 Diagrama de implementación CDU Análisis y detección de anomalías en red*

La pipeline del SIEM cuenta con 3 etapas. La primera contiene 2 reglas que comprueban si un log tiene alguna IP privada y en caso de ser así almacena esta en el campo **privateIP**. La segunda, si la primera es verdadera, comprueba contra la LookUp Table si la IP privada está dentro de las definidas o no. Por último, la tercera etapa mueve el log al stream **not known IP** si no encuentra la IP en el listado de IPs definidas. La pipeline se encuentra disponible en el Anexo F.4.5.

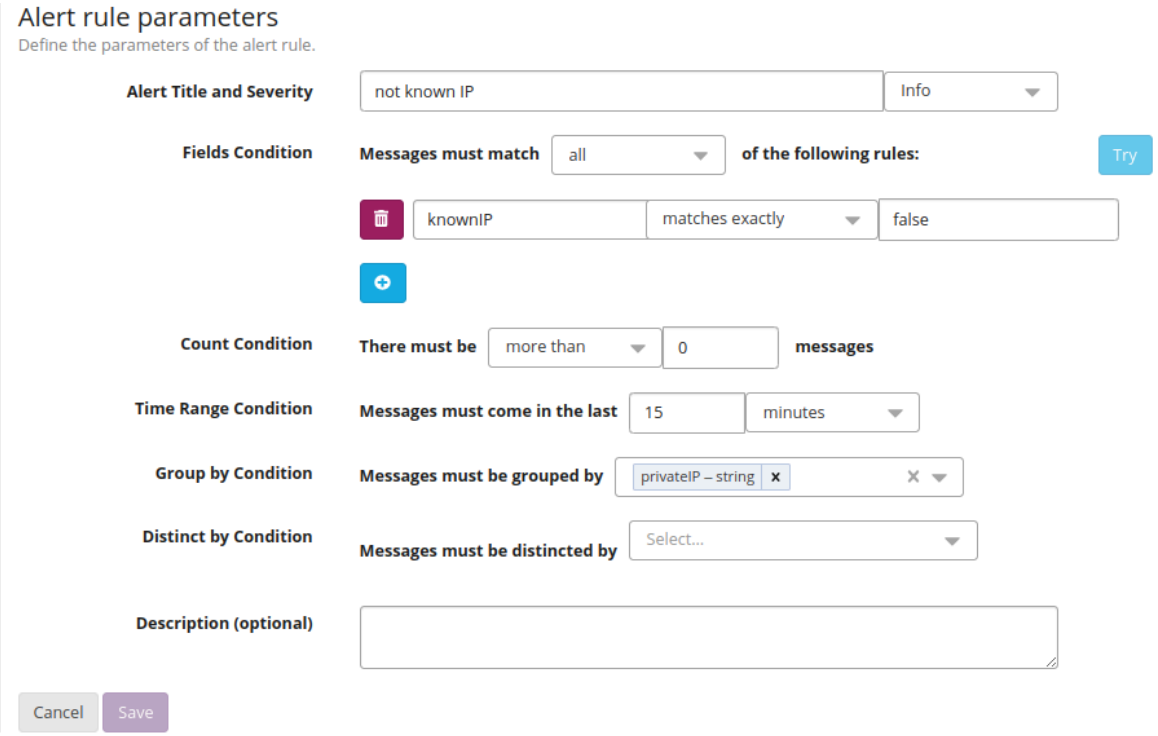


*Figura 5.2.8.2 Implementación alerta SIEM Not known IP*

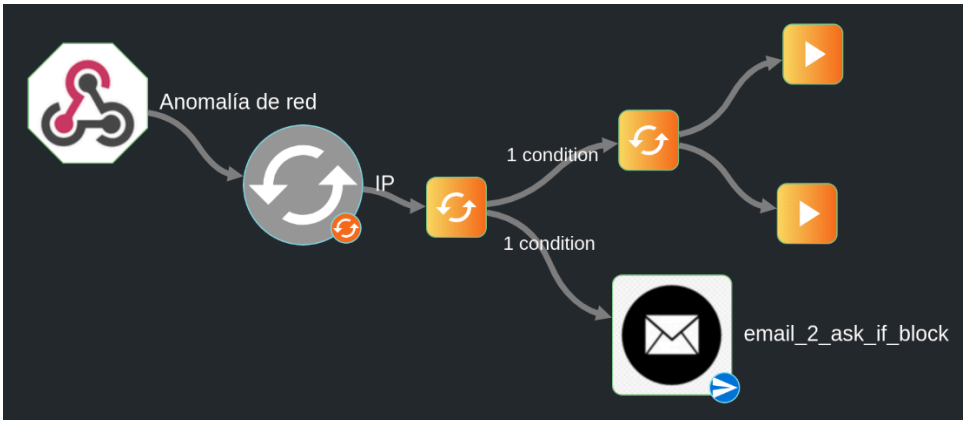


*Figura 5.2.8.3 Implementación workflow SOAR Análisis y detección de anomalías en red*

### 5.2.9 Implementación de herramienta Rescue-Terminal

Para implementar la Rescue-Terminal se ha optado por emplear las “active-response” de Wazuh, más concretamente se ha empleado un script de código propio programado en Python que abre una escucha en el Socket designado para acceder a la terminal con privilegios. Para acceder es necesario conocer una contraseña previamente definida. En caso de fallar la contraseña o de que en un tiempo de 5 minutos desde el lanzamiento del servidor de escucha no se haya autenticado nadie, este se cierra automáticamente por seguridad. Este enfoque es similar a lo que desde la perspectiva del atacante se denomina bind-shell. Para este CDU se ha modificado la configuración del EDR, esta se encuentra disponible en Anexo F.5.2.

Para lanzar esta respuesta activa de forma manual se ha empleado un workflow dentro del SOAR, el cual se comunica con la API del EDR y ejecuta el ya mencionado script en Python.

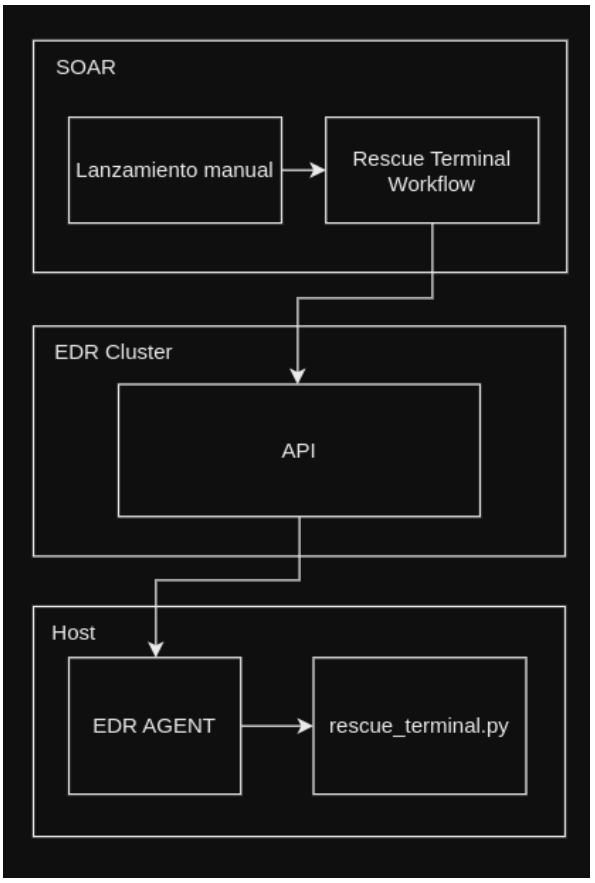


*Figura 5.2.9.1. Diagrama de implementación CDU Rescue-Terminal*

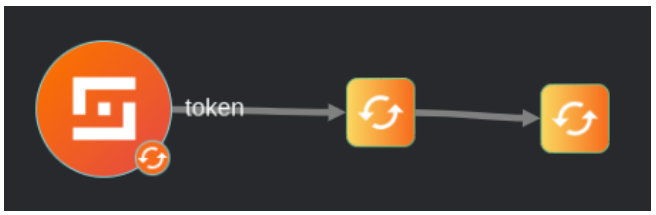


*Figura 5.2.9.2 Implementación workflow SOAR Rescue-Terminal*

### 5.2.10 Cuarentena completa de Hosts comprometidos

Una forma de solucionar el problema del desplazamiento lateral es dejar incomunicada una máquina si se determina que ha sido comprometida. Sin embargo, no se puede recurrir al cortafuegos ya que las comunicaciones dentro del mismo segmento de red no se bloquearían y se produciría el desplazamiento lateral; así pues, la máquina comprometida se tiene que incomunicar a nivel de equipo, es decir, de forma individual, aislarla de tal forma que no pueda recibir ni enviar tráfico de red. Para realizar este CDU se ha vuelto a modificar la configuración del EDR, el cambio se encuentra en el Anexo F.5.3.

Dado que la confirmación de si una máquina ha sido comprometida es un proceso que requiere por lo menos de una comprobación manual, este CDU al igual que el anterior se ejecuta de forma manual. Para implementarlo, se ha creado un workflow en el SOAR que recurre a la API del EDR para bloquear todo el tráfico entrante o saliente a la máquina comprometida empleando el comando ya predefinido de "firewall-drop".

