Universitat Oberta de Catalunya

uoc.edu

Implementación de SIRP Open Source

Raúl Romero Cabello
Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)
Seguridad Empresarial

Miguel Ángel Flores Terron
Víctor Garcia Font

01/06/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Implementación de SIRP Open Source*
Nombre del autor:	*Raúl Romero Cabello*
Nombre del consultor/a:	*Miguel Ángel Flores Terron*
Nombre del PRA:	*Víctor Garcia Font*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC)*
Área del Trabajo Final:	*Seguridad Empresarial*
Idioma del trabajo:	*Castellano*
Palabras clave	*ELK, Wazuh, Hive-Cortex-MISP*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

La finalidad de este trabajo es disponer de una plataforma SIRP con software de código abierto que pueda ser escalable y altamente disponible con fines pedagógicos en un entorno de simulación pero que dotado de los recursos necesarios hardware necesarios pueda ser implementable para la gestión de incidentes de cualquier empresa u organización que necesite implementar este tipo de soluciones como parte de su SGSI.

A nivel metodológico, en este trabajo se ha seguido los siguientes pasos:

1- Analizar las funcionalidades necesarias y ver que opciones hay disponibles

2- Escoger los componentes y diseñar una solución en base al estudio realizado y al alcance propuesto

3- Implementar e integrar los diferentes componentes/productos que forman la solución

4- Realización de pruebas, conclusiones y documentar posibles mejoras

5- Completar la documentación y presentación sobre el trabajo realizado

Como resultado tenemos una plataforma SIRP operativa que permite la gestión de los incidentes desde su detección a la respuesta, con su correspondiente documentación de posibles incidentes y cierre, que permite compartir información (IoCs) con terceros si se precisa y que utiliza fuentes externas de inteligencia para ayudar a los analistas de seguridad en los que puede ser el día a día de cualquier empresa u organización, todo ello utilizando tecnologías actuales de cloud para su implementación y gestión.

Abstract (in English, 250 words or less):

The purpose of this work is to have a SIRP platform with open source software that can be scalable and highly available for pedagogical purposes in a simulation environment but that equipped with the necessary hardware resources can be implementable for incident management of any company or organization that needs to implement this type of solutions as part of its ISMS.

At the methodological level, this work the following steps have been followed:

1- Analyze the required functionalities and see what options are available
2- Choose the components and design a solution based on the study carried out and the proposed scope
3- Implement and integrate the different components/products that make up the solution
4- Testing, findings and documenting possible improvements
5- Complete the documentation and presentation on the work done

As a result we have an operational SIRP platform that allows the management of incidents from their detection to the response, with its corresponding documentation of possible incidents and closure, which allows to share information (IoCs) with third parties if necessary and that uses external sources of intelligence to help security analysts in what can be the day to day of any company or organization, all using current cloud technologies for its implementation and management.

Índice

Lista de figuras

Tablas

Ilustraciones

1. Introducción

1.1 Contexto y justificación del Trabajo

No cabe duda y más en el contexto actual que vivimos en estos tiempos que todas las empresas y organizaciones están inmersas en un proceso de digitalización a marchas forzadas. En mayor o menor medida dependiendo de su ámbito de negocio están intentando seguir ofreciendo sus servicios o realizar su gestión a través de Internet.

No todas partían desde el mismo punto y cada empresa u organismo tiene diferente grado de madurez en este ámbito, hay empresas que ya estaban inmersas en el proceso y solo han tenido que acelerarlo, con lo que ya tenían parte del trabajo o al menos tenían diseñada la estrategia, pero hay empresas que han tenido que partir de cero y ponerse en una situación que no habían planificado y que en muchos casos depende la continuidad de sus servicios y su negocio.

Dicho esto, una mayor digitalización y por tanto dependencia de la tecnología informática e Internet hacen necesario aumentar las capacidades de las empresas y organizaciones en materia de ciberseguridad, en cuyo caso podemos decir lo mismo que del proceso de digitalización, hay empresas y organismos más preparados que otros.

En cualquier caso, dada la creciente aparición de amenazas y riesgos a los que se exponen las empresas y organizaciones públicas o privadas, debido como ya se ha comentado a la mayor dependencia de la tecnología y de la red de redes, así como la difuminación de lo que se conoce como el perímetro de exposición, la adoptación de infraestructuras y servicios en la nube, el cumplimiento de legislación como GDPR, etc. por comentar algunos motivos. Todas las empresas u organismos independientemente de su tamaño son objetivo de diversos actores ya sean con objetivos maliciosos (ciberdelincuencia, ciberespionaje, insiders, hacktivistas, etc.) o accidentales, todas son susceptibles de sufrir ciberincidentes.

La pregunta no es si sucederá o no, dado que los ataques cibernéticos se están dando constantemente en el mundo, sino cuando sucederá y como de resilientes seremos a ellos (que capacidad tendrá nuestra empresa u organismo de sobreponerse a ello). El concepto de resiliencia es mucho más amplio de lo que trataremos en este proyecto, dado que implica las medidas de continuidad de negocio/servicio que tenga la empresa u organización que no será objetivo de este trabajo y que deberían entrar en funcionamiento en caso de incidentes graves o muy graves. Dentro de las actividades en materia de ciberseguridad que deben llevar a cabo todas las empresas, están los planes de gestión de incidentes de ciberseguridad y una parte esencial es disponer de recursos ya sean propios, subcontratados o mediante la adquisición de servicios profesionales externos que permita detectar, contener y responder a estos ciberincidentes.

En este punto es donde entra una plataforma SIRP (Security Incident Response Platform) o lo que es lo mismo una plataforma que permite gestionar los incidentes de ciberseguridad detectados. Esta plataforma debe ayudar a los analistas de seguridad a detectar las amenazas, contenerlas y responder a ellas de la forma más eficiente posible, por lo que debe dar las herramientas necesarias para que un SOC o un CSIRT puedan realizar su trabajo.

La aportación que realiza este proyecto es seleccionar los componentes y simplificar la creación de esta plataforma ya sea para que una empresa la pueda implementar en sus instalaciones (lo que se conoce como on-premise) o lo pueda hacer en la nube en caso de disponer de personal cualificado para administrar la plataforma, analizar y desarrollar mejoras en la misma o bien para que sea implementado en un servicio que podríamos decir SOC as a Service, es decir, una empresa podría dedicarse a implementar este tipo de plataforma y ofrecer sus servicios a otras empresas proveyendo la plataforma y/o el personal para su gestión.

Otra cosa que deberíamos mencionar es que las redes de cibercriminales o ciberdelincuentes cada vez disponen de más recursos y en muchos casos incluso hay colaboración entre ellos, hoy en día es muy habitual oír hablar de “malware as a service”, es decir, un grupo crea un software malicioso que permite realizar ataques y lo vende o lo alquila a un tercero que será el encargado de utilizarlo contra su objetivo, cada vez hay más especialización. Para combatir esto se requiere que los equipos que trabajan digamos en el lado de los “buenos” también colaboren ya sea con las administraciones públicas (léase CCN-CERT, INCIBE, etc.), así como con otras empresas compartiendo información que posibilite detectar y atajar las amenazas a las que todos estamos expuestos.

1.2 Objetivos del Trabajo

Los principales objetivos para este TFM son los siguientes:

Objetivos a nivel de investigación y estudios:

- Realizar una revisión bibliográfica sobre la detección de amenazas y gestión de incidentes
- Estudiar diversas fuentes que permite la detección de amenazas (IDS/IPS, software EDR, logs, etc.)
- Estudiar un sistema de gestión de información, eventos y logs como puedes ser ELK y como se utiliza desde el punto de vista de seguridad (SIEM)
- Estudiar herramientas de gestión de incidentes, como se integran con las fuentes de datos para enriquecer los casos y como gestionar la respuesta

Objetivos a nivel de implantación y desarrollo:

- Instalar y configurar un entorno de laboratorio para simulación de un escenario de pruebas (PoC) del SIRP, SIEM y añadir elementos que alimenten con eventos e incidentes dichos productos para verificar su funcionamiento e integración
- La solución tiene que ser escalable y debería ser fácilmente instalable, reproducible en un entorno real (automatización del proceso)
- Integrar en la herramienta de solución de incidencias respuesta a los casos simulados de amenazas

Objetivos a nivel académico:

- Generar los entregables parciales de la solución en base a los plazos de entrega establecidos en el TFM
- Desarrollar la memoria final del TFM.
- Elaborar una presentación y vídeo de síntesis sobre el TFM

1.3 Enfoque y método seguido

Como ya se ha comentado en el apartado 1.1 de Introducción, este tipo de plataformas es necesario para cualquier tipo de empresa u organización que quiera proteger sus activos de amenazas internas o externas independientemente de su tamaño.

Podríamos pensar en tres estrategias:

- Desarrollar una plataforma propia
- Recurrir a productos comerciales
- Implementar herramientas open source

La primera opción queda descartada, teniendo en cuenta que debe servir para cualquier tipo de empresa u organización independientemente de su tamaño, el presupuesto o equipo necesario para desarrollar, mantener y administrar una plataforma de este tipo, aunque fuera con los elementos mínimos considero que sería inabordable, ineficiente y como se suele decir, sería reinventar la rueda. No sería factible disponer de una solución operativa en un tiempo razonable.

Recurrir a productos comerciales, es una solución no descartable en general, de hecho las grandes empresas y organismos que disponen de presupuesto suficiente es como abordarían normalmente este problema, existen mucho productos de mercado SIEMs, EDRs y SOARs, podríamos listar algunos fabricantes que suministran todos o algunos de estos componentes, por dar algunos ej. tenemos IBM QRadar, Splunk como plataformas SIEM, en cuanto a EDRs prácticamente cualquier empresa que se dedica al mundo de los antivirus ha reconvertido su negocio hacia los EDRs (Sophos, McAfee, Trend Micro, Panda, etc.), así como algo más novedoso que son las plataformas SOAR como IBM Resilient o PaloAlto Demisto, que permite gestionar los incidentes aunando la información de las otras herramientas y proporcionando respuestas (orquestando respuesta) a los posibles incidentes detectados. ¿porque no seguimos este método para implementar nuestra plataforma

SIRP?, porque el presupuesto no está al alcance de todos, no son productos que se pueda permitir en muchos casos una PIME e incluso para las grandes empresas supone un gran coste en licencias e igualmente por muchas facilidades que proporcionen igualmente requieren de personal cualificado para manejarlas.

Así que recurrimos a los productos open source, cuya desventaja principal es que en caso de tener algún problema este queda en manos de la comunidad y en un entorno empresarial no siempre es una buena solución por tiempo y necesidad de una solución, pero por el contrario considero que es una buena apuesta cuando existe una amplia comunidad que da soporte a un cierto producto, al ser open source su código es abierto lo que facilita también la detección de bugs, al no haber nada oculto en su código. En este ámbito de productos existen soluciones con suficiente madurez para poder utilizar en entornos empresariales con ciertas garantías por sus años de desarrollo, evolución, adaptación a los tiempos y necesidades actuales, así como a la integración entre ellas.

La opción de montar un SIRP con software open source es la que se trabajará en este TFM tal como indica su propio título y para ello tal como podemos ver en el apartado 1.5, se investiga diversos productos que pueden realizar esta función apoyándose unos en otros con diferentes objetivos que se desarrollan durante el proyecto y que quedarán brevemente explicados en el apartado de estado del arte.

La metodología que seguirá el proyecto es la siguiente:

- Análisis de las funcionalidades de los diferentes componentes
- Diseño de la solución
- Implementación e integración de productos
- Fase de pruebas de los diferentes casos de uso
- Completar la documentación y presentación

La idea es empezar por la teoría de los diferentes componentes, elegir los productos y tener un diseño definitivo de la solución. Montar un entorno funcional básico e integrar los diferentes productos y finalmente pasar a la parte final de pruebas. La memoria se deberá ir actualizando a medida que avance el proyecto con las decisiones tomadas y sus motivaciones, así como tenerla disponible en los diferentes entregables, para finalmente concluir y revisar el documento completo.

Se tratará no solo de que el entorno detecte posibles incidentes y los podamos gestionar como casos a través de la herramienta correspondiente, sino que además estas herramientas se puedan instalar de una forma lo más automatizada posible y si es posible gestionar también de forma lo más automática posible la respuesta e intentando evitar las manualidades para que sea un entorno fácilmente reproducible y escalable como solución real.