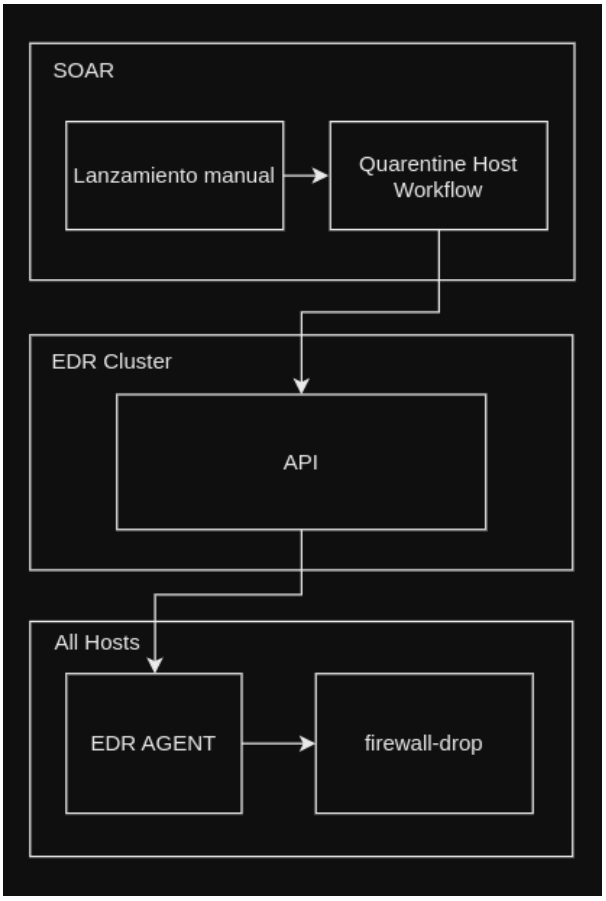


*Figura 5.2.10.1 Diagrama de implementación CDU Cuarentena completa de Hosts comprometidos*

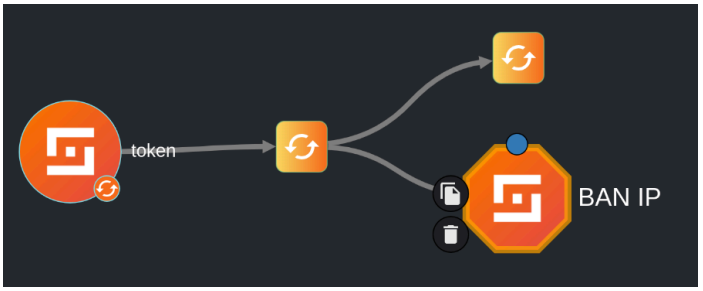


*Figura 5.2.10.2 Implementación workflow SOAR Cuarentena completa*

## 5.3 Configuración del SIEM

### 5.3.1 Configuración de clientes

Se debe configurar en cada cliente como corresponda el envío de logs al servidor Graylog, para entornos Linux la forma más sencilla es mediante el uso de la herramienta rsyslog. Para el despliegue en Nologin, se ha seguido este método.

En el propio SIEM se tiene que configurar, en el caso de que aún no exista, un INPUT asociado a ese protocolo en un puerto determinado. Si es necesario emplear un protocolo no estándar se puede recurrir a una máquina de transformaciones intermedias mediante un código propio.

### 5.3.2 Carga de configuraciones

Graylog contiene los denominados Content-Packs, los cuales permiten que a partir de un estado de la herramienta se genere una copia de seguridad de los distintos elementos y configuraciones del mismo. Esta copia se puede usar para hacer una instalación idéntica a la original de forma muy sencilla.

## 5.4 Configuración del SOAR

### 5.4.1 Carga de configuraciones

Para cargar todo lo configurado simplemente debemos emplear la interfaz web y subir todos los workflows previamente exportados.

Una vez hemos importado todo simplemente debemos acceder a los distintos Workflows y activar los triggers.

## 5.5 Configuración del EDR

### 5.5.1 Puesta en marcha de los agentes del EDR

Para lanzar un nuevo agente simplemente debemos ir a la interfaz web, rellenar un formulario y nos dará un comando para ejecutar, este debe ser ejecutado en el host que queremos incluir en nuestros endpoint monitorizados

### 5.5.2 Carga de configuraciones

Las modificaciones que siempre se deben realizar en la herramienta están incorporadas de forma automática en los scripts de instalación, no obstante si queremos cambiar la configuración de los endpoints esto es posible y sencillo mediante el uso de la configuración de grupos que se encuentra sincronizada por defecto.

# Capítulo 6

# Experimentación

Con el despliegue realizado sobre un entorno real, se han llevado a cabo algunas pruebas para verificar y probar la solución diseñada. Todos los resultados obtenidos y el detalle concreto de las pruebas exactas realizadas se encuentran en el Anexo G.

## 6.1 Pruebas de carga

Una parte relevante del funcionamiento del sistema que ha estado presente desde el análisis de requisitos es que este pueda gestionar altas cargas de trabajo, es por ello que se han realizado pruebas de carga individuales sobre cada uno de los componentes para poner a prueba dicha propiedad. Para ello se ha llevado al límite cada uno de los diferentes elementos de la solución:

SIEM: El sistema soporta perfectamente cargas excesivamente altas de logs en cuanto a procesado se refiere. Se ha visto que el principal cuello de botella no es el SIEM, sino la red o el balanceador de carga.

EDR: Se ha comparado la utilización de recursos físicos en los distintos nodos conforme se aumentaba la cantidad de equipos monitorizados mediante un agente. Los resultados muestran que el número de agentes no supone un incremento significativo en el consumo de recursos.

SOAR: Se ha analizado el aumento del tiempo de respuesta para un determinado workflow conforme el número de instancias simultáneas del mismo workflow aumentaban hasta llegar a 50. Asimismo, se ha comprobado cómo cambia el tiempo de ejecución total de 100 workflows conforme cambia el número de nodos workers.

Las gráficas generadas durante las pruebas se pueden encontrar en el Anexo G.
La conclusión general de las distintas pruebas de carga es que el sistema tolera de forma satisfactoria grandes cargas de trabajo y que en caso de ser necesario este sistema puede escalar para aumentar la capacidad de procesado de dicha carga de trabajo.

## 6.2 Pruebas de escenarios de fallo

Estas pruebas fueron realizadas durante la fases de diseño e implementación para asegurar que cada uno de los elementos aseguraba la propiedad de HA o en su defecto la de DR.

La metodología para el desarrollo de las pruebas fue observar el funcionamiento de cada una de las herramientas conforme sus distintos nodos van “fallando”. En todos los casos (exceptuando los nodos que hacen que el sistema solo tenga DR y no HA), se pudo perder 1 nodo de cada tipo y el sistema seguía funcionando en las mismas condiciones sin que la pérdida se notara a nivel de servicio.

## 6.3 Pruebas de CDUs

Se ha seguido una metodología de desarrollo basada en pruebas, es decir, una vez se comenzaba la operación del SOC para cierto CDU, no se avanzaba a otro sin haber completado este y sin que este haya pasado las pruebas concretas del mismo. Dicho de otra forma, se han probado todos los CDUs de forma iterativa e intensiva uno a uno, a lo largo del desarrollo. Las diferentes pruebas realizadas para cada uno se detallan en el Anexo G. Para las pruebas se han empleado en todo momento situaciones reales a excepción del CDU del phishing, en el que en vez de emplear el servidor principal de la empresa, por la importancia de este mismo, se ha empleado un servidor Postfix, no obstante los mails que han sido testeados contra el sistema si eran reales.

# Capítulo 7

# Conclusiones

El proyecto, tras sus distintas fases, ha alcanzado y sobrepasado los objetivos planteados inicialmente. Durante su realización se han añadido nuevos escenarios que inicialmente no habían sido planteados, como el de las Rescue-Terminal, para enriquecer tanto la funcionalidad como el alcance del proyecto. Asimismo, para continuar enriqueciendo la solución planteada, se añadió el EDR, el cual de forma inicial solo se había planteado como una remota posibilidad dada la limitación temporal de este trabajo académico. Otro aspecto que se mejoró durante el proyecto fue que inicialmente solo se debía plantear una solución distribuida conformada por varios servicios, sin que estos tuviesen necesariamente que ser distribuidos, escalables y tolerantes a fallos, no obstante dichos aspectos fueron tomados en consideración para plantear la solución final.

Además, la solución propuesta ha conseguido subsanar mediante distintos enfoques las carencias de algunas de las herramientas empleadas para poder alcanzar los requisitos especiales comentados en los objetivos del proyecto, más concretamente los aspectos de *alta disponibilidad y recuperación de desastres*.

Desde un punto de vista económico, también se ha logrado demostrar que es posible realizar una solución de seguridad informática empresarial de bajo coste, a pesar de la creencia generalizada de que segurizar una empresa de forma adecuada y profesional resulta extremadamente caro.

Dado que el proyecto se ha realizado sobre una red real en un entorno de producción, esto ha causado ciertas dificultades a lo largo del mismo. Las características especiales del proyecto relativas al HA y DR también han tenido sus implicaciones problemáticas sobre la red.
Otras dificultades que destacan han sido el estado del arte difuso y la mala documentación de algunas herramientas.

Todo este trabajo ha sido posible gracias a los distintos proyectos OpenSource y a toda la comunidad que los respalda. Tras compararlo con un entorno de soluciones comerciales queda demostrado que a pesar de las limitaciones de estos proyectos, estos son lo bastante potentes para brindar un buen servicio como solución SOC. Es cierto que la cantidad de esfuerzo requerido inicialmente puede ser algo mayor dado que estos no son soluciones *Out of the Box*, pero, no obstante, es posible de llevar a cabo tal y como aquí queda demostrado.

Cabe destacar que siguiendo la filosofía Open Source, durante el desarrollo de este proyecto se ha contribuido de forma proactiva en la mejora de una de las herramientas empleadas en el mismo, concretamente el SOAR, tanto reportando bugs como mejorando la documentación de la herramienta empleada, incluyendo en esta el diseño arquitectural de DR planteado durante el proyecto.

## 7.1 Trabajo futuro

Los siguientes pasos serían emplear este proyecto en clientes finales de forma gradual, enriqueciendo durante este proceso dicho proyecto mediante la explotación de nuevos CDUs o la mejora de los ya existentes teniendo en cuenta aspectos mucho más detallados.

Otro posible punto de mejora sería añadir más herramientas a la solución, que en este caso no se han incorporado dada la magnitud esperada de este trabajo académico. Tales herramientas podrían ser un NIDS o un HoneyPot, entre otras cosas.