1.4 Planificación del Trabajo

	Actividad	Categoría	Inicio	Fin	Días
1	**Planificación**	Objetivo			
1.1	Establecer problema a resolver	Tarea	20/02/2021	22/02/2021	3
1.2	Definición de objetivos	Tarea	23/02/2021	24/02/2021	2
1.3	Descripción metodología	Tarea	25/02/2021	26/02/2021	2
1.4	Planificación del Trabajo	Tarea	27/02/2021	02/03/2021	4
1.5	Estado del arte	Tarea	28/02/2021	02/03/2021	3
1.6	Entrega planificación	Hito	02/03/2021	02/03/2021	1
2	**Análisis y diseño solución**	Objetivo			
2.1	Análisis en profundidad de los componentes de detección y respuesta	Tarea	03/03/2021	12/03/2021	10
2.2	Análisis en profundidad de los componentes de recolección, indexación y almacenamiento de información y eventos	Tarea	13/03/2021	21/03/2021	9
2.3	Análisis en profundidad de gestión de incidentes y compartición de información	Tarea	22/03/2021	28/03/2021	7
2.4	Elección final de los elementos que compondrán la solución	Tarea	13/03/2021	28/03/2021	16
2.5	Diseño definitivo de la solución y requisitos	Tarea	29/03/2021	30/03/2021	2
2.6	Entrega documentación análisis	Hito	30/03/2021	30/03/2021	1
3	**Implementación**	Objetivo			
3.1	Instalación de las plataformas que darán soporte a la PoC	Tarea	31/03/2021	04/04/2021	5
3.2	Instalación y configuración productos de detección y respuesta	Tarea	05/04/2021	11/04/2021	7
3.3	Instalación y configuración productos de recolección, indexación y almacenamiento de información y eventos	Tarea	12/04/2021	18/04/2021	7
3.4	Instalación y configuración productos de gestión de incidentes	Tarea	19/04/2021	27/04/2021	9
3.5	Productos instalados	Hito	27/04/2021	27/04/2021	1
4	**Integración**	Objetivo			
4.1	Configuración de la integración entre los diferentes productos	Tarea	28/04/2021	09/05/2021	12
4.2	Procedimentación de instalación automatizada	Tarea	10/05/2021	16/05/2021	7
4.3	Plan de pruebas	Tarea	17/05/2021	18/05/2021	2
4.4	Documentar en la memoria	Hito	28/04/2021	18/05/2021	21
5	**Testing**	Objetivo			
5.1	Ejecutar pruebas unitarias	Tarea	19/05/2021	20/05/2021	2
5.2	Ejecutar pruebas de integración	Tarea	21/05/2021	22/05/2021	2

5.3	Generar eventos y simular amenazas	Tarea	23/05/2021	24/05/2021	2
5.4	Pruebas de gestión de incidentes	Tarea	25/05/2021	26/05/2021	2
5.5	Pruebas de respuestas automatizadas	Tarea	27/05/2021	28/05/2021	2
5.6	Simulación completa	Hito	29/05/2021	29/05/2021	1
6	**Presentación**	Objetivo			
6.1	Finalizar y revisar memoria	Tarea	24/05/2021	01/06/2021	9
6.2	Elaborar conclusiones y resumen	Tarea	29/05/2021	01/06/2021	4
6.3	Entrega documento memoria	Hito	01/06/2021	01/06/2021	1
6.4	Elaborar presentación y video	Tarea	02/06/2021	08/06/2021	7
6.5	Presentación y video sintexis	Hito	08/06/2021	08/06/2021	1

Tabla 1 - Planificación temporal del proyecto

	Actividad	Categor	Inicio	Fin	Día	Proc
1	**Planificación**	**Objetivo**	**20/02/2021**	**02/03/2021**		100%
1.1	Establecer problema a resolver	Tarea	20/02/2021	22/02/2021	3	100%
1.2	Definición de objetivos	Tarea	23/02/2021	24/02/2021	2	100%
1.3	Descripción metodología	Tarea	25/02/2021	26/02/2021	2	100%
1.4	Planificación del Trabajo	Tarea	27/02/2021	02/03/2021	4	100%
1.5	Estado del arte	Tarea	28/02/2021	02/03/2021	3	100%
1.6	Entrega planificación	Hito	02/03/2021	02/03/2021	1	100%
2	**Análisis y diseño solución**	**Objetivo**	**03/03/2021**	**30/03/2021**		100%
2.1	Análisis en profundidad de los compon	Tarea	03/03/2021	12/03/2021	10	100%
2.2	Análisis en profundidad de los compon	Tarea	13/03/2021	21/03/2021	9	100%
2.3	Análisis en profundidad de gestión de in	Tarea	22/03/2021	28/03/2021	7	100%
2.4	Elección final de los elementos que con	Tarea	13/03/2021	28/03/2021	16	100%
2.5	Diseño definitivo de la solución y requis	Tarea	29/03/2021	30/03/2021	2	100%
2.6	Entrega documentación análisis	Hito	30/03/2021	30/03/2021	1	100%
3	**Implementación**	**Objetivo**	**31/03/2021**	**27/04/2021**		100%
3.1	Instalación de las plataformas que dará	Tarea	31/03/2021	04/04/2021	5	100%
3.2	Instalación y configuración productos d	Tarea	05/04/2021	11/04/2021	7	100%
3.3	Instalación y configuración productos d	Tarea	12/04/2021	18/04/2021	7	100%
3.4	Instalación y configuración productos d	Tarea	19/04/2021	27/04/2021	9	100%
3.5	Productos instalados	Hito	27/04/2021	27/04/2021	1	100%
4	**Integración**	**Objetivo**	**28/04/2021**	**18/05/2021**		100%
4.1	Configuración de la integración entre lo	Tarea	28/04/2021	09/05/2021	12	100%
4.2	Procedimentación de instalación autom	Tarea	10/05/2021	16/05/2021	7	100%
4.3	Plan de pruebas	Tarea	17/05/2021	18/05/2021	2	100%
4.4	Documentar en la memoria	Hito	28/04/2021	18/05/2021	21	100%
5	**Testing**	**Objetivo**	**19/05/2021**	**29/05/2021**		100%
5.1	Ejecutar pruebas unitarias	Tarea	19/05/2021	20/05/2021	2	100%
5.2	Ejecutar pruebas de integración	Tarea	21/05/2021	22/05/2021	2	100%
5.3	Generar eventos y simular amenazas	Tarea	23/05/2021	24/05/2021	2	100%
5.4	Pruebas de gestión de incidentes	Tarea	25/05/2021	26/05/2021	2	100%
5.5	Pruebas de respuestas automátizadas	Tarea	27/05/2021	28/05/2021	2	100%
5.6	Simulación completa	Hito	29/05/2021	29/05/2021	1	100%
6	**Presentación**	**Objetivo**	**24/05/2021**	**08/06/2021**		50%
6.1	Finalizar y revisar memoria	Tarea	24/05/2021	01/06/2021	9	100%
6.2	Elaborar conclusiones y resumen	Tarea	29/05/2021	01/06/2021	4	100%
6.3	Entrega documento memoria	Hito	01/06/2021	01/06/2021	1	100%
6.4	Elaborar presentación y video	Tarea	02/06/2021	08/06/2021	7	0%
6.5	Presentación y video sintexis	Hito	08/06/2021	08/06/2021	1	0%

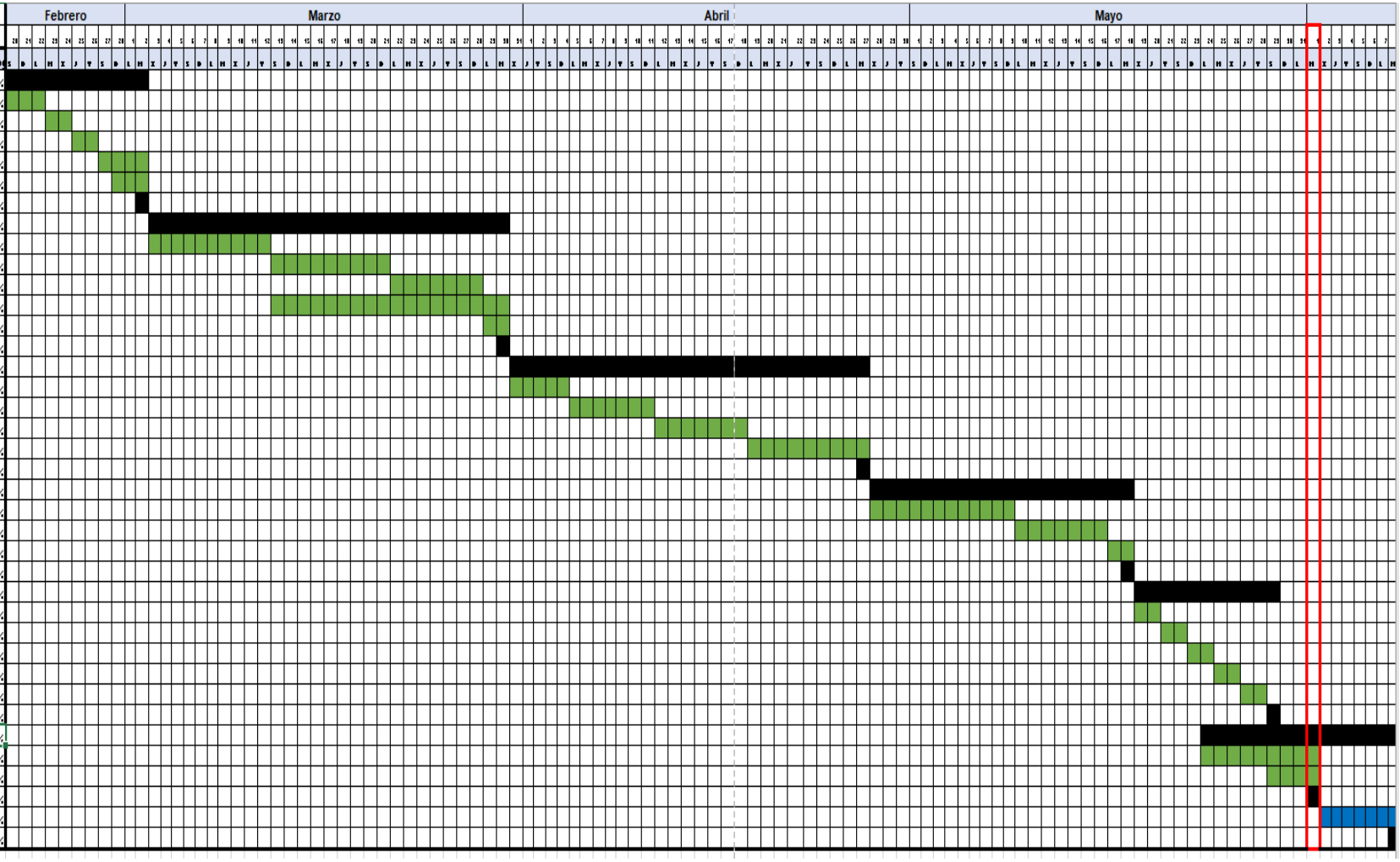


Tabla 2 - Diagrama de Gantt

1.5 Breve sumario de productos obtenidos

Para poder disponer de una plataforma de gestión de incidentes operativa necesitamos disponer de elementos de detección de amenazas o posibles ataques, debemos disponer de un espacio donde almacenar los eventos generados por las herramientas de detección donde poder correlacionar, analizar y buscar patrones, un lugar donde poder observar la información de las diferentes fuentes de eventos a monitorizar, necesitamos poder generar alertas en base a esa información y gestionar de forma centralizada esos posibles incidentes, de forma que todos los participantes, ya sean técnicos de TI, analistas de seguridad o responsables puedan compartir información y enriquecer un caso abierto, así como poder compartir esa información entre todas las partes interesadas en el proceso.

Para poder realizar todas estas actividades se han seleccionado una serie de productos open source que permitan aportar soluciones en las diferentes etapas de la gestión de un incidente y que podemos integrar en nuestra solución SIRP, podemos ver un breve análisis básico inicial de productos, componentes y funcionalidades que pueden ser candidatos a formar parte de la solución a implementar.

- Detección y respuesta

WAZUH [1] entre otras funcionalidades es lo que se conoce como un HIDS, tiene la parte servidor que se integra con ELK (del que se hablará más adelante) y un agente que se instala en los equipos a proteger tanto para Windows, como Linux. Se integrará en la solución para formar parte del sistema de detección.

IPS es un sistema que permite proteger a los equipos a nivel de red, se trata de un NIDS que por tanto deberían permitir detectar patrones a ataque a nivel de red y por tanto formaría parte de la estrategia de detección (que no puede cubrir WAZUH, dado que existen elementos de red en los cuales no se puede disponer de un agente). También formaría parte de nuestro sistema de respuesta, dado que el IPS tiene capacidad de responder bloqueando ciertos tipos de ataques. Se trata de utilizarlo como otra fuente de ingestión de información de seguridad en nuestro ELK.

Los dos productos open source más conocidos que cubren la funcionalidad mencionada son **SNORT** que es un proyecto con una larga trayectoria y **Suricata** que es bastante más moderno, pero que cada día está obteniendo más soporte de la comunidad.