Un último punto que se podría mejorar sería un trabajo de desarrollo de código, más concretamente del SOAR empleado, de forma que el frontend y el backend de dicha herramienta sean compatibles con un despliegue HA, como lo son las demás herramientas empleadas en este TFG.

# Bibliografía

[1] M. Vielberth, F. Böhm, I. Fichtinger and G. Pernul, "Security Operations Center: A Systematic Study and Open Challenges," in *IEEE Access*, vol. 8, pp. 227756-227779, 2020, doi: 10.1109/ACCESS.2020.3045514. https://ieeexplore.ieee.org/abstract/document/9296846

[2] Verizon Data Breach Investigations Reports (DBIR). https://www.verizon.com/business/resources/reports/dbir/2023/summary-of-findings/

[3] Threat landscape 2022 European Union Agency for Cybersecurity, ENISA. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022

[4] 11 Strategies of a World-Class Cybersecurity Operations Center. https://www.mitre.org/sites/default/files/2022-04/11-strategies-of-a-world-class-cybersecurity-operations-center.pdf

[5]Instituto Nacional de Ciberseguridad, INCIBE. SOC. https://www.incibe.es/incibe-cert/blog/respondiendo-incidentes-industriales-soc-ot

[6] Glosario CCN-CERT. https://www.ccn-cert.cni.es/publico/seriesCCN-STIC/series/400-Guias_Generales/401-glosario_abreviaturas/index.html

[7] Graylog. Cluster architecture. https://go2docs.graylog.org/5-0/planning_your_deployment/planning_your_deployment.html?tocpath=Planning%20Your%20Deployment%7C_____0

[8] Shuffle. Architecture. https://shuffler.io/docs/architecture

[9] Wazuh. Architecture. https://documentation.wazuh.com/current/getting-started/architecture.html

[10] Cuadrante mágico Gartner SIEM. https://www.fortinet.com/content/dam/fortinet/images/diagrams/diagram-gartner-mq-siem.jpg

[11] Guía de mercado del SOAR Gartner. https://www.gartner.com/doc/reprints?id=1-2E8WFZSW&ct=230623&st=sb

[12] Cuadrante mágico Gartner EDR. https://www.crowdstrike.com/wp-content/uploads/2023/03/Picture1-1.png

# Anexo A

# Gestión del tiempo

El proyecto se ha desarrollado a lo largo de 4 meses en NOLOGIN invirtiendo alrededor de 350 horas. A continuación se puede observar el diagrama de Gantt en la figura A.1 donde se puede ver el tiempo invertido cada semana por tareas y por subproyecto. Dado que este TFG fue realizado antes de haber cursado por primera vez la asignatura de seguridad informática, una vez cursada la asignatura se revisó el contenido de esta memoria en Enero.

Las distintas fases han sido el análisis y estudio del funcionamiento de cada una de las herramientas a implementar, el estudio de las herramientas que se van a emplear para alcanzar dichas implementaciones, una vez elegidas las herramientas concretas diseñar el sistema e implementarlo, una vez implementado este debe ser probado. Además se debe documentar todo tanto de forma técnica para la propia empresa como para este TFG por lo que la documentación se va realizando durante el desarrollo y no únicamente al final.

| Fases | Junio | | | Julio | | | | Agosto | | | | Septiembre | Enero | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 1 | 2 |
| Análisis y estudio | | | | | | | | | | | | | | |
| Diseño | | | | | | | | | | | | | | |
| Implementación | | | | | | | | | | | | | | |
| Explotación | | | | | | | | | | | | | | |
| Pruebas | | | | | | | | | | | | | | |
| Documentación | | | | | | | | | | | | | | |

*Tabla A.1 Diagrama de Gantt*

| Parte del proyecto | Color |
|---|---|
| SIEM | |
| SOAR | |
| EDR | |
| SIEM + SOAR | |
| SIEM + SOAR + EDR | |
| Nada | |

*Tabla A.2 Leyenda del diagrama de Gantt*

# Anexo B

# Detalle herramientas del estado del arte

## B.1 SIEM

Para extraer los requisitos relativos al SIEM, se ha tomado como referencia la solución de IBM, dado que este es uno de los mejor valorados y dada la relación de partners existente entre IBM y Nologin. Tras analizar dicho SIEM se extrajeron los siguientes requisitos. Se adjunta un informe profesional de la situación actual de las herramientas de propietario para justificar dicha elección.


*Figura B.1.1 Magic Quadrant for Security Information and Event Management[10]*

A continuación se explica de forma resumida el funcionamiento y los conceptos asociados a la solución de referencia elegida. La herramienta distingue entre 2 tipos de fuentes de datos dependiendo del origen de los datos *Flow data* (routers/switches) y *Event Data* (todo lo demás). En ambos casos los datos se recogen mediante los Flow Collector y los *Event Collector* respectivamente, los datos son normalizados y parseados. Posteriormente los datos son procesados por los *Flow Processor* y los *Event Processor* respectivamente, en esta etapa son ejecutados por el *Custom Rules Engine (CRE)*, este es el responsable de generar las *alertas* y las *ofensas*. Existe una 3º capa que permite a los usuarios realizar búsquedas sobre los datos para analizar,reportar e investigar las *alertas* y *ofensas*.

## B.2 SOAR

Dado que para el SOAR no existe una comparativa realizada por la empresa Gartner, se han extraído los requisitos de un reporte de mercado sobre los elementos clave de un SOAR elaborado también por dicha entidad[11], adicionalmente se ha realizado un estudio de la solución comercial de PaloAlto Networks que emplea la empresa actualmente.

A continuación se explica de forma resumida el funcionamiento y los conceptos asociados a la solución de referencia elegida. La herramienta basa su funcionamiento en *Playbooks*, cada *playbook* contiene en su interior una serie de acciones a ejecutar, se permiten ejecuciones condicionales así como ejecuciones repetitivas. Los *Playbooks* a su vez cuentan con 2 componentes interesantes para ser más efectivos, el primero de ellos son las denominadas *Integrations* estas permiten de forma muy simple interactuar con otros elementos comunes en una red empresarial o del propio SOC. El segundo elemento son *Scripts* programados en Python o Javascript, estos permiten al equipo encargado de manejar la herramienta crear sus propias utilidades.

## B.3 EDR

Para extraer los requisitos se han empleado los EDR de Palo Alto Networks y de CrowdStrike dado que estos están bien valorados y se disponía de ambas opciones. Tras analizar dichos EDRs se extrajeron los siguientes requisitos.


*Figura B.3.1 Magic Quadrant for Endpoint detection and response[12]*

A continuación se explica de forma resumida el funcionamiento y los conceptos asociados a la solución de referencia elegida. La herramienta consta de un servidor centralizado y de unos agentes instalados en los endpoints, los agentes analizan todo lo que ocurren en los endpoints y envían dichos datos al servidor central. El servidor central en base a datos irregulares, patrones de comportamiento y uso de inteligencia compartida por medio de cloud detecta las potenciales amenazas conteniendo estas antes de que se manifiesten. Ofrece un listado detallado de la situación de cada agente monitorizado, destaca la capacidad de poder acceder de forma remota como usuario privilegiado a una máquina para aplicar parches de seguridad sobre la misma o para realizar investigaciones forenses mediante el uso de la denominada *Live-Terminal*.

# Anexo C

# Comparativa de las herramientas

## C.1 SIEM

Durante el análisis se contemplaron más opciones que no aparecen recogidas en la tabla. Estas fueron descartadas por diversas razones, tales como una documentación escasa y pobre o por ser soluciones muy complejas que abarcan demasiados aspectos aparte de los deseados.

| **COMPARATIVA SIEM** | | | | |
|---|---|---|---|---|
| Requisito | Graylog | ELK | Fluentd | Wazuh |
| RG1 | | | | |
| RG2 | | | | |
| RG3 | | | | |
| RG4 | | | | |
| RF1 | | | | |
| RF2 | | | | |
| RF3 | | | | |
| RF4 | | | | |
| RF5 | | | | |
| RF6 | | | | |
| RF7 | | | | |
| RF8 | | | | |
| RF9 | | | | |

*Tabla C.1.1 Comparativa de requisitos soluciones SIEM*

De entre las soluciones estudiadas y reflejadas en la tabla, las 2 únicas realmente viables específicamente como SIEM son Graylog y ELK, no obstante se ha elegido emplear Graylog dado que la documentación oficial es más completa y detallada. De entre las otras 2 opciones, Fluentd se descartó por ser simplemente un agregador de logs, y Wazuh porque a pesar de tener capacidades de EDR, le faltaban algunas características como la correlación de logs entre distintos sistemas monitorizados y el enriquecimiento de logs.

## C.2 SOAR

Al igual que con el análisis de soluciones SIEM, se contemplaron más opciones que no aparecen recogidas en la tabla, pero estas fueron descartadas por diversas razones tales como ser proyectos archivados y sin soporte o tener como enfoque principal otras tareas de forma que la funcionalidad SOAR pasa a ser algo secundario.

| COMPARATIVA SOAR | | |
|---|---|---|
| Requisito | Catalyst | Shuffle |
| RG1 | | |
| RG2 | | |
| RG3 | | |
| RG4 | | |
| RF1 | | |
| RF2 | | |
| RF3 | | |
| RF4 | | |
| RF5 | | |
| RNF1 | | |

*Tabla C.2.1 Comparativa de requisitos soluciones SOAR*

De entre las 2 soluciones posibles, en la tabla queda reflejado que específicamente como SOAR resulta mejor opción Shuffle. Catalyst se ha descartado además de por no cumplir todos los requisitos, porque está planteada como una herramienta de ticketing para investigación, estando todo el sistema centrado en tickets, no obstante en la alternativa de Shuffle nos encontramos con un SOAR en el sentido más estricto, ya que se encuentra centrado exclusivamente en workflows (concatenación de acciones en base a entradas y condicionales) y en aplicaciones que son utilizadas en dichos workflows.

## C.3 EDR

No se han encontrado tantas soluciones EDR *open source* como en el resto de apartados ya comentados. Al ser tecnologías quizá algo más avanzadas, la mayoría de los productos de este tipo son de pago. No obstante, y tal y como aparece en la tabla anterior, se han podido analizar tres interesantes productos que casi en su mayoría satisfacían los requisitos del proyecto.

| **COMPARATIVA EDR** | | | |
|---|---|---|---|
| Requisito | Ossec | Wazuh | OpenEDR |
| RG1 | | | |
| RG2 | | | |
| RG3 | | | |
| RG4 | | | |
| RF1 | | | |
| RF2 | | | |
| RF3 | | | |
| RF4 | | | |
| RF5 | | | |

*Tabla C.3.1 Comparativa de requisitos soluciones EDR*

De entre las 3 posibles opciones, OpenEDR a pesar de ser bastante completa quedó completamente descartada dado que solo se encontraba disponible para entornos Windows. De entre las otras 2 opciones se eligió Wazuh debido a que esta se basa en Ossec, un sistema de detección de intrusos HIDS (Host Intrusion Detection System) multiplataforma, aportando todas las funcionalidades y reglas de este pero en un formato más accesible y con una mejor documentación.

# Anexo D

# Detalles del análisis de herramientas

## Graylog

Los aspectos más significativos de la herramienta son los siguientes:

- Índices: Almacenan los mensajes guardados, contienen las configuraciones a nivel de sharding, replicación, criterios de orden de los mensajes almacenados y capacidad máxima.
- Adaptadores: Se encargan de dada una Key consultar un feed ya sea interno o externo, para ello puede consultar ficheros CSV o realizar consultas de API.
- Lookup tables: Funcionan como tablas HASH, es decir para una key nos devuelven el valor asociado, se emplean para enriquecer los logs.
- Pipelines: Secuencias de acciones de procesado avanzado de logs de forma individual para su enriquecimiento
- Streams: Una vez un log ha pasado por todas las etapas de procesado pertinentes, se almacena en los streams, cada stream almacena los logs en el índice que tenga configurado, las correlaciones se realizan sobre estos.
- Inputs: Servicios de escucha de los logs, tienen asociado un puerto, protocolo de comunicación a nivel de red y a nivel de aplicación.
- Extractors: Distintos tipos de parser para extraer la información del log bruto, entre ellos destacan las expresiones GROK que son agrupaciones de expresiones REGEX.
- Alert rules(plugin): Permite configurar de forma muy rápida y sencilla las búsquedas periódicas que disparan los eventos, asimismo crea dichos tipos de eventos y sus notificaciones básicas asociadas.

## Shuffle

Los aspectos más significativos de la herramienta son los siguientes:

- Workflows: Serie de acciones a ejecutar cuando se dispara cierto trigger, permiten ejecución condicional, llamadas a otros workflows y ejecución en bucle.
- Triggers: Elementos encargados de lanzar los workflows, son webhooks, ejecuciones programadas.
- Apps: Todas las acciones se realizan por medio de lo que se denominan Apps, estas ofrecen una interfaz muy clara para realizar ciertas acciones comunes, por ejemplo mandar un correo electrónico. Además en caso de necesitarlo se pueden crear Apps propias de forma sencilla.
- Liquidity: Sintaxis que permite la manipulación de datos de forma sencilla teniendo para ello comandos como puede ser “first” para quedarnos con el primer elemento de una lista.

# Wazuh

Los elementos más interesantes de la herramienta son los siguientes:

- Agentes: Son los nodos monitorizados, nos permite ver de forma rápida el estado general del conjunto así como de forma granular el estado individual de cada uno de ellos.
- Security Events: Nos permite visualizar de forma rápida los últimos eventos significativos en cuanto a seguridad tales como un acceso Root, la escucha en un puerto, también permite ver el listado completo de eventos.
- Integrity Monitoring: Muestra los hashes de los distintos ficheros relevantes para el SO, en Windows adicionalmente también monitoriza las claves de registro, así como también muestra un historial de los cambios realizados sobre dichos ficheros.
- Security configuration assessment: Evalua en base a los benchmark elaborados por el CIS(Center for Internet Security), este comprueba desde paramétros de kernel hasta permisos de ficheros.
- Vulnerabilities: De forma periódica se realizan análisis del sistema en busca de amenazas conocidas.
- MITRE ATT&CK: Analiza el sistema teniendo en cuenta técnicas observadas en el mundo real, clasificando estas por tácticas lo cual permite un análisis bastante granular.
- Regulatory Compliance: Analiza los requerimientos que exigen distintas normativas de referencia para establecer si un sistema es seguro o no. Por defecto cuenta con las siguientes normativas PCI DSS, GDPR, HIPAA, NIST 800-53, TSC.
- Inventory data: Permite visualizar en tiempo real las especificaciones de hardware y sistema operativo, las interfaces de red, puertos activos, los paquetes instalados y los procesos activos.
- Rules: Implementan la parte de detección mediante el análisis simple de logs, se pueden personalizar.
- Active Response: Lanza de forma inmediata un comando(un script python) cuando se activa una regla concreta, esto permite una primera defensa instantánea hasta que todo el sistema en su conjunto responde con la defensa a nivel global.

# Anexo E

# Detalles de la arquitectura interna de las soluciones

## E.1 SIEM

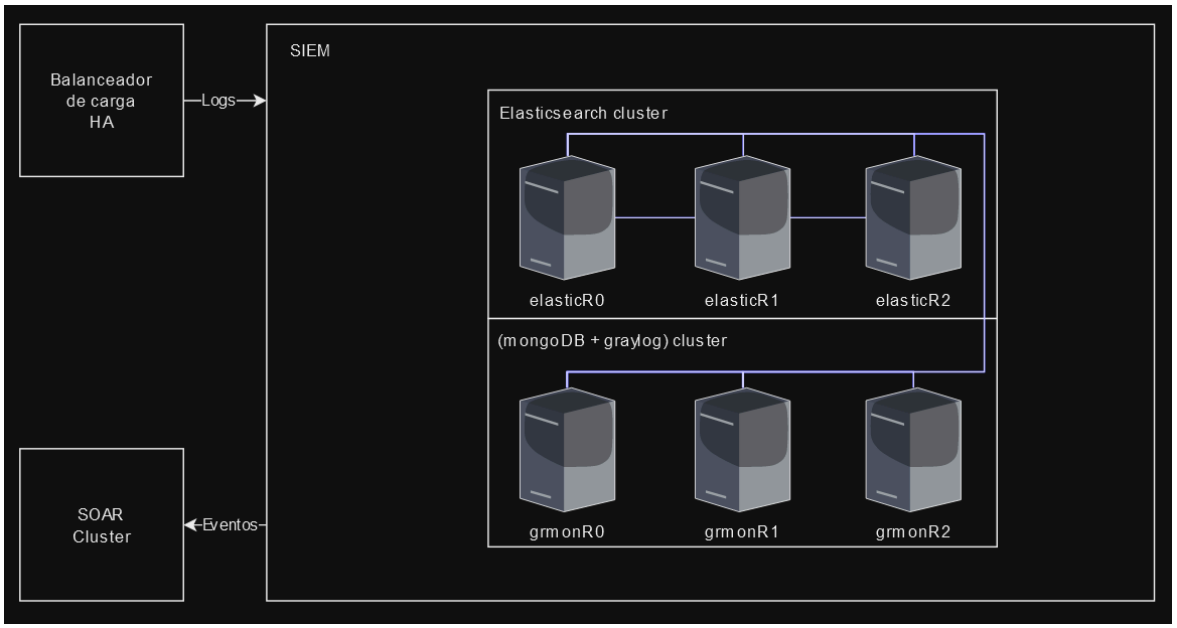


*Figura E.1.1 Arquitectura interna del SIEM*

Por un lado las máquinas elasticR se encargan de ejecutar la base de datos distribuida ElasticSearch. En esta configuración todos los nodos tienen los mismos roles. No obstante si el sistema a implementar fuese de un gran tamaño se podrían asignar roles concretos a cada nodo, pudiendo distinguir entre los siguientes roles: master, data, data_content, data_hot, data_warm, data_cold, data_frozen, ingest, ml, remote_cluster_client y transform. Su labor en el SIEM es la de guardar toda la información relativa a los logs, índices y streams, además de ser usado por el motor de búsqueda.

Por el otro lado nos encontramos con las máquinas grmonR, las cuales ejecutan la base de datos distribuida MongoDB en formato de replicaset, este formato se traduce en varias réplicas, una de ellas es la primaria y es con la que se interactúa vía API, esta replica la información en los nodos secundarios, todos los nodos se envían latidos entre sí. Esta base de datos es empleada por el SIEM para guardar toda la información de sus configuraciones tales como las definiciones de eventos y alertas entre otras. El principal servicio de estas máquinas es Graylog, y cada máquina tiene su propia interfaz web. No obstante lo interesante es que la carga de trabajo de procesado de logs se reparte entre ellas dado que no son independientes sino que están clusterizadas tal y como se puede apreciar.

El SIEM recibe los logs desde los balanceadores de carga en HA y envía las notificaciones de eventos al SOAR.