Firewall, es un elemento tradicional de seguridad y que formaría parte de los mínimos exigidos en cualquier instalación de una empresa u organización. Puede aportar algunas capacidades en cuanto a detección a nivel de red, en nuestro caso utilizaremos uno de capa 4 del modelo OSI. También puede formar parte del sistema de respuesta en cierto tipo de incidentes.

Actualmente disponemos de **iptables** y **firewalld** disponibles en la mayoría de distribuciones Linux, el primero es bastante antiguo aunque todavía hay versiones que lo utilizan y el segundo es más moderno y es por el que se han decantado las últimas versiones de las distribuciones más comerciales (RHEL, SUSE, etc.), son firewalls de tipo stateful, es decir, que analizan los paquetes para aplicarles reglas teniendo en cuenta el estado de las conexiones.

Tenemos otro firewall open source bastante conocido que es **Pfsense** que también está basado en código abierto (FreeBSD), es una solución que viene preparada (personalizada) para realizar la función de firewall y enrutador, que permite la adición de otros componentes de seguridad como por ejemplo SNORT en el mismo equipo.

WAF se trata de un componente de detección y respuesta, es un firewall de capa 7 OSI, lo que se conoce como firewall aplicativo y nos sirve para proteger aplicaciones web de cierto tipo de amenazas (como por ejemplo el Top 10 OWASP). Podemos utilizar el **mod_security** que es un software de código abierto que permite integrarse con los más conocidos y utilizados servidores web Apache y NGINX.

Proxy es un elemento que permite enriquecer la información vía SIEM de actividad de los usuarios en su navegación y puede ser también un punto donde aplicar respuesta si por ej. tenemos un IOC que sea una URL.

Sysmon [2] es un servicio gratuito del sistema de Windows (que está dentro de la suite de herramientas Sysinternals), y un controlador de dispositivo que, una vez instalado en un sistema, permanece residente durante los reinicios del sistema para monitorear y registrar la actividad del sistema en el registro de eventos de Windows. Proporciona información detallada sobre las creaciones de procesos, las conexiones de red y los cambios en el tiempo de creación del archivo. Al recopilar los eventos que genera utilizando Windows Event Collection o agentes SIEM y analizarlos posteriormente, puede identificar la actividad maliciosa o anómala y comprender cómo los intrusos y el malware operan en su red (aporta información de eventos, pero se requiere de definición de alertas en el SIEM para qué sirva como sistema de detección).

- Recolección, indexación y almacenamiento de información y eventos

ELK es la base de nuestro SIEM, se trata de tres productos de código abierto que suelen trabajar de forma conjunta el Elastic Stack por eso se les conoce por sus siglas unificadas. Hablamos de **Elasticsearch, Logstash y Kibana**. Es una plataforma que permite ganar en observabilidad de todo lo que sucede en nuestro entorno, enriqueciéndolo con información de diversas fuentes y así mejorar la seguridad.

Se describe brevemente la funcionalidad de cada uno de los componentes:

Logstash [3] tal como se describe en la web de Elastic, ingesta, transforma y envía de forma dinámica (en tiempo real) de datos independientemente de su

formato o complejidad. Deriva estructura a partir de datos no estructurados con grok, descifra las coordenadas geográficas de las direcciones IP, anonimiza o excluye los campos sensibles y facilita el procesamiento general. Se puede almacenar la información en otros repositorios, pero en nuestra solución se utilizará Elasticsearch que forma parte de la misma solución como y se ha comentado.

Elasticsearch [4] es un motor de búsqueda y analítica de RESTful (mediante API estándar REST) distribuido capaz de abordar un número creciente de casos de uso (en nuestro caso utilizado como SIEM). Como núcleo del Elastic Stack, almacena de forma central los datos para poder realizar búsquedas.

Kibana [5] es una interfaz de usuario de código abierto que permite visualizar los datos almacenados por Elasticsearch. Existen plantillas para poder configurar dashboards predeterminados de búsqueda (por ej. la integración con Wazuh).

Filebeat [6] es un agente que monitoriza los ficheros de log y eventos que se configura en un equipo para reenviarlos a Elasticsearch o Logstash para su indexación.

Packetbeat [7] es un elemento que también forma parte de lo que llaman Beats de ELK y que permite monitorizar la red, es un analizador de paquetes que envía información de un host o un contenedor a Logstash o Elasticsearch.

Aquí se expone un breve resumen de su funcionalidad, pero volveremos a tratarlo de forma más extensa en los siguientes capítulos de esta memoria.

Elastalert [8] es un marco simple para alertar sobre anomalías, picos u otros patrones de interés de los datos en Elasticsearch. Funciona combinando Elasticsearch con dos tipos de componentes, tipos de reglas y alertas. Elasticsearch se consulta periódicamente y los datos se pasan al tipo de regla, que determina cuándo se encuentra una coincidencia. Es componente trabaja con Elasticsearch pero no forma parte del Elastic Stack. También se menciona **Praeco** para poder implementar las alertas de forma gráfica, esta herramienta trabaja con Elastalert.

- Gestión de incidentes y compartición de información

TheHive [8] es un componente básico para la plataforma SIRP, se define a si mismo como una solución escalable, de código abierto diseñada para la gestión de incidentes de seguridad desde su apertura al cierre. Aunque hay que decir, que si solo dispusiéramos de este componente todo sería manual y de hecho no tendríamos capacidades de detección o respuesta, sería una solución incompleta.

Cortex [8] es otro producto de software del mismo equipo que TheHive y que lo complementa con el enriquecimiento de datos. Cortex permite el uso de "analizadores" para obtener información adicional sobre los indicadores que ya

están presentes en sus registros. Permite la consulta de servicios de terceros sobre indicadores como IP, URL y hash de archivo, y etiquetará la alerta con esta información adicional. Permite mantener de forma centralizada el uso de esos analizadores que de otro modo requeriría que cada analista que participa en un incidente de seguridad realizara de forma manual, manejando en algunos casos credenciales, etc. Es un componente que no solo añade información si la hay a un caso, sino que eficiente la gestión de un incidente.

MISP [8] es una plataforma de intercambio de amenazas de código abierto mantenida por CIRCL que, entre muchos otros usos, permite al operador suscribirse a fuentes de inteligencia de amenazas. Estos feeds pueden ser suscripciones de pago o feeds mantenidos por la comunidad de varias organizaciones (como pueden ser otras empresas u organismos oficiales como por ej. el INCIBE). Permite compartir lo que se denomina IoCs de forma que se pueda prevenir o dar respuesta a un incidente creando las reglas de detección o bloqueo en base a los patrones.

- Elementos opcionales

VPN este elemento depende de la implementación, si estamos realizando un montaje en una red local, no es necesario implementar este componente en la solución, si se trata de una empresa u organización con varias sedes es posible que ya tenga implementada una solución de este tipo. En el caso de una implementación en la nube ya sea como IaaS, PaaS o un SaaS podría ser una opción necesaria si el proveedor de cloud no suministra una solución de este tipo para proporcionar una conectividad segura a través de Intenet. A nivel de software libre tenemos la posibilidad de utilizar **OpenVPN** (una VPN basada en túneles SSL, no permite IPSec) que ya he utilizado en algún proyecto y también tenemos **OpenSWAN** (un proyecto de software libre qué si implementa IPSec y que se lleva utilizando desde el año 2005, recientemente han publicado una nueva versión).

PKI su uso sería necesario para garantizar la confidencialidad de los canales de comunicación entre componentes, así como la autenticidad de los componentes que forman parte de la solución. Aquí podemos utilizar **openssl** que es una solución de código abierto que está incluida en todas las distribuciones Linux. Existen otras opciones como utilizar certificados autofirmados, que garantizan el cifrado del canal, pero al no usar una CA confiable no podrían tener autenticidad de los componentes, también existe la opción de utilizar una PKI externa gratuita como puede ser **Let's Encrypt**.

IntelMQ [9] es una solución para recopilar y procesar fuentes de seguridad (como archivos de registro) mediante un protocolo de cola de mensajes. Es una iniciativa impulsada por la comunidad llamada IHAP (Proyecto de automatización de manejo de incidentes) que fue diseñada conceptualmente por CERT / CSIRT europeos durante varios eventos de InfoSec. Su objetivo principal es brindar a los respondedores de incidentes una manera fácil de recopilar y procesar inteligencia sobre amenazas, mejorando así los procesos de manejo de incidentes. Este producto al igual que MISP permite obtener

feeds de fuentes externas, el IntelMQ permite integrar también los feeds procedentes de MISP.

VulnScan [10] este componente permite realizar scans de vulnerabilidades sobre los equipos que se quieren proteger existen varias opciones que se pueden utilizar de forma gratuita (Nessus Essencials), OpenVAS, etc. Está incluido en la lista de componentes opcionales dado que en una plataforma SIRP no sería un elemento indispensable, aunque si tiene una función clara de soporte y puede ayudar en un caso de incidente a determinar el tipo de respuesta (ej. si se detecta un ataque a una vulnerabilidad contra un equipo concreto que sabemos que no tiene dicha vulnerabilidad, esta información se puede correlacionar).

2. Análisis y diseño de solución

2.1 Análisis de los componentes de detección y respuesta

En la metodología de gestión de incidentes existen varias fases que se tienen que abordar, es un proceso de mejora continua que empieza por la fase de preparación y prevención en la cual se deben realizar todas las acciones posibles para poder estar en disposición de responder ante un incidente (equipo de CSIRT, backups, herramientas gestión de incidencias, análisis de riesgos, herramientas de detección y monitorización, etc.), para no extender demasiado este apartado nos centraremos en prepararnos para la detección y empezaremos por las herramientas necesarias, estás pueden variar en función de los assets a proteger, de las organizaciones, presupuestos, etc., pero hoy en día hay ciertas herramientas que se pueden considerar básicas en este ámbito y en cuanto a presupuesto siempre se puede recurrir a la colaboración con la comunidad opensource.

Los S.O. y middleware en general suelen tener su apartado reservado para la seguridad y disponen de logs donde se registran las actividades relacionadas con accesos, ejecución de comandos, etc. aunque forma parte de la bitácora necesaria para poder detectar un incidente de seguridad es insuficiente, existen múltiples eventos que escapan al control previsto en los S.O. y middleware, así como en las aplicaciones que utilizan las organizaciones.

Hoy en día es necesario tener otras herramientas que trabajen en los equipos clientes y servidores para detectar usos de los sistemas que vayan contra la política de seguridad de la organización. Se necesitan herramientas que puedan detectar el malware y bloquear su actividad, si es posible incluso eliminarlos sería deseable y protegernos de otro tipo de ataques que se puedan realizar sobre los sistemas que existen en una organización y por eso necesitamos herramientas de detección y respuesta, lo que se conoce como EDR (Endpoint Detection and Response).

Los EDR son la unión de varias funcionalidades de seguridad y depende mucho del fabricante las funciones que este nos ofrezca, normalmente suelen incluir un antivirus/antimalware más o menos simple, basado en firmas (hoy en día bastante obsoleto) o basado en comportamiento e inteligencia de amenazas, en algunos casos combinación de ambas, algunas incluyen un HIDS como es el caso de Wazuh, etc.

Otro elemento de detección y respuesta que nos puede interesar tener en nuestra organización es un NIDS o NIPS, en el primer caso para detección y alerta en caso de detectar alguna amenaza y en el segundo caso con capacidades de bloqueo de esta amenaza detectada.

Las soluciones de detección a nivel de red, son soluciones que se complementan con las anteriores a nivel de equipo, los EDR trabajan a nivel de Host y pueden detectar comportamientos que no pueden verse a nivel de red (bien sea porque las comunicaciones están cifradas, cosa bastante habitual hoy día, aunque existen soluciones para obviar este inconveniente) o por no

poder monitorizar todos los segmentos de red de una organización, es lógico que no se pueda recoger información de todas las comunicaciones entre equipos de una empresa u organización según su tamaño y segmentación de la misma red, por el volumen de tráfico, etc.

También existen equipos “cerrados”, lo que se suele conocer como Appliances que no permiten instalar software de terceros bien por imposibilidad técnica (a veces tienen un SO propio) o bien por imposibilidad administrativa o de licencia, se puede perder el soporte del fabricante.

Para esta función se han analizado dos de los productos open source más conocidos Snort y Suricata.

2.1.1 Wazuh

Como ya se ha comentado anteriormente Wazuh es una solución open source que nos permite monitorizar nuestros dispositivos para detectar amenazas, la integridad de nuestros sistemas y dar respuesta a incidentes entre otras funciones (como puede ser el “compliance” poder verificar que los equipos cumplen con la política de seguridad de la organización).

¿Por qué Wazuh?

Aunque pueden existir otras opciones, es una suite que integra múltiples funcionalidades que de otra manera sería necesario implementar con diferentes componentes de software y que por tanto no tendríamos la integración que podemos tener con este producto. Es una solución con suficiente soporte de la comunidad (hay muchas soluciones que aparecen y desaparecen en poco tiempo) y con la suficiente madurez como para implementarse en una organización, lo cual se consigue a base de ir publicando nuevas versiones solucionando bugs y añadiendo funcionalidades al producto (en el momento de realizar este estudio del producto van por la versión 4.1.4), lleva funcionando desde 2015, es un fork de otro proyecto anterior llamado OSSEC.

Aparte de la comunidad también existe una empresa que puede dar soporte sobre el producto, punto importante en muchas organizaciones sin el cual no se lanzarían a implementar esta solución, la empresa que da soporte tiene el mismo nombre que el producto Wazuh Inc. Disponen de dos servicios de soporte:

- Standard: 8x5 en horario de oficina (respuesta máxima en 8 horas)
- Premium: 24x7 (respuesta máxima en 4 horas)

También tienen la opción de ser consumido en modo SaaS contratando este servicio a Wazuh Inc.

Es un producto escalable por tanto permite implementarse en una organización independientemente de su tamaño mediante una estructura cliente/servidor, los equipos a proteger tendrían la parte cliente en este caso (Wazuh Agent) y

podemos tener uno o varios servidores (Wazuh Server), pueden crecer en horizontal para recibir información de los agentes, así como para su configuración.

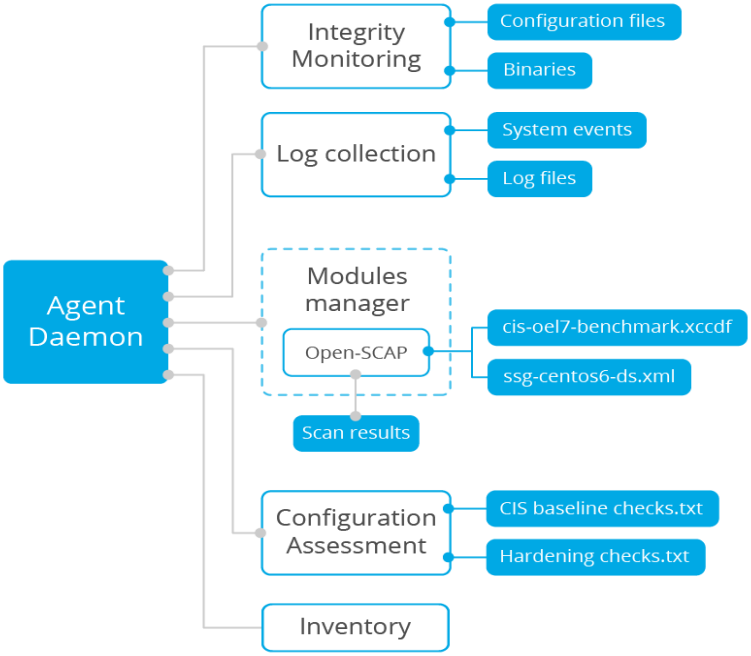


Ilustración 1 - Capacidades agente Wazuh

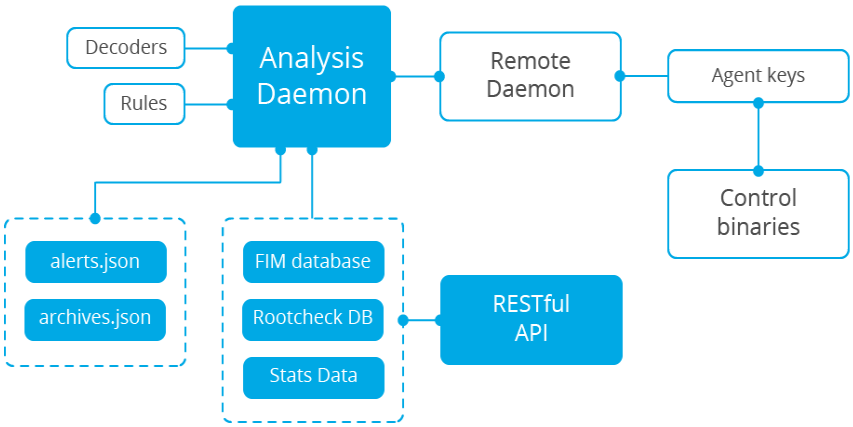


Ilustración 2 - Capacidades Wazuh Server

Es multiplataforma lo cual también es importante dado que las organizaciones pueden ser muy heterogéneas en ese sentido, si bien la plataforma de usuario predominante suele ser Windows, también hay quien opta por Mac o Linux, en la parte de servidores aún se abre más el abanico de posibilidades (aunque hoy en día principalmente tenemos Linux y Windows), podemos tener nuestro Wazuh Agent tanto en Windows, Linux, Mac OS, AIX, Solaris y HP-UX.

Tiene múltiples opciones de implementación en soluciones tipo IaaS o CaaS, en el primer caso dispone de múltiples opciones de instalación en un único servidor con una OVA, tenemos el SO y el software preinstalado, es una

solución rápida para poder trabajar inmediatamente con el producto, aunque no recomendado para una organización mediana/grande por temas de performance, se dispone también de instrucciones para la instalación de sus componentes de forma distribuida y con automatismos para puppet y ansible.

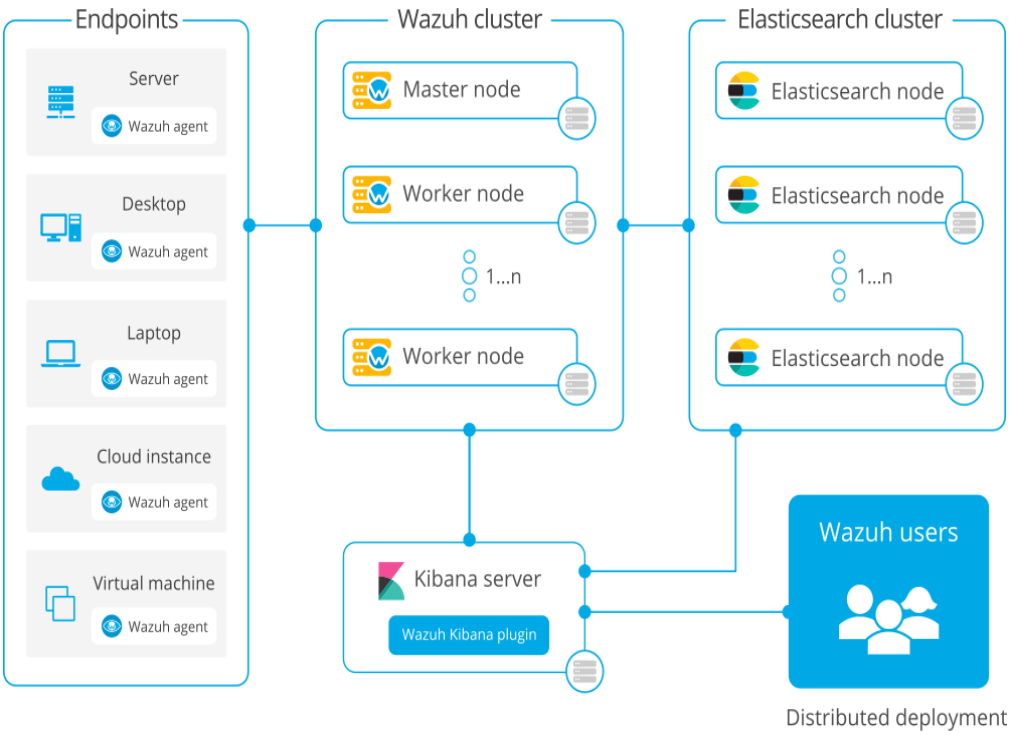


Ilustración 3 - Instalación distribuida Wazuh

Respecto a su implementación en CaaS podemos hacerlo vía Docker y Kubernetes por lo tanto es una solución muy flexible y que se adapta muy bien al panorama de las implantaciones de sistemas actuales tanto on-premise dentro de la organización, en clouds públicos, privados o híbridos.

Otra opción por la que es atractiva esta solución es por su integración con Kibana un componente del Elastic Stack que más adelante hablaremos de él, lo cual nos permite visualizar la información generada por el producto sin tener que dedicarle un tiempo adicional en este punto.

Por último, es una solución que nos tiene que ayudar para proporcionar seguridad en una organización, pero a su vez es una solución que implementa las funciones de seguridad en su funcionamiento que normalmente puede requerir una organización, comunicaciones seguras, autenticación, etc.

Capacidades de Wazuh

A continuación, veremos un pequeño resumen de las principales funcionalidades de Wazuh.

Los eventos que generan los Wazuh Agent son enviados al Wazuh Server donde se pueden definir alertas como alertas en función de su severidad, por defecto a partir de nivel 3 saltaría una alerta.

Los agentes también tienen la capacidad de realizar respuestas activas, el producto viene con algunos scripts para realizar ciertas tareas de respuesta como puede ser bloquear un acceso con el firewall local o crear una ruta nula.

Existe la posibilidad de realizar tareas con Wazuh en equipos sin agente mediante la conexión SSH como pueden ser routers, firewalls o switches.

Las reglas de detección que dispone el producto se gestionan centralizadamente desde un repositorio de Github y se debe configurar para que se actualicen por si aparecen reglas nuevas o también se pueden añadir reglas propias para mejorar las capacidades de detección.

El producto puede integrarse mediante API con otros sistemas:

- Slack: puede crear mensajes en slack para publicar alertas
- PagerDuty: es un servicio SaaS para incident response donde se puede crear un servicio para publicar alertas en su Incident Dashboard
- VirusTotal: se puede integrar para permitir la inspección de ficheros maliciosos usando la BBDD de VirusTotal
- Custom Integration: se utiliza para integrarse con productos de terceros, es una opción genérica que permite mediante scripts la integración

Es posible acceder a Wazuh Server vía web con un navegador o bien interaccionar con él mediante la Wazuh API, una RESTful API que permite gestionar el Wazuh, así como realizar diversas consultas.

Integración con Kibana, como ya se ha comentado es una de las funcionalidades que tiene este producto, dispone de un plugin que permite a los usuarios ver y analizar las alertas de Wazuh almacenadas en Elasticsearch. Los usuarios pueden obtener estadísticas por agente, buscar alertas y filtrarlas utilizando diferentes visualizaciones. Se integra con la API de Wazuh para recuperar información sobre la gestión y configuración de los agentes, logs, reglas definidas, grupos, etc.

Otras funcionalidades del producto son el scan de vulnerabilidades en base a las BBDD de CVEs y a la detección del software instalado, el Wazuh también hace inventario, así como el scan de compliance de políticas y configuraciones de seguridad del equipo.

2.1.2 Snort

Es un software open source dedicado a la detección de amenazas, puede actuar como un NIDS o como un NIPS.

Utiliza una serie de reglas que definen la actividad maliciosa en la red y usa estas reglas ver si hacen match con los paquetes que de red que analiza.

Estas reglas aparte de construir las propias si es necesario pueden obtener de las contribuciones de la comunidad o bien pagar el soporte de subscripción, detrás de esta subscripción esta Cisco Talos quien desarrolla y testea las reglas.

La última versión es Snort 3.1 disponible para Linux, Windows y FreeBSD.

Capacidades de SNORT

Como ya se ha indicado, el producto trabaja analizando los paquetes de red y revisando si hacen match con las reglas definidas.

- Puede realizar la función de NIDS o NIPS, como es habitual en el primer caso se trata de estar fuera de línea y que algún elemento de red (switch o router) envíe le tráfico al puerto de red donde está conectado el equipo con Snort instalado.
- Se pueden definir reglas nuevas de forma sencilla para aumentar las capacidades de detección.
- Permite el reemsamblado de streams y paquetes IP fragmentados en función del destino para evitar ataques de evasión
- Puede analizar varios paquetes de forma simultánea (nuevo en la versión 3).

2.1.2 Suricata

Suricata es un producto open source que implementa un motor para la detección de amenazas en la red. Este motor tiene capacidad para realizar la función de NIDS o NIPS (inline) en tiempo real.

Se centra en la seguridad, usabilidad y eficiencia. Y trabaja con entradas y salidas en formatos muy utilizados como son YAML y JSON de forma que se puede integrar con SIEMs como por ejemplo ELK.

Está disponible para Windows, Linux, BSD y MacOS y actualmente (mientras se redacta este documento) la última versión estable disponible es la 6.0.2

El software está respaldado por OISF (Open Information Security Foundation), una organización sin ánimo de lucro dedicada a fomentar la comunidad y el soporte a tecnologías de seguridad como Suricata.

Capacidades de Suricata

El producto funciona mediante la construcción de reglas y firmas usando un lenguaje que permite modelar incluso amenazas complejas.