# E.2 SOAR

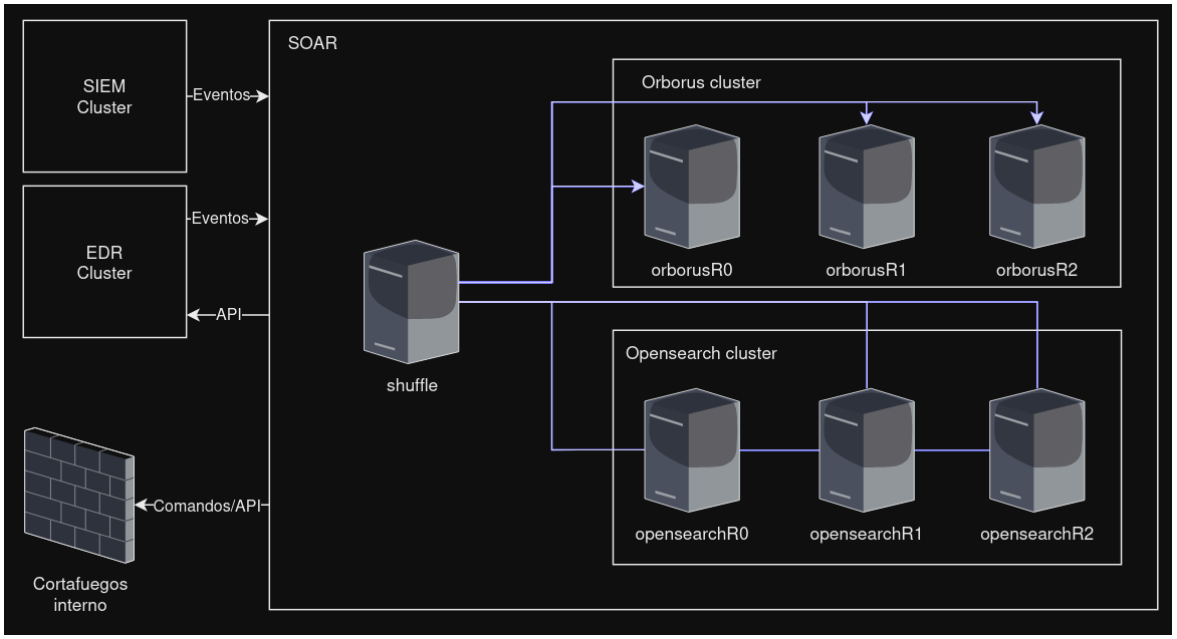


*Figura E.2.1 Arquitectura interna del SOAR*

De forma similar a la arquitectura anterior, contamos con una base de datos distribuida, esta concretamente es Opensearch, y en ella se almacena toda la información relativa al SOAR, desde los Workflows, Apps, Ficheros y variables de entorno configuradas hasta cada una de las distintas ejecuciones de Workflows con una traza de ejecución completa.

Asimismo contamos con un cluster de nodos Worker, estos son los denominados Orborus. Son los encargados de ejecutar cada paso de los Workflow cuando estos son lanzados. Este tipo de despliegue permite escalar de forma lateral la cantidad máxima de ejecuciones paralelas de Workflows.

Por último para ejecutar tanto el backend como el frontend contamos con el denominado nodo Shuffle. En esta ocasión, no se ha configurado una arquitectura HA para este tipo de nodo debido a que todavía no se ha desarrollado una buena gestión de la concurrencia para esta parte de la herramienta y que la misma no está probada. No obstante, como es un requisito del proyecto, se ha implementado un mecanismo automatizado de disaster recovery (DR) con el que el nodo es levantado en cuestión de segundos de forma rápida y sin pérdida de información.

Tanto los nodos Orborus como el nodo Shuffle se han desplegado empleando Docker como herramienta de control de contenedores.

El SOAR recibe alertas por parte del SIEM y del EDR. Este envía llamadas a API/comandos remotos o bien al EDR o bien al cortafuegos interno para que estos se apliquen sobre el mismo o sean reenviados a otros puntos del sistema.

# E.3 EDR

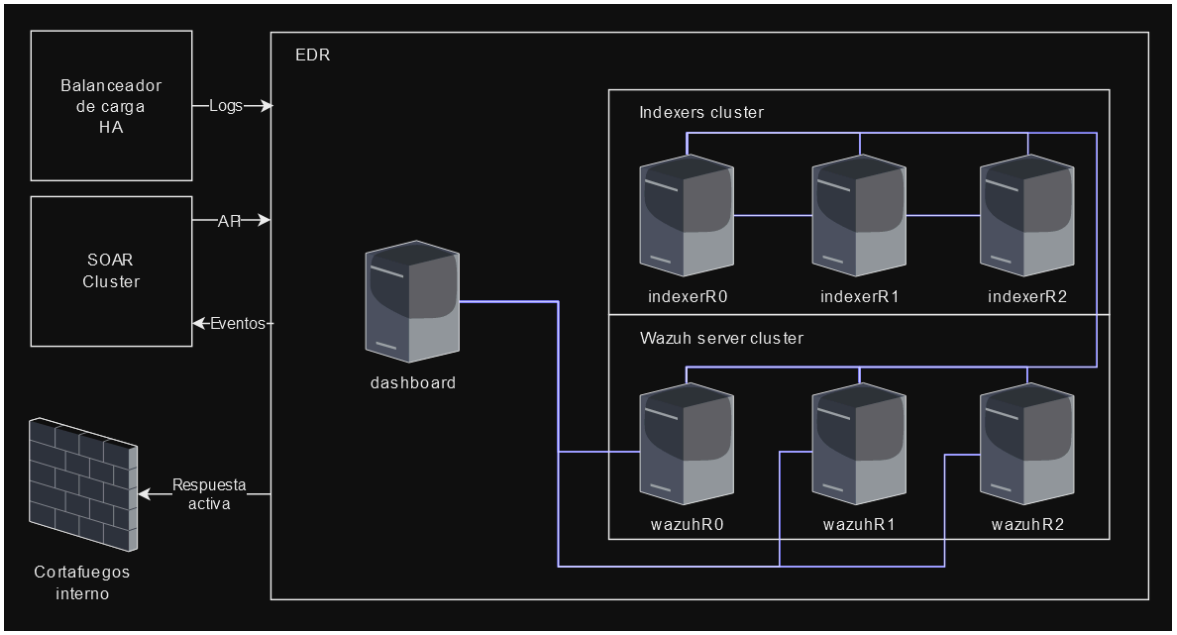


*Figura E.3.1 Arquitectura interna del EDR*

De forma similar a las arquitecturas anteriores, la herramienta emplea una base de datos distribuida, esta concretamente está en los nodos indexerRX, estos emplean por debajo la base de datos Elasticsearch, en ella se almacena toda la información relativa a los endpoints y a las configuraciones de la herramienta.

También contamos con un cluster de nodos servidor. Estos son los encargados de recibir y procesar toda la información de los endpoint, donde uno de ellos actúa como master y los otros dos como workers. Son los denominados wazuhRX.

Por último para ejecutar el frontend contamos con el denominado nodo dashboard. A pesar de que el despliegue de un solo nodo de este tipo evita que obtengamos el ya mencionado HA, dado que este solo contiene el frontend web no resulta relevante para el funcionamiento propio del sistema, solo es útil para el analista que lo quiera emplear, por lo que a pesar de ser un único punto de fallo no existiría pérdida alguna de funcionalidad o información. De cualquier manera, se ha generado un procedimiento de DR asociado.

Para que los agentes se puedan comunicar con los servidores Wazuh y enviarles las comunicaciones pertinentes se ha decidido emplear una estructura intermedia de balanceadores de carga que distribuyen toda la información recibida a los distintos nodos servidor.

El EDR puede recibir llamadas a API desde el SOAR, adicionalmente también puede enviar notificaciones de eventos a este. Para enviar las respuestas activas a los endpoints se envían a través del cortafuegos interno.

# E.4 Balanceador de carga en HA

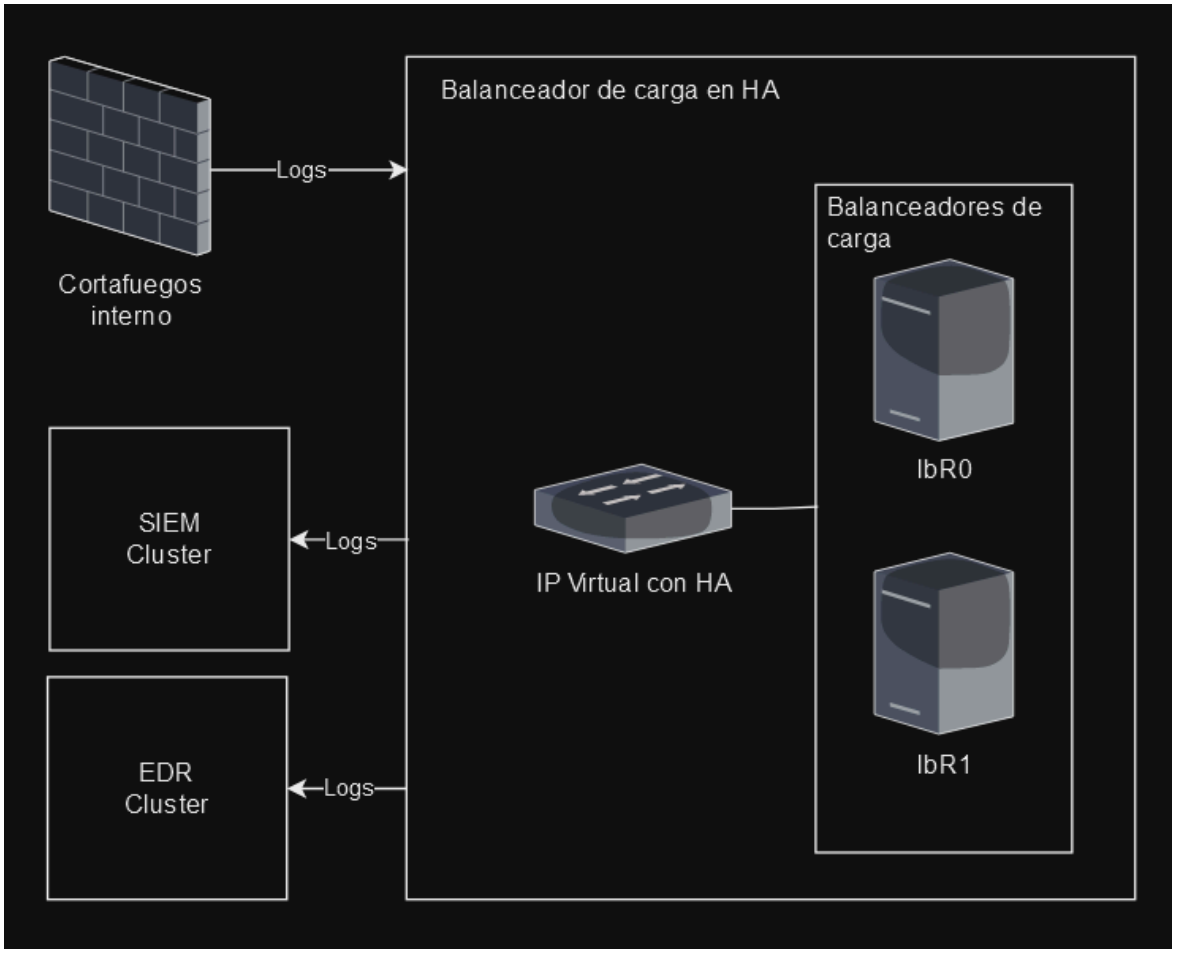


*Figura E.4.1 Arquitectura interna de los balanceadores de carga en HA*

Dado que se agrega mucha información de entrada tanto para el SIEM como para el EDR y esta necesita ser distribuida de forma equivalente entre los servidores que conforman cada uno de dichos clusters, surge la necesidad de crear un balanceador de carga, no obstante dado el contexto y el requerimiento de la HA, no es suficiente con un balanceador.

Para crear una estructura de balanceador de carga en HA, primero se definen una serie de réplicas de balanceadores de carga que comparten una misma configuración. Para tener siempre una IP disponible aunque alguno de los balanceadores de carga falle se emplea una IP virtual, la cual en caso de que un balanceador caiga, se autoasigna a otra de las réplicas de los balanceadores de forma que se mantiene el servicio.

El balanceador en HA recibe los logs provenientes del cortafuegos interno y los envía al EDR o al SIEM según corresponda.

Como implementaciones concretas de balanceador se ha empleado Nginx, como IP virtual se ha decidido emplear Keepalived.

# Anexo F

# Elementos básicos comunes

En este anexo se encuentran todos los elementos básicos configurados en cada una de las herramientas, para hacer posible el funcionamiento de los distintos CDUs implementados.
Concretamente a continuación se detallan los siguientes elementos:

- Lookup Tables: Son las encargadas de hacer posible el enriquecimiento en el SIEM.
- Subflows: Son workflows que son llamados desde los workflows principales en el SOAR, se usan para reutilizar la implementación de las funcionalidades repetidas en distintos workflows.
- Pipelines: Manipulan los logs en el SIEM, hacen uso de las Lookup Tables y modifican los logs y donde se almacenan.
- Configuraciones EDR: Son pequeños fragmentos de código que se deben incluir en la configuración del EDR, para hacer posible el funcionamiento de algunos de los CDUs que lo utilizan.

## F.1 Lookup Tables

Para la fase de enriquecimiento de los logs se emplean las denominadas Lookup tables. Cada una de estas se compone de una memoria caché y de un Data Adapter. El Data Adapter se encarga de realizar consultas sobre una clave que se le pase, pudiendo obtener información sobre dicha clave tanto a través de API externas como de ficheros CSV que se encuentren dentro del propio Graylog o en un host remoto. Para poder elaborar los distintos CDUs se han empleado las siguientes Lookup tables:

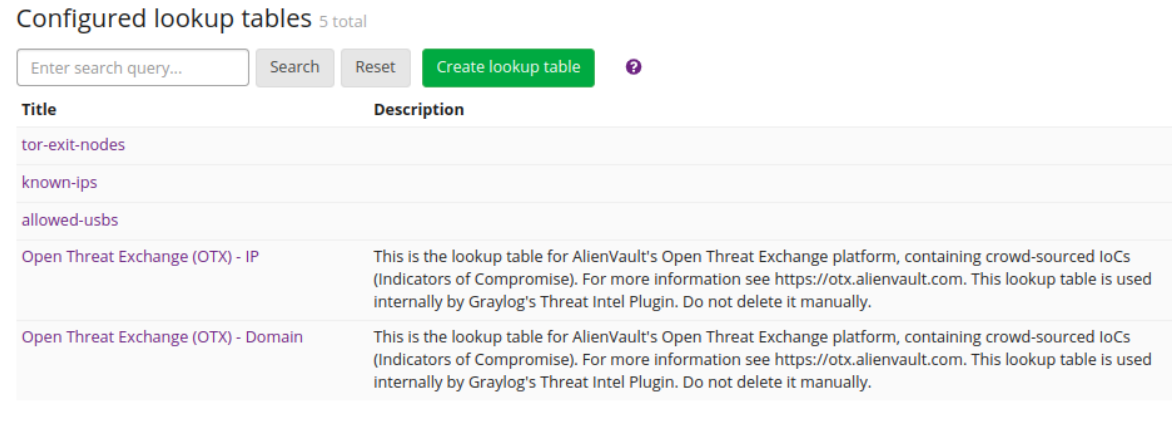


*Figura F.1.1 SIEM LookUp Tables*

Aunque hay una gran cantidad de feeds externas con las que obtener información, para los CDUs propuestos se han utilizado APIs que obtienen las IPs de los nodos conocidos de TOR, así como las feeds de AlienVault (OTX) para dotar al sistema de unos IOCs (indicadores de compromiso) confiables.

## F.2 Subflows

A lo largo de los distintos CDUs se dan ciertas tareas comunes repetitivas, como por ejemplo el bloqueo de una IP en el cortafuegos. Es por ello por lo que para no tener que implementarlas de forma repetitiva en cada uno de los Workflow principales, se han implementado mediante Workflows secundarios denominados subworkflows.

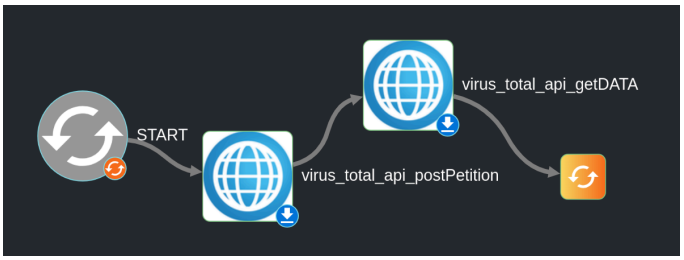


*Figura F.2.1 SOAR Subflow Analizar URLs VirusTotal*

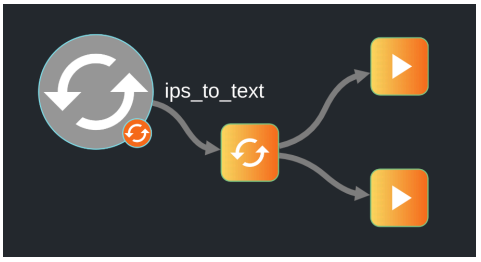


*Figura F.2.2 SOAR Subflow Bloquear IP en el cortafuegos*

En el primer subworkflow se realizan dos peticiones sobre la API de VirusTotal la primera de ellas se encarga de escanear una determinada URL y la segunda se encarga de obtener los resultados del análisis previamente realizado. El segundo subworkflow realiza una primera etapa de parseo y posteriormente actualiza el CSV que contiene las IPs bloqueadas, dicho CSV es leído de forma periódica por el cortafuegos mediante las denominadas listas dinámicas externas, posteriormente en el cortafuegos hay una regla encargada de bloquear toda IP que se encuentre en dicha lista.

## F.3 Scheduled Workflows

Otra de las capacidades interesantes del SOAR es la ejecución de forma automática de Workflows cada cierto tiempo. Concretamente, para aportar la interactividad a uno de los CDUs desarrollados (Análisis y detección de anomalías en red), se ha optado por que una vez tomada la decisión de bloquear una IP por parte del analista, este envíe un correo electrónico que sigue cierto formato a una dirección especial de correo electrónico. De esta manera, el SOAR revisa de forma periódica la bandeja de entrada de esta cuenta especial y en caso de existir algún correo que siga el formato especificado el SOAR bloquea la IP y elimina el correo de la bandeja de entrada.

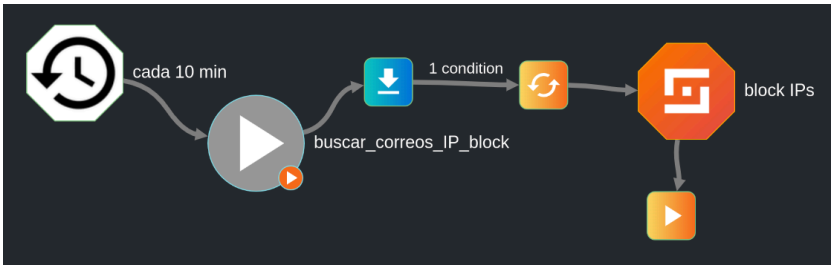


*Figura F.3.1 SOAR Scheduled Workflow Revisar correos para bloquear IPs*

# F.4 Pipelines

## F.4.1 Dangerous mail

Stage 1:

```
rule "Check IP against Threat Intelligence"
when
   has_field("HOSTNAME") and contains(to_string($message.APP),"postfix/qmgr")
then
  let result = threat_intel_lookup_domain(to_string($message.hostname), "domain_name");
  set_field("isBad", result);
end
```

*Tabla F.4.1.1 SIEM Pipeline Dangerous mail Stage 1*

Stage 2:

```
rule "Route to dangerous MAIL and remove from new_mail"
when contains(to_string($message.isBad),"true")
then
   route_to_stream("dangerous_mail");
   remove_from_stream("new_mail");
end
```