- Motor: NIDS, NIPS y NSM, análisis offline de ficheros PCAP (captura de paquetes de red), grava el tráfico usando el PCAP logger y se integra con Linux Netfilter (firewall nátivos de Linux).
- Soporta IPv4 y IPv6 y decodifica algunos protocolos de tunneling, dispone de un motor de flujos (Flow Engine) y su motor TCP permite tracear sesiones y reensamblar flujos de comunicaciones, así como paquetes IP fragmentados en función del destino (para evitar las técnicas de evasión habituales en la detección de NIDS y NIPS).
- Parsea protocolos de diferentes capas: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, Ethernet, PPP, PPPoE, Raw, SLL, VLAN, QINQ, MPLS, ERSPAN, VXLAN, Geneve.
- Decodifica protocolos de capa de aplicación como: HTTP, HTTP/2, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP/CIP, DNP3, NFS, NTP, DHCP, TFTP, KRB5, IKEv2, SIP, SNMP, RDP, RFB, MQTT y se van añadiendo nuevos desarrollados con lenguaje Rust.
- En cuanto a las salidas permite diferentes tipos de logs y formatos.
- Es un software multi-thread
- Filtrado de alertas y eventos
- Utiliza fuentes de reputación IP
- Gestiona datasets para ayudar en la detección (IoCs, DNSs, URI, etc.)
- Incorpora herramientas para la actualización de reglas (Suricata-Update) y para verificación de las mismas desarrolladas (Suricata-Verify).

2.2 Análisis de componentes recolección y almacenamiento de eventos

Para recolectar eventos, procesarlos, almacenarlos y poder realizar consultas dispondremos del Elastic Stack también conocido como ELK por las siglas de sus componentes Elasticsearch, LogStash y Kibana.

Esta suite de herramientas es la que dará soporte a nuestro SIEM y necesitaremos los agentes que proporciona esta solución open source para enviar logs y tráfico de red a nuestro repositorio ELK, lo que se conoce como “beats”.

No analizaremos todos los “beats” que tiene Elastic, solo aquellos que nos pueden servir para nuestra plataforma SIRP.

Elasticsearch también es el sistema de almacenamiento utilizado por el producto TheHive que trataremos más adelante en profundidad en este mismo documento.

ELK es una plataforma open source y por lo tanto podemos hacer nuestra instalación, pero hay una empresa que se encarga de mantener este software y que vende sus servicios profesionales de soporte o bien el consumo de dicho software en modalidad SaaS.

2.2.1 Filebeat

Es un agente ligero que permite reenviar y centralizar datos de logs. Monitoriza los ficheros configurados y permite enviarlos a Elastichsearch o Logstash para su indexación.

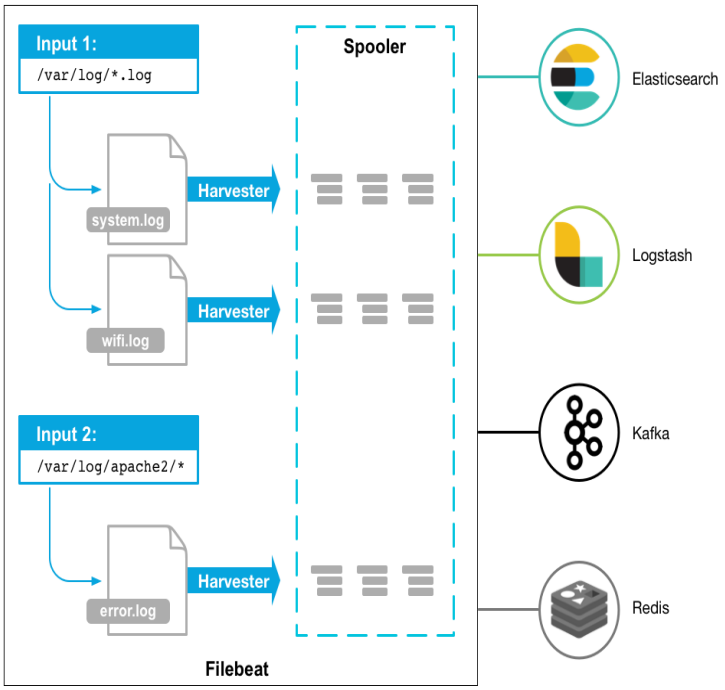


Ilustración 4 - Monitorización y envío de logs

Para recolectar estos logs se ha de instalar este agente en cada uno de los equipos que se quiere monitorizar y enviar logs a nuestro SIEM y configurar los ficheros de los cuales se quiere recoger información.

También hay que indicar que formato tienen estos logs, para que se pueda "parsear" la información de forma que luego se puedan explotar estos ficheros de logs mediante consultas.

Disponemos de versiones para Windows y Linux, en el momento de realizar este trabajo de análisis la última versión disponible es la 7.12. También se puede instalar sobre Docker.

2.2.2 Winlogbeat

Su función es similar a la del agente Filebeat que hemos visto en el punto anterior, aunque en este caso es un componente específico de sistemas

Windows, dado que permite recolectar y enviar la información que se genera en el log de eventos de este S.O.

Aunque puede ser útil disponer de todos los eventos de Windows centralizados en Elastic en nuestro caso nos centraremos en los del Security Events Log.

Al igual que Filebeat la última versión publicada es la 7.12

2.2.3 Auditbeat

Este agente se instala en las plataformas Linux y permite extraer información de las librerías de Audit de este S.O. sin necesidad de cambiar la configuración del Daemon auditd.

De esta forma podemos enviar datos de las acciones realizadas por los usuarios en el S.O. Linux a la plataforma SIEM.

La última versión disponible es la 7.12 y también disponible como Docker.

2.2.4 Packetbeat

Este es el último de los “beats” que se van a analizar para nuestra plataforma SIRP como elemento de recolección de información de la red.

Este agente permite monitorizar el tráfico de red y analizar en tiempo real los paquetes que se envían o reciben por un host o un container y enviarlo a Logstash o Elasticsearch.

Esta función puede ser muy útil en nuestra plataforma si no podemos tener un NIDS o NIPS en todas las ubicaciones de red de nuestra organización y nos puede permitir analizar el tráfico de red entre equipos de nuestra red local, tráfico que habitualmente no haremos pasar a través de un NIDS o NIPS.

Packetbeat puede analizar los paquetes y decodificar los datos a nivel de aplicación HTTP, DNS, etc. por poner unos cuantos ejemplos de protocolos de capa 7.

La última versión disponible es la 7.12 (como en los “beats” anteriores) y también dispone de instalación en Docker.

2.2.6 Logstash

Es un componente de la suite de productos de Elastic que permite la recepción de logs, el parseado y la transformación de los mismos en tiempo real independientemente de su formato o complejidad.

En nuestro caso se utiliza para enviar los datos a Elasticsearch, pero la salida de datos pueden ser otros, envíos por correo, bbdd, etc.

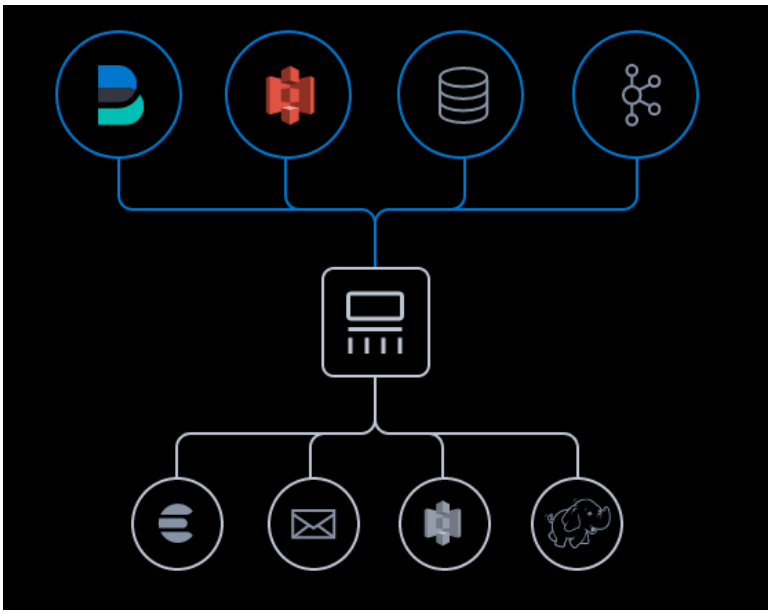


Ilustración 5 - Entradas / Salidas Logstash

Instalable sobre Linux, Windows, MacOS o en Docker, versión disponible en el momento de realizar este documento la 7.12

Capacidades de Logstash

Entradas: Ingesta fácilmente desde tus logs, métricas, aplicaciones web, almacenes de datos.

Filtros: Se pueden configurar filtros personalizados, aunque el producto ya dispone de una biblioteca de plantillas.

- Deriva estructura a partir de datos no estructurados con grok
- Descifra las coordenadas geográficas a partir de las direcciones IP
- Anonimiza datos PII y excluye campos sensibles por completo
- Facilita el procesamiento general, independientemente de la fuente de datos, el formato o el esquema.

Salidas: Elasticsearch es la base de la suite de productos para búsqueda y analítica de datos, pero no es la única opción posible. Logstash tiene una variedad de salidas que te permiten enrutar los datos donde lo desees, lo que te brinda la flexibilidad de desbloquear una gran cantidad de casos de uso posteriores.

2.2.7 Elasticsearch

Es el núcleo de la suite, el componente central de la herramienta. Es un motor de búsqueda y analítica distribuido con una RESTful API.

Una de sus funcionalidades principales es facilitar la búsqueda y para ello tiene un indexador que permite organizar la información de forma que se pueda responder a las consultas realizadas de forma ágil.

Está pensado para dar soporte a grandes volúmenes de datos, es una solución escalable se puede crear un cluster que permite añadir nodos para tener más capacidad de procesamiento de datos.

La herramienta está centrada en la búsqueda, permite búsquedas estructuradas y no estructuradas.

Actualmente, la última versión disponible del producto es la 7.12 y puede instalarse sobre Linux, Windows o MacOS, así como con tecnología de contenedores Docker.

Capacidades de Elasticsearch

Permite la gestión de los nodos en “Data tiers”, se pueden configurar nodos en función del performance de los mismos en tres categorías Hot, Warm y Cold y permite automatizar el movimiento de datos entre los mismos, aunque no lo utilizaremos para este TFM, en un SIEM real podríamos tener guardados los logs más recientes en nodos tipo Hot, donde la búsqueda tiene que ser en tiempo real y los datos más antiguos podrían quedar relegados a nodos con storage de bajo coste tipo Cold por ej., esto puede ser una necesidad en organizaciones que requieran almacenamiento de logs durante largos periodos de tiempo por requerimientos legales.

Son muchas las funciones que tiene el producto para disponer de replicas, snapshots, etc., como buen producto para usar en un entorno corporativo serio, tiene múltiples opciones para mantener el performance y la disponibilidad, no trataremos estas funciones aquí dado que el uso del producto que haremos en este TFM solo rascará la superficie.

Dispone también de un grupo de funcionalidades de seguridad, necesarias en este tipo de productos que pueden contener datos sensibles, una de las cosas básicas es que se necesita un buen control de acceso (la herramienta provee mecanismos RBAC y ABAC para afinar el acceso de los diferentes recursos almacenados en Elastic) y guarda un log de auditoría de accesos correctos o erróneos para poder detectar ataques. Otro punto importante es la encriptación de los datos, esta si es posible en las comunicaciones con Elasticsearch, lo que sería la encriptación de datos local se la delega al S.O. de los nodos.

Otra funcionalidad de seguridad que dispone el producto es el filtrado de IPs, puede aplicar filtrado de IP a clientes de aplicaciones, clientes de nodos o

clientes de transporte, además de otros nodos que intentan unirse al clúster de Elastic.

En cuanto a la autenticación tiene varios métodos para gestionarla: nativa, LDAP, SAML, etc. y permite utilizar SSO con Kibana (no es necesario utilizar usuarios/passwords diferentes al acceder vía Kibana o directamente a Elasticsearch).

Elasticsearch permite la definición de alertas y tiene un amplio set de notificaciones posibles: email, webhooks, IBM Resilient, Jira, Microsoft Teams, Slack, etc.

El producto provee de una potente RESTful API y JSON, lo cual permite el acceso al producto a través de código de aplicaciones en múltiples lenguajes Java, Go, .NET, PHP, Javascript, Perl, Python, Ruby, etc.

Dispone de más funcionalidades que se pueden consultar en la web del producto, pero que no detallaremos en este documento.

2.2.8 Kibana

Este componente es una interfaz de usuario open source que permite visualizar los datos de Elasticsearch de forma gráfica y visual a través de su interface web.

Se revisarán las principales características, aunque el producto tiene múltiples posibilidades para la explotación de datos, cuadros de mando, etc. no dedicaremos mucho tiempo ya que como se ha comentado es básicamente una interface visualización de datos.

En nuestro caso nos permitirá visualizar datos recopilados en Elasticsearch de las diferentes fuentes y en concreto de Wazuh. Puede ser una herramienta muy útil para un analista para encontrar patrones, tendencias o incluso con su capacidad de mostrar información geoespacial (sobre un mapa).