*Tabla F.4.1.2 SIEM Pipeline Dangerous mail Stage 2*

## F.4.2 URL Enrichment

```
rule "Check URL against Threat Intelligence"
when
   has_field("URLFilename")
then
   let result = threat_intel_lookup_domain(to_string($message.URLFilename), "domain_name");
   set_field("suspiciousUrl", to_string(result));
end
```

*Tabla F.4.2.1 SIEM Pipeline URL Enrichment Stage 1*

## F.4.3 Allowed USBs

Stage 1:

```
rule "Check USB against allowed USBs"
when
   has_field("ID") and has_field("NUMBER")
then
   let result = lookup_value("allowed-usbs",to_string($message.ID) + to_string($message.NUMBER));
   set_field("allowedUsb", result);
end
```

*Tabla F.4.3.1 SIEM Pipeline Allowed USBs Stage 1*

Stage 2:

```
rule "Route to usb attached and not allowed"
when contains(to_string($message.allowedUsb),"false")
then
   route_to_stream("usb attached and not allowed");
end
```

*Tabla F.4.3.2 SIEM Pipeline Allowed USBs Stage 2*

## F.4.4 TOR Enrichment

Stage 1:

```
rule "Check IP against TOR Exit Nodes"
when
    has_field("DestinationAddress")
then
    let result = lookup_value("tor-exit-nodes",to_ip($message.DestinationAddress),"false");
    set_field("isTor", result);
end
```

*Tabla F.4.4.1 SIEM Pipeline TOR Enrichment Stage 1*

Stage 2:

```
rule "Route to access to TOR network and remove from https web traffic"
when contains(to_string($message.isTor),"true")
then
    route_to_stream("access to TOR network");
    remove_from_stream("https web traffic");
end
```

*Tabla F.4.4.2 SIEM Pipeline TOR Enrichment Stage 2*

## F.4.5 Known IPs

Stage 1:

```
rule "Check for DestinationAddress is Private IP"
when
    has_field("DestinationAddress") && // Check if the field exists
    regex("^10\\.|^172\\.1[6-9]\\.|^172\\.2[0-9]\\.|^172\\.3[0-1]\\.|^192\\.168\\.",
to_string($message.DestinationAddress)).matches == true
then
    set_field("privateIP", to_string($message.DestinationAddress)); // Add the new field if condition is met
end
```

*Tabla F.4.5.1 SIEM Pipeline Known IPs Stage 1*

```
rule "Check for SourceAddress is Private IP"
when
    has_field("SourceAddress") && // Check if the field exists
    regex("^10\\.|^172\\.1[6-9]\\.|^172\\.2[0-9]\\.|^172\\.3[0-1]\\.|^192\\.168\\.",
to_string($message.SourceAddress)).matches == true
then
    set_field("privateIP", to_string($message.SourceAddress)); // Add the new field if condition is met
end
```

*Tabla F.4.5.2 SIEM Pipeline Known IPs Stage 1*

Stage 2:

```
rule "Check if privateIP is in KnownIPs CSV"
when
    has_field("privateIP")
then
    let result = lookup_value("known-ips",to_string($message.privateIP));
    set_field("knownIP", result);
end
```

*Tabla F.4.5.3 SIEM Pipeline Known IPs Stage 2*

Stage 3:

```
rule "Route to not known IP"
when contains(to_string($message.knownIP),"false")
then
    route_to_stream("not known IP");
end
```

*Tabla F.4.5.4 SIEM Pipeline Known IPs Stage 3*

# F.5 Configuración adicional EDR

## F.5.1 Ataque SQL Injection

```
<integration>
    <name>shuffle</name>
    <hook_url>https://10.111.3.63:3443/api/v1/hooks/webhook_4189588a-ae52-403b-8ef3-7f3254a3162b</hook_url>
    <rule_id>31103</rule_id>
    <group>sql_injection</group>
    <alert_format>json</alert_format>
  </integration>
```

*Tabla F.5.1.1 EDR Configuración adicional CDU Ataque SQL Injection*

## F.5.2 Rescue-Terminal

```
<command>
    <name>rescueterm</name>
    <executable>rescue_terminal.py</executable>
    <timeout_allowed>yes</timeout_allowed>
  </command>
  <active-response>
    <command>rescueterm</command>
   <location>local</location>
  </active-response>
```

*Tabla F.5.2.1 EDR Configuración adicional CDU Rescue-Terminal*

## F.5.3 Cuarentena completa de Hosts comprometidos

```
<active-response>
    <command>firewall-drop</command>
    <location>local</location>
  </active-response>
```

*Tabla F.5.3.1 EDR Configuración adicional CDU Cuarentena completa de Hosts comprometidos*

# Anexo G

# Resultados experimentales

En este Anexo se encuentra todo el detalle relativo a la experimentación realizada, se recogen tanto los resultados obtenidos como explicaciones de las pruebas realizadas.

El anexo cuenta con 2 subcapítulos: El primero de ellos trata todo lo relacionado a las pruebas de carga realizadas sobre cada uno de los componentes del sistema. El segundo de ellos explica las pruebas exactas realizadas para cada CDU, en estas se exploran todas las posibilidades de los workflows implementados.

## G.1 Pruebas de carga

### G.1.1 SIEM

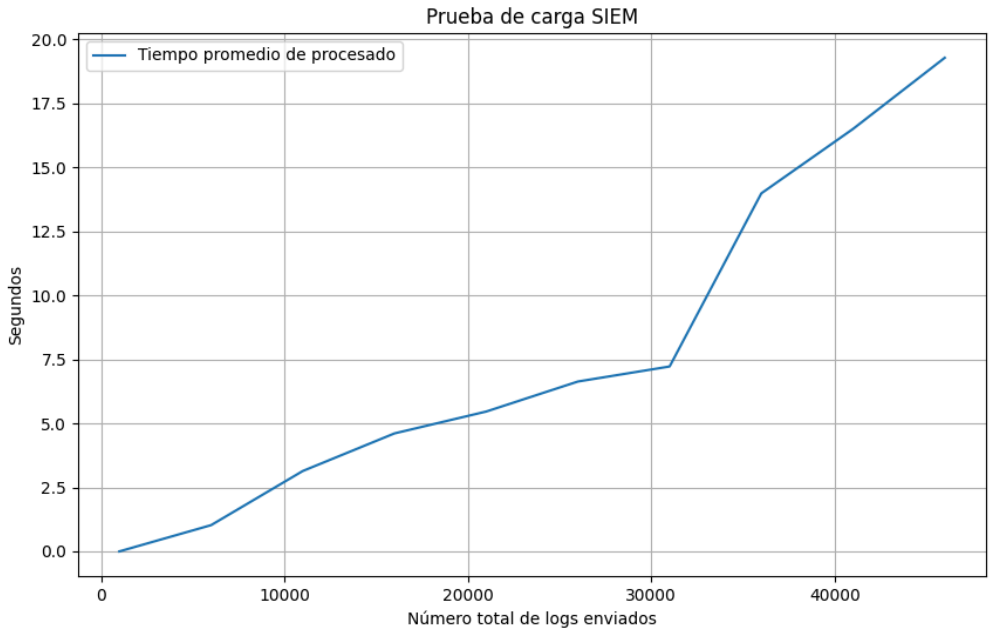


*Figura G.1.1.1 Prueba de carga SIEM*

Por otro lado también se ha probado a saturar completamente mediante el envío masivo de logs al SIEM, con un envío de 100000 logs en un plazo de 2 segundos, la conclusión extraída de dicha prueba es que a pesar de que el tiempo de procesado aumente, el limitante real del SIEM es la red dado que hace de cuello de botella tanto en la máquina desde la que se realiza el envío como en el balanceador de carga, es decir el SIEM soporto todos los logs manteniendo el funcionamiento esperado. Para la prueba de carga se ha empleado un log del cortafuegos, el cual es uno de los más costosos en tiempo de procesado debido a la cantidad de parámetros que debe parsear.

## G.1.2 SOAR

La primera prueba trata de evaluar el incremento del tiempo medio de ejecución de cada workflow, empleando un único worker y conforme el número de workflows lanzados aumenta.

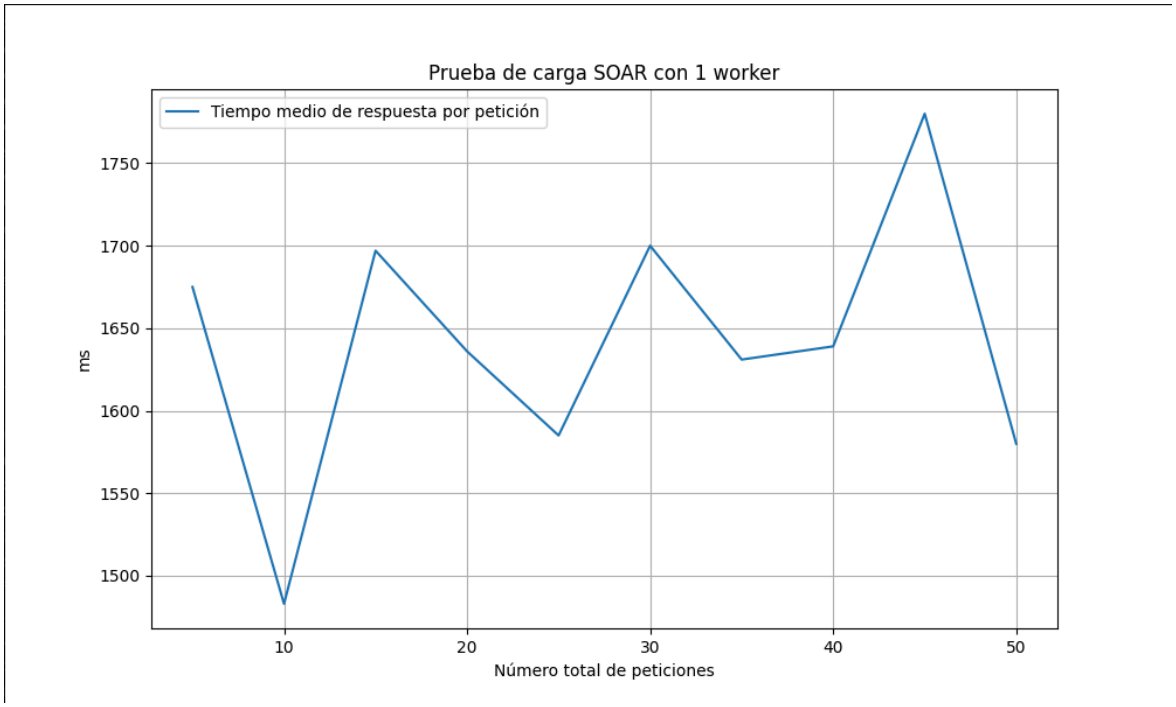


*Figura G.1.2.1 Prueba de carga SOAR*

Como se puede observar en la gráfica, el tiempo medio de ejecución por workflow aumenta conforme lo hace el número de workflows a ejecutar, no obstante se puede observar que este aumento no es muy significativo, dado que a pesar de tener un solo nodo worker el tiempo medio de ejecución por workflow teniendo 50 workflows en ejecución simultánea es de menos de 2 segundos. Si a esto añadimos el dato de que la solución está pensada para escalar de forma lateral este resultado es muy bueno.