Al igual que el resto de la suite ELK está disponible para Linux, Windows y MacOS y para la plataforma de contenedores Docker.

Capacidades de Kibana

Como ya se ha comentado la capacidad principal del producto es la visualización de datos.

- Kibana Lens permite crear pantallas de datos con drag-and-drop
- Time series, charts, métricas, tablas de datos, información geoposicional (representa datos sobre un mapa), etc. son elementos que se pueden configurar y mostrar en Kibana

- Crear y configurar dashboards
- Interface de consola, permite utilizar código para hacer las consultas a Elasticsearch
- Generar grafos de análisis
- Permite la exportación de reports en modo PDF o imagen PNG
- Exportar información vía CSV

2.3 Análisis de componentes gestión de incidentes

Hasta ahora hemos analizado los productos que nos permiten detectar amenazas y en algunos casos responder de forma autónoma cosa que puede hacer un EDR y un NIPS, también hemos visto que pueden generar eventos, alertas y que estos se pueden enviar a nuestra plataforma SIEM, así como los logs de los S.O. y middleware, incluso el tráfico de red recibido por los equipos.

Con toda esa información en caso de que una amenaza se convierta en un incidente necesitamos una plataforma donde poder gestionarlo y esa plataforma es principalmente TheHive, con ayuda de Cortex y MISP.

2.3.1 TheHive

Hive es una herramienta open source que permite gestionar incidentes de seguridad, diseñado para ser utilizado por SOCs, CSIRTs, CERTs, etc. en general cualquiera que quiera gestionar incidentes de seguridad que necesitan ser investigados.

Es una solución escalable ya que permite crecer horizontalmente en número de nodos para mejorar el performance, necesita Java para su funcionamiento y utiliza Elasticsearch como repositorio de datos.

La herramienta permite compartir información, enriquecerse con fuentes externas (MISP) y responder a los incidentes con ayuda de otro componente del que hablaremos más adelante Cortex.

En el momento de realizar este estudio la última versión disponible es la 4.1.0
Se puede obtener servicios profesionales para la instalación, configuración y soporte del producto, lo cual es un valor adicional para su uso en organizaciones.

La implantación del producto se puede realizar tanto en una máquinas física o virtual (IaaS), como containerizada vía Docker (CaaS).

Capacidades de TheHive

Tal como se resume en la web del proyecto TheHive, estas son las funcionalidades básicas del producto.

Colaboración:

- Varios analistas de SOC y CERT pueden colaborar en las investigaciones simultáneamente
- Gracias a la transmisión en vivo incorporada, la información en tiempo real relacionada con casos, tareas, observables e IOC nuevos o existentes están disponibles para todos los miembros del equipo.
- Las notificaciones especiales les permiten manejar o asignar nuevas tareas y obtener una vista previa de los nuevos eventos de MISP y alertas de múltiples fuentes, como informes de correo electrónico, proveedores de CTI y SIEM. Luego pueden importarlos e investigarlos de inmediato.

Elaboración:

- Los casos y las tareas asociadas se pueden crear utilizando un motor de plantillas simple pero potente.
- Puede agregar métricas y campos personalizados a sus plantillas para impulsar la actividad de su equipo, identificar el tipo de investigaciones que requieren un tiempo significativo y buscar automatizar tareas tediosas a través de cuadros de mando dinámicos.
- Los analistas pueden registrar su progreso, adjuntar pruebas o archivos notables, agregar etiquetas e importar archivos ZIP protegidos con contraseña que contienen malware o datos sospechosos sin abrirlos.

Actuación:

- Agregue uno, cientos o miles de observables a cada caso que cree o impórtelos directamente desde un evento MISP o cualquier alerta enviada a la plataforma.
- Clasifíquelos y fíltrelos rápidamente.
- Aproveche el poder de Cortex y sus analizadores y respondedores para obtener información valiosa, acelerar su investigación y contener las amenazas.
- Aproveche las etiquetas, marque las IOC, los avistamientos e identifique los observables vistos anteriormente para alimentar su inteligencia de amenazas.
- Una vez completadas las investigaciones, exporte las IOC a una o varias instancias de MISP.

El producto TheHive provee de una REST API para la interacción y automatización de tareas aparte del acceso vía web para usuarios interactivos.

2.3.2 Cortex

Cortex es un analizador de observables y un motor de respuesta activa que permite en conjunción con TheHive ayudar en la fase de contención de un incidente de seguridad.

La herramienta es open source y permite automatizar el tratamiento de los observables exponiendo una REST API.

Aunque Cortex es free eso no significa que los analizadores y respondedores que incluye al hacer peticiones a terceros no requieran de un pago o suscripción por uso.

Capacidades de Cortex

Cortex se puede usar de forma independiente usando su “interface” web o bien utilizarlo conjuntamente con la plataforma de gestión de incidentes TheHive, como ya se ha comentado permite automatizar y ejecutar de forma simultánea analizadores y respondedores para múltiples casos.

Analizadores / Respondedores:

- Dispone de un gran conjunto de analizadores o se pueden crear propios, así como respondedores usando cualquier lenguaje de programación que pueda correr sobre Linux.
- Estos analizadores o respondedores al utilizarlos en la herramienta Cortex pueden estar disponibles para todo el equipo de trabajo (esto es útil por varios motivos, cada miembro del equipo de trabajo no tiene que realizar sus propios desarrollos y pueden compartir API KEYS de servicios externos que pueden ser free o de pago.
- Permite consultar varias instancias de MISP (componente que más adelante trataremos).

Interacción:

- TheHive puede conectarse a una o varias instancias de Cortex y puede analizar un gran número de observables a la vez, así como dar respuestas activas.
- Con el motor de informes de TheHive, es fácil analizar la salida de Cortex y mostrarla de la forma que desee.
- También puede usar Cortex como un producto independiente usando su interfaz de usuario web para administrar múltiples organizaciones, analizadores y configurar límites de consultas.
- Cortex puede interactuar con otros productos a través de su API REST o mediante Cortex4py.

Ejecución:

- Cortex viene con más de cien analizadores para servicios populares como VirusTotal, Joe Sandbox, DomainTools, PassiveTotal, Google Safe Browsing, Shodan y Onyphe. Identifique contactos abusivos, analice archivos en varios formatos como OLE y OpenXML para detectar macros VBA, genere información útil en archivos PE, PDF y mucho más.

- Los analizadores de Cortex también se pueden consultar desde MISP para enriquecer los eventos y ampliar la cobertura de sus investigaciones.

2.3.3 MISP

Es una plataforma open source para tratamiento de inteligencia de amenazas y un crear un estándar abierto para poder compartir esa inteligencia entre organizaciones y los respectivos CERTs, CSIRTs, etc.

La clave de este producto es la automatización, permitiendo compartir los indicadores de compromiso (IoCs) con las diferentes herramientas de las plataformas de detección y respuesta ante amenazas (IDSs, SIEMs, etc.)

La plataforma permite guardar de forma estructura los IoCs y compartirlos con otros MISP y su facilidad de uso hemos que se haya extendido su uso.

En el momento de realizar este análisis la mayor versión de MISP es la 2.4, para instalar el producto se puede recurrir a una OVA, instalar sobre un S.O. ya existente o bien con Docker, en la documentación del producto también se incluyen múltiples opciones para realizar su instalación con puppet, ansible, etc.

Recomiendan la instalación sobre Ubuntu 20.04 LTS.

Capacidades de MISP

Como ya se ha comentado uno de los mayores logros de esta herramienta ha sido su extendido uso y la compartición de información entre organizaciones.

Los ataques realizados por ciberdelincuentes e incluso promovidos o respaldados en mayor o menor medida por estados crecen en número y sofisticación y esto hace necesario que las organizaciones también trabajen de forma conjunta para protegerse de estas amenazas y MISP es una herramienta que fomenta este tipo de interacción de forma automatizada, compartiendo la inteligencia que obtiene cada organización tras analizar e investigar un ataque.

En la web de MISP podemos encontrar sus funcionalidades [11]:

- Permite disponer de una base de datos de indicadores y de IoCs que permite almacenar información técnica y no técnica sobre muestras de malware, incidentes, atacantes e inteligencia.
- Correlación automática, encontrando relaciones entre atributos e indicadores de malware, campañas de ataques o análisis. La correlación también puede habilitarse o inhabilitarse por evento o por atributo.

- Proporciona un modelo de datos flexible donde los objetos complejos se pueden expresar y vincular para expresar inteligencia de amenazas, incidentes o elementos conectados.
- Facilita el intercambio de datos utilizando diferentes modelos de distribuciones. MISP puede sincronizar automáticamente eventos y atributos entre diferentes MISP. Se pueden utilizar mecanismos de filtrado para que cada organización decida que comparte y que no.
- Provee una interfaz de usuario intuitiva para que los usuarios finales creen, actualicen y colaboren en eventos y atributos / indicadores. Una interfaz gráfica para navegar sin problemas entre eventos y sus correlaciones.
- Provee una funcionalidad de grafo de eventos para crear y ver relaciones entre objetos y atributos. Funciones de filtrado avanzado y lista de advertencias para ayudar a los analistas a contribuir con eventos y atributos.
- Almacenamiento de datos en un formato estructurado (que permite el uso automatizado de la base de datos para diversos fines)
- Exportar información para generar IDS (Suricata, Snort y Bro son compatibles de forma predeterminada), OpenIOC, texto sin formato, CSV, MISP XML o salida JSON para integrar con otros sistemas (IDS de red, IDS de host, herramientas personalizadas)
- Importación de datos masiva, importación por lotes, importación de texto libre, importación desde OpenIOC, GFI sandbox, ThreatConnect CSV o formato MISP, se ofrecen múltiples opciones. Herramienta flexible de importación de texto libre para facilitar la integración de informes no estructurados en MISP.
- Intercambio de datos: intercambio automático y sincronización con otras partes y grupos de confianza utilizando MISP.
- Herramienta flexible para importar e integrar en MISP cualquier fuente de inteligencia u OSINT de terceros. MISP ya incluye de entrada muchas fuentes.
- Delegación de compartir: permite un mecanismo pseudo-anónimo simple para delegar la publicación de eventos / indicadores a otra organización (esto puede ser útil para no desvelar que tipo de ataques sufre tú organización, puede ser información sensible y que por tanto no se quiera compartir).
- API flexible para integrar MISP con otras herramientas propias. Se incluye una librería Python denominada PyMISP para buscar, agregar o actualizar atributos de eventos, manejar muestras de malware, etc.
- Clasificación personalizable o utilización de clases definidas. La taxonomía (clasificación) puede ser local para tu MISP pero también se puede compartir entre instancias de MISP. MISP viene con un conjunto predeterminado de taxonomías y esquemas de clasificación para respaldar un estándar como ENISA, Europol, DHS, CSIRT o muchas otras organizaciones.
- Vocabularios de inteligencia llamados MISP galaxy y que se incluyen con los actores de amenazas existentes, malware, RAT, ransomware o MITRE ATT & CK que se pueden vincular con eventos en MISP.

- Soporte para observaciones de organizaciones sobre indicadores y atributos compartidos. Se puede contribuir con observables a través de la interfaz de usuario de MISP, vía API como documento MISP o documentos de observación STIX.
- Compatibilidad con STIX: exportación de datos en formato STIX (XML y JSON), en formato STIX 2.0.
- Cifrado integrado y firma de notificaciones vía PGP o S/MIME dependiendo configurable por el usuario.
- Canal de publicación-suscripción en tiempo real dentro de MISP para obtener automáticamente todos los cambios (por ejemplo, nuevos eventos, indicadores, avistamientos o etiquetado).

En nuestra instalación configuraremos MISP para importar información, dado que no tendremos casos para poder compartir, aunque se podría realizar alguna configuración simulando un entorno real de trabajo en el cual podemos contribuir a la comunidad.

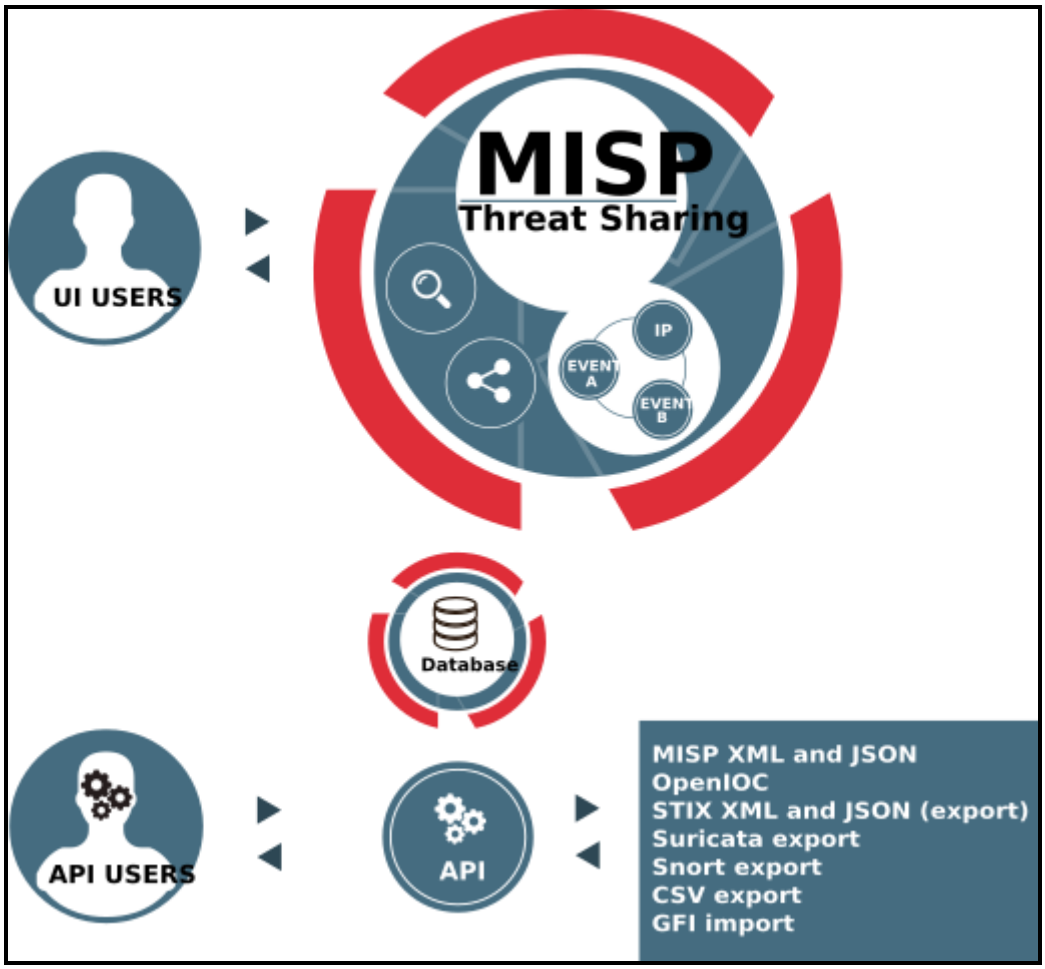


Ilustración 6 - Esquema de MISP

2.4 Elementos que compondrán nuestra plataforma SIRP

Después de haber analizado los diferentes componentes que podrían formar parte de una solución SIRP y que se han tratado en los apartados anteriores ha llegado el momento de escoger que implementaremos en nuestra solución y porque estos componentes y no otros.

Para empezar, definiremos los casos de uso que tendrá nuestra plataforma lo cual facilitará la elección de los componentes, esto no significa que la plataforma no pueda dar respuesta a más casos de uso, pero sí que no serán tratados en este TFM.

2.4.1 Casos de uso

Los casos de uso propuestos para este TFM consisten en varias casuísticas de gestión de incidentes que pongan a prueba los diferentes elementos de nuestra plataforma.

1) Gestión de un incidente con malware detectado por el EDR, reportado al SIEM, creando un caso, investigándolo y añadiendo la información complementaria que sea necesaria hasta su resolución

2) Gestión de un incidente con amenazas detectables por red con nuestro NIPS, reportado al SIEM, creación del caso, investigación, adición de información si procede y respuesta

3) Gestión de un incidente con información de eventos recogidos por el SIEM, creación de caso, investigación, adición de información si procede y respuesta

La idea es que con las herramientas indicadas se puedan gestionar los incidentes, con enriquecimiento de información de fuentes externas y si es posible con respuestas automatizables.

Para la presentación y defensa del TFM se escogerá uno de los tres casos de uso para mostrar el trabajo realizado y cómo se comporta en un entorno de pruebas generado al uso y que se documentará a partir del punto 3.

2.4.2 Componentes de la solución

Estos serían los componentes de nuestra plataforma SIRP que incluye como ya se ha tratado los elementos de detección y respuesta, la gestión de eventos y logs de seguridad, así como la gestión de incidentes con integración con automatismos de tratamiento de los observables de los incidentes, así como de las respuestas a los mismos para mejorar en la fase de contención.

También incluimos un elemento que nos permite compartir y colaborar con otras organizaciones con nuestros incidentes analizados, ya investigados y

trabajados y también enriquecer nuestros casos con la información que podamos recolectar.

- Wazuh
- Sysmon
- Suricata
- ELK
- Elastalert
- TheHive
- Cortex
- MISP

Sobre algunas decisiones que se han tomado para escoger los componentes, en el caso del NIPS, se han estudiado tanto Snort, como Suricata que se podría decir que son líderes para esta función en software open source.

Se ha escogido Suricata de cara a la implementación por varios motivos, es un software más nuevo y lleva un diseño de código más moderno (ha nacido multithread), lo cual normalmente mejora el rendimiento. Snort según su documentación parece funcionar en modo multiproceso, a partir de la versión 3 (anteriormente se podía usar de esta manera arrancando varias instancias), ahora se incorpora un parámetro para ello.

Las reglas de Snort se puede usar también en Suricata, en este tipo de software aparte de las capacidades propias del producto, su capacidad de detección depende de sus reglas para ser mejor en la detección de amenazas.

Por otro lado, también lo he escogido por referencias de integración tanto de Wazuh, como del MISP.

Se ha incluido también Elastalert que no se ha explicado en profundidad en este capítulo para la generación de alertas y creación de casos en TheHive, aunque dado que el requisito es la generación de casos para la investigación de incidentes en TheHive es posible utilizar también la opción que incluye este producto TheHive4py, así que temporalmente incluimos el componente, pero en la fase de implementación e integración veremos finalmente cual es la mejor opción.

El resto de componentes de la solución se han escogido en base a los casos de uso que se han propuesta tratar con esta plataforma SIRP.

2.5 Diseño de la solución

La solución para nuestra plataforma SIRP consiste en implementar los componentes indicados en el apartado anterior como si se tratara de un servicio cloud (SOCaaS), se trataría de ofrecer a nuestros posibles clientes una plataforma completa de gestión de incidentes de seguridad en la cual podrían trabajar como usuarios o delegar completamente en nosotros (su proveedor el servicio de SOC y la plataforma asociada) para realizar todas las tareas.

En este caso y para simular un entorno real necesitaremos elementos adicionales que no se han comentado anteriormente porque no forman parte de la solución SIRP como tal, como es la necesidad de disponer de una conexión VPN entre nuestro site en el cloud y la empresa u organización a la que gestionaremos su seguridad con nuestra plataforma.

En la red de la empresa a la que dará servicio esta plataforma SIRP se deberán ubicar los elementos de detección y respuesta necesarios, como son los agentes de Wazuh y los IPS, así como habilitar los accesos necesarios para poder gestionar la instalación, recepción de logs en la plataforma SIEM incluida en nuestra solución SIRP y permisos para poder ejecutar los scripts de respuesta en función del tipo de alerta detectada si procede.

A continuación, se muestra un esquema lógico de la solución y sus componentes.

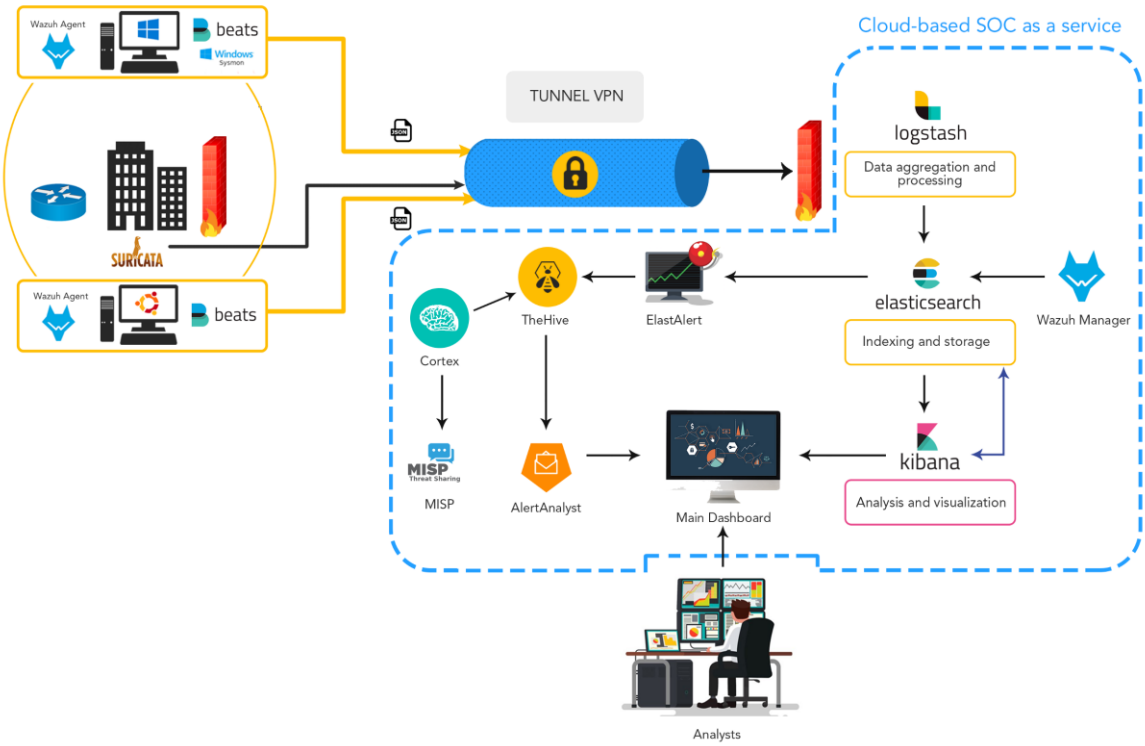


Ilustración 7 - Esquema lógico de componentes

3. Implementación

3.1 Opciones de implementación

Una vez decididos los componentes que integran la solución, ahora hay que tomar decisiones sobre la implementación. La implementación tiene como requisitos tener alta disponibilidad y ser una solución escalable, por lo que se requiere un alto componente de automatización.

Descartadas directamente las máquinas físicas porque no tengo equipos suficientes para poder simular un entorno operativo, la solución se basará en máquinas virtuales, en mi caso utilizando VirtualBox, pero la solución de virtualización a priori es indiferente para el montaje a realizar.

Así pues, se han sopesado las múltiples opciones de instalación de los productos, así como sus versiones, dado que no todas están disponibles para instalar en cualquier modalidad, es decir, las últimas versiones de los productos mencionadas en el capítulo anterior de este documento están pensadas para hacer instalaciones manuales descargando los productos de las webs de la organización que patrocina o dirige el proyecto open source correspondiente.

Por poner un ej. para que se entienda lo comentado en el punto anterior, la versión documentada para instalar con Wazuh de ELK no es la de la web de Elastic sino la Opendistro Elasticsearch, aunque en la web de Wazuh se tratan opciones de instalación también con la opción free de Elastic, la última versión de este producto que es la 7.12 en el caso concreto de Kibana no tiene disponible el plugin de Wazuh (está disponible para la 7.10.2).

Aunque se podría considerar alguna opción más para la implantación, se han valorado las siguientes:

3.1.1 Instalación manual de los productos

Los diferentes productos disponen de opción de instalación más o menos documentada para hacer la instalación y configuración de los diferentes elementos vía paquetes estándar de SO y/o fuentes vía repositorios principalmente de tipo GIT.

También existen OVAs con los productos ya preinstalados y configurados sobre un SO Linux como es el caso del CSIRT-KIT.

Esto puede ser muy útil para un entorno de pruebas y para practicar con la funcionalidad de los productos de forma relativamente rápida (sobre todo el caso del CSIRT-KIT), pero como en nuestro caso queremos tener automatización, alta disponibilidad y escalabilidad de la solución pues este método ha sido descartado.

3.1.2 Instalación automatizada con Ansible

Ansible es una herramienta de software de código abierto que se ha hecho muy popular en los últimos años. Permite la gestión de configuraciones y despliegue de aplicaciones y productos de forma centralizada, simplificando mucho, mediante recetas llamadas playbooks (en un lenguaje declarativo YAML) que utilizan módulos específicos para por ejemplo gestionar paquetes de SO, copia de ficheros y otras muchas funciones, el producto incluye muchos módulos y además la comunidad sigue creando nuevos. Existen repositorios como ansible-galaxy donde se pueden encontrar multitud de ejemplos de playbooks y roles (concepto de ansible) para realizar diversas tareas en aplicaciones y sistemas gestionados.

En el caso concreto de la implantación de la plataforma SIRP, de los productos que tenemos contemplados en la web de Wazuh ya facilitan procedimiento, aunque igual requiere algunos retoques que permiten hacer la instalación de Wazuh y Elastic, así como la instalación de agentes Wazuh y Filebeat.