Dado que en la documentación de la herramienta no aparece de forma explícita el escalado horizontal mediante el uso de varios workers, era necesario comprobar esta propiedad por lo que se ha realizado la siguiente prueba: Para un número N de workflows a ejecutar, se ha medido el tiempo absoluto de ejecución variando en cada prueba el número W de nodos workers. Los resultados obtenidos son los siguientes:

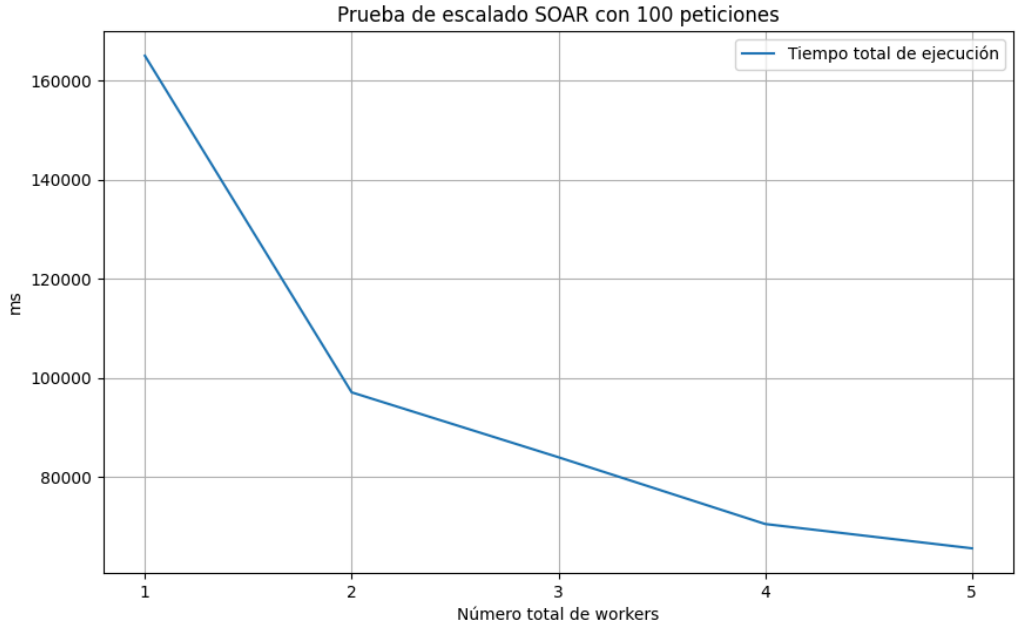


*Figura G.1.2.2 Prueba de escalado SOAR*

Como se comprobó durante las primeras fases de análisis e implementación de la herramienta, esta cuenta con la propiedad de escalado horizontal siendo la diferencia entre emplear un solo nodo worker y dos nodos muy significativa, ya que casi se divide el tiempo total de ejecución a la mitad, si seguimos aumentando la cantidad de nodos se siguen obteniendo ganancias en rendimiento pero no son tan significativas como esta primera.

## G.1.3 EDR

Un factor importante en un EDR que se encarga de monitorizar/defender unos endpoints es conocer cómo aumenta la cantidad de recursos empleados por el servidor en comparación a como aumenta el número de endpoints que cuentan con el agente instalado. Para medir esto se emplearon las siguientes métricas: El porcentaje de RAM empleado por el servidor y el la carga media de trabajo de los últimos 5 minutos, es decir para obtener unas medidas estables una vez instalado un nuevo agente se debía esperar 5 minutos para poder apreciar los cambios en dicha carga media.

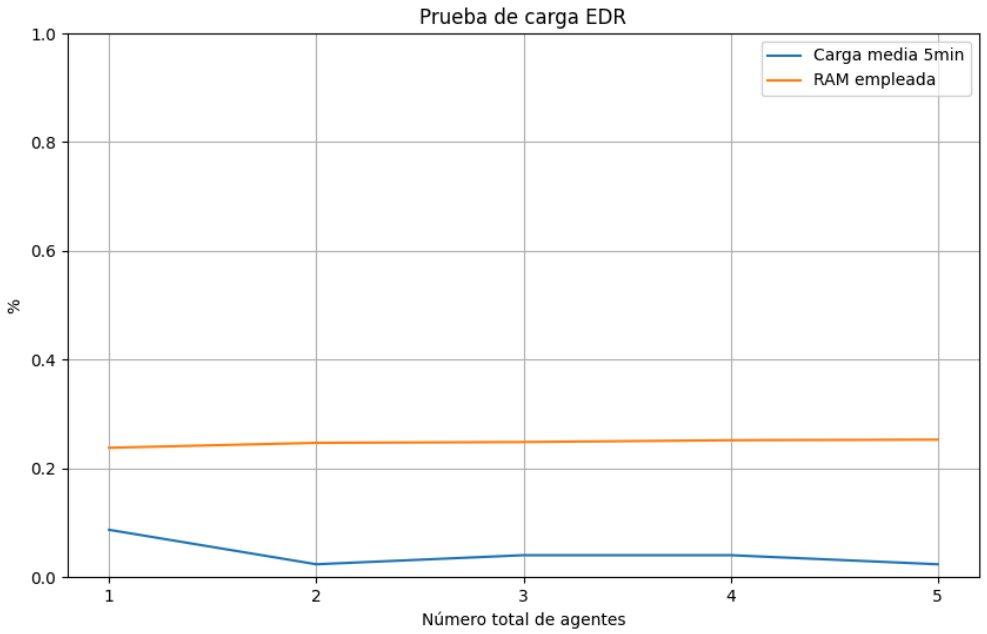


*Figura G.1.3.1 Prueba de carga EDR*

Como cabía esperar el servidor mantiene la cantidad de recursos empleados de forma estable a pesar del aumento de endpoints monitorizados, asimismo cabe destacar que el servidor se encuentra clusterizado como se ha explicado anteriormente por lo que la carga de trabajo se reparte entre todos los nodos por igual.

## G.2 Pruebas funcionamiento CDUs

### G.2.1 Análisis de phishing

Se han probado todos los casos contemplados: Recibir un correo desde una IP maliciosa, recibir un correo cuyo contenido tiene enlaces maliciosos, recibir un correo que trae consigo ficheros adjuntos sospechosos, recibir un correo no malicioso.

### G.2.2 Fuerza bruta a SSH

Se ha probado a atacar los siguientes tipos de cuenta: usuario estándar como admin/root, usuario que sigue la denominación nominal pero no se encuentra en el AD, usuario que sigue la denominación nominal y si existe en el AD. Asimismo también se han probado accesos normales de un usuario que sí existe en el AD.

### G.2.3 Accesos a webs con URLs maliciosas conocidas

Se han realizado pruebas de acceso a webs seguras conocidas como google.es y a webs con mala reputación tales como thepiratebay.org .

### G.2.4 Ataque DDOS

Se ha simulado un ataque DDOS mediante la bajada del Threshold, sobre la cantidad de tráfico de red, que determina cuando se está sufriendo este tipo de ataque.

### G.2.5 Data Leaks mediante USB + Ataque Rubber Ducky

Se ha probado a emplear: un dispositivo USB permitido, un dispositivo USB no autorizado y un USB no autorizado seguido de la ejecución de muchos comandos en un corto intervalo de tiempo.

### G.2.6 Ataque SQL Injection

Se ha empleado un servidor web propio como entorno de pruebas, sobre este se ha ejecutado una SQL Injection de forma manual, así como de forma automática con la herramienta sqlmap. También se ha probado el uso normal de dicho servidor web.

### G.2.7 Accesos a la red TOR

Para comprobar la detección y corte instantáneo de accesos a la red TOR, se han realizado conexiones a esta red de dos formas distintas, la primera de ellas mediante un navegador web normal preparado para acceder a esta red, concretamente Brave, la segunda forma ha sido emplear el propio navegador TOR.

### G.2.8 Análisis y detección de anomalías en red

Dada la naturaleza especial de este CDU, primero se ha realizado una etapa de aprendizaje de la red y posteriormente para simular un ataque de intrusión no autorizada en la red se ha lanzado una MV con una IP autoasignada, que no existía en la fase de aprendizaje.

## G.3 CDUs Manuales

### G.3.1 Rescue-Terminal

Se han probado todos los casos contemplados por el script Python de la bindShell es decir lanzar esta y acceder correctamente, lanzarla e introducir una contraseña errónea y por último lanzar la terminal y no acceder en el tiempo límite de 5 minutos.

### G.3.2 Cuarentena completa Hosts comprometidos

Se ha creado una MV la cual ha sido puesta en cuarentena completa, en esta se han hecho pruebas de conectividad de red, tanto antes como después de aplicar la cuarentena.