Para el resto de productos elegidos para la plataforma SIRP, igual existen Playbooks ya preparados para la instalación, como ya se ha comentado la comunidad facilita esto en muchos casos vía repositorios GIT, etc.

3.1.3 Instalación con Docker / Docker Compose

Docker es una de las plataformas de contenedores más utilizada hoy en día, permite segregar en lo que se conoce como microservicios las aplicaciones e infraestructura necesaria para su ejecución.

El contenedor se diferencia entre otras cosas de la virtualización, en que es mucho más ligero, dado que no tienes que implementar el SO completo, sino una capa mínima de sistema y las librerías necesarias para correr la aplicación/producto.

Permite trabajar con imágenes pre-creadas y los contenedores se crean y destruyen y se vuelven a recrear en base a estas imágenes. Si se requiere una persistencia, los contenedores requieren crear volúmenes o mapear directorios sobre el host donde se ejecutan los contenedores, de forma que se persistan los datos aunque el contenedor sea destruido.

A nivel de seguridad, solo exponen aquellos puertos que son necesarios que sean visibles desde el exterior, el resto de puertos solo son visibles internamente por la red de contenedores (que a priori esta oculta al exterior).

Docker Compose es una herramienta que permite “orquestar”, más bien automatizar mediante una receta descriptiva en YAML que contenedores se han de crearse y con qué características y recursos (redes y volúmenes básicamente).

En nuestro caso concreto para la plataforma SIRP sería una buena solución y nos permite cumplir con el objetivo de automatización, para la parte de alta disponibilidad y escalabilidad bastaría con disponer de más hardware con capacidad para ejecutar contenedores y aplicar la receta personalizada para crear y gestionar los contenedores necesarios.

Existe solución para contenedores para Wazuh, Elasticsearch (tanto OpenDistro, como la más corporativa de Elastic), así como para TheHive, Cortex y MISP, en algunos casos no trabajan con las últimas versiones de producto, pero permiten utilizar dichas imágenes. Siempre existe también la opción de generarse una imagen propia o basada en la modificación de una existente, mediante el uso de archivos de configuración Dockerfile.

3.1.4 Instalación sobre Kubernetes

Llegados a este punto ya tenemos una solución para poder desplegar de forma automática los componentes aplicativos adaptando las “recetas” a los diferentes equipos como ya se ha comentado en el punto anterior, pero para hacer una verdadera orquestación de los componentes tenemos Kubernetes.

Kubernetes (habitualmente nombrado como k8s) tal como se indica en su página web es una plataforma portable y extensible open source para administrar cargas de trabajo y servicios, al igual que con la opción de Docker-Compose tiene una forma declarativa de gestionar el despliegue de componentes (microservicios), ejecutando el deploy de los PODs y servicios necesarios para el funcionamiento de la aplicación o producto. Y todo ello, mediante un entorno de gestión APIficado.

Google liberó este proyecto en 2014 y desde entonces se ha ido popularizando hasta el punto que hay varias versiones que implementan kubernetes (kubeadm, minikube, kops, etc.), cada una tiene unas funcionalidades básicas comunes (por ej. los comandos) pero luego se diferencian en los componentes base de la solución así como la aplicación del mismo (por ej. minikube se suele utilizar en entornos monomodo y/o para hacer pruebas con la tecnología kubernetes, aunque es una solución 100 % operativa).

Kubernetes es una solución tan flexible que permite la portabilidad entre diferentes proveedores de Cloud públicos (Azure, AWS, GCP, etc.) o Cloud privados (on-premise). Es una plataforma que trata de potenciar las metodologías agiles y el DevOps, está pensada para dar soporte al CI/CD.

En nuestro caso utilizaremos Docker como plataforma de contenedores, pero es una solución tan flexible que a priori se independiza de la plataforma de contenedores escogida (CRI, Containerd …), múltiples herramientas de gestión, addons, etc.

Antes hemos comentado también el concepto de POD, que es la unidad mínima de Kubernetes que vendría a ser como una envoltura de uno o más contenedores que cumplen unas ciertas características.

Una vez introducida la tecnología, vamos a nuestro caso práctico que es la plataforma SIRP con los componentes ya mencionados. Existe documentación sobre la instalación de Wazuh / Elasticsearch en plataforma k8s, en el caso del resto de componentes, aunque es posible que exista alguna solución a nivel de comunidad open source, lo que si que nos podemos basar en las implementaciones realizadas directamente en Docker o Docker-Compose para desarrollar nuestra propia receta de instalación partiendo de las imágenes ya disponibles (TheHive, Cortex y MISP).

Así pues, con este sistema de implantación podemos tener una solución con alta disponibilidad, escalabilidad y automatización en el despliegue y su mantenimiento.

3.1.5 Solución de instalación escogida

Después de tratar las diferentes opciones que disponemos para la implantación de nuestra solución SIRP, se ha considerado que la mejor opción es utilizar la plataforma Kubernetes porque proporciona todas las ventajas que puede necesitar un despliegue de este tipo con múltiples componentes.

Para la instalación de los productos base de la plataforma SIRP, los ya mencionados Wazuh, Elasticsearch, TheHive / Cortex y MISP utilizaremos Kubernetes y para el despliegue de los agentes, léase agente de Wazuh y los varios Beats utilizaremos la herramienta comentada también en los puntos anteriores Ansible.

Por lo tanto, se intentará utilizar la documentación y procedimientos suministrados en las diferentes webs de estos proyectos open source adaptándolos a nuestro entorno de simulación, el cual pasamos a detallar en los siguientes puntos de este documento.

Para el entorno kubernetes se ha escogido implementar un Cluster en HA con Kubeadm, en la cual tendremos 3 nodos master y 4 nodos worker (todos tienen instalado el agente kubelet), en terminología kubernetes los nodos master o control plane son nodos que contienen los componentes apiserver, controller-manager, scheduler y etcd (bbdd clave-valor que guarda información del cluster, configuraciones, secretos, servicios, etc. etc.).

Los nodos master ya que podríamos decir que son nodos de administración en este caso de kubernetes también se han reaprovechado para instalar ansible

Por temas de seguridad, se debe inhabilitar el despliegue de aplicativos u otros componentes en los nodos master, por lo que dispondremos de 4 nodos efectivos para desplegar nuestra solución SIRP.

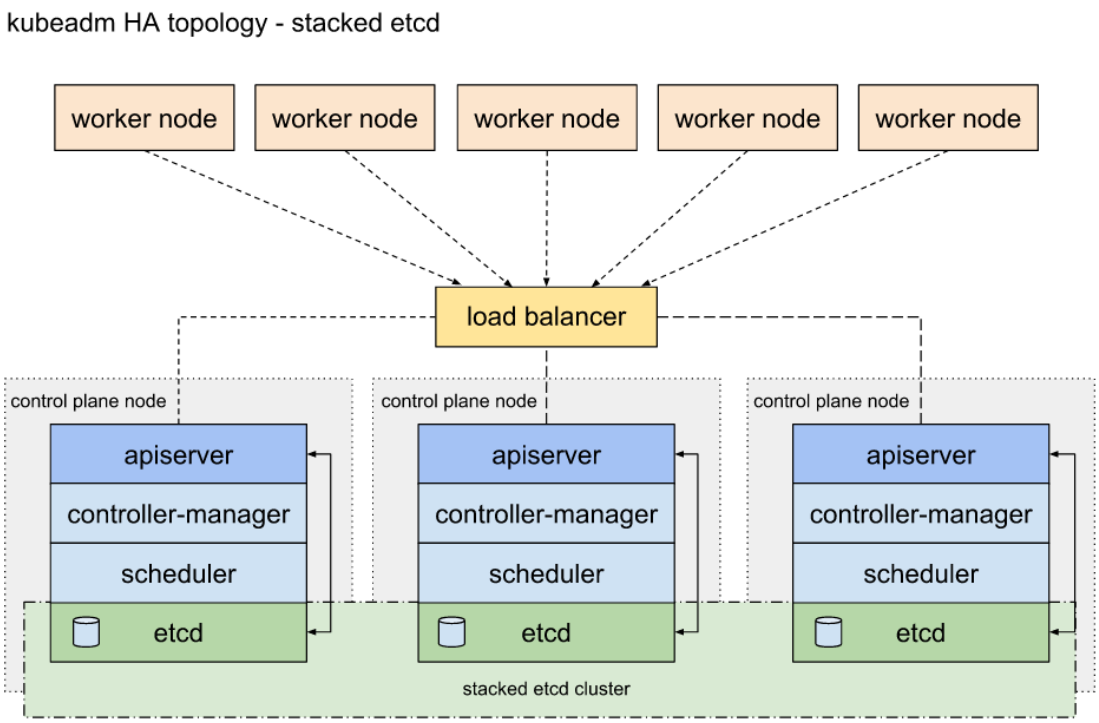


Ilustración 7 - HA kubernetes etcd stacked (kubernetes.io)

Hay dos modelos para montar HA de Kubernetes con kubeadm, el modelo topológico de la imagen que es el seguido en este proyecto “etcd stacked” y el modelo “etcd external”.

3.2 Recursos

En el entorno de simulación utilizado en este proyecto que se ha implementado podemos dividir los recursos utilizados en dos partes, los que componen la red empresarial a proteger y lo que sería la plataforma SIRP (simulación de un entorno Cloud con Kubernetes).

Toda la solución corre sobre 4 equipos físicos con procesadores Intel i7 QuadCore y 32 GB de memoria RAM.

3.2.1 Plataforma SIRP (simulación Cloud On-premise)

Los recursos utilizados en la simulación del entorno Cloud On-premise o Cloud Privado son los siguientes:

- 1 VM - Firewall Pf-Sense (2 vCPUs, 4 GB RAM y GB espacio en disco) Software:
    - Appliance Pf-Sense (basado en FreeBSD)

- 7 VMs - Cluster Kubernetes:
    - 3 VMs nodos Master (2 vCPUs y 4 GB RAM y 100 GB disco)
    - 4 VMs nodos Worker (2 vCPUs y 8 GB RAM y 100 GB disco)

  Software:

  Base:
    - S.O. CentOS 8.3
    - HAProxy 1.8.23
    - Keepalived 2.0.10
    - GlusferFS 9.1
    - Docker CE 20.10.6
    - Docker-Compose 1.29.1
    - Kubernetes 1.21.0 (kubectl, kubeadm, kubelet)
    - Ansible 2.9.18 (solo en nodos Master)

  Aplicativos:
    - Wazuh 4.1.1
    - Opendistro Elasticsearch 1.12.0 (Elasticsearch y Kibana 7.10)
    - Filebeat 7.10
    - TheHive 4.1.4
    - Cortex 3.1.1
    - MISP 2.4.142

3.2.2 Red Empresarial

Los recursos utilizados en la red empresarial son los siguientes:

- 1 VM – Firewall Pf-Sense (2 vCPUs, 4 GB RAM y GB espacio en disco)
  Software:
    - Appliance Pf-Sense (OVA basada en FreeBSD)

- 1 VM – Suricata (IPS)
  Software:
    - S.O. CentOS 8.3
    - Suricata 5.0.6
    - Wazuh Agent 4.1.5

- 1 VM – DNS externo, Proxy Usuarios, Proxy Revers
  Software:
    - S.O. CentOS 8.3
    - Bind 9.11
    - Apache 2.4.37
    - Squid 4.4
    - Wazuh Agent 4.1.5

- 1 VM – Web Server, App. Server y BBDD
  Software:
    - Appliance BWA (OVA basada en Ubuntu)

- 1 VM – Domain Controller AD y DNS interno
  Software:
  - Windows Server 2019
  - Sysmon v13.20
  - Wazuh Agent 4.1.5

- 1 VM – Estaciones de trabajo
  Software:
  - Windows 10
  - Wazuh Agent 4.1.5

3.3 Otros servicios

Nuestra plataforma SIRP utiliza un servicio DNS externo (dominio cyberhome.es) que no está dentro del entorno, podría estarlo, pero he creído que no tiene sentido (podría tenerlo sí lo enlazamos quizá con el servicio kube-dns de Kubernetes que es la pieza dedicada a resolver nombres de recursos).

Por otro lado, para tener el entorno securizado y al margen de que el canal simulado a través de Internet tenga una conexión cifrada por VPN, los productos requieren certificados para su securización, los que están expuestos al usuario (fuera del cluster Kubernetes), así como los que se requieren en los procedimientos de instalación de productos han sido firmados por una CA externa (privada). Otras opciones pueden ser PKIs públicas con el correspondiente coste asociado o recurrir a CAs públicas gratuitas tipo Let'sEncrypt.

3.4 Esquema del entorno de simulación

En apartados anteriores hemos visto un esquema de la solución a nivel de componentes, ahora se muestra un esquema del entorno con los equipos (máquinas virtuales) que componen tanto la solución SIRP, como la empresa u organización a proteger.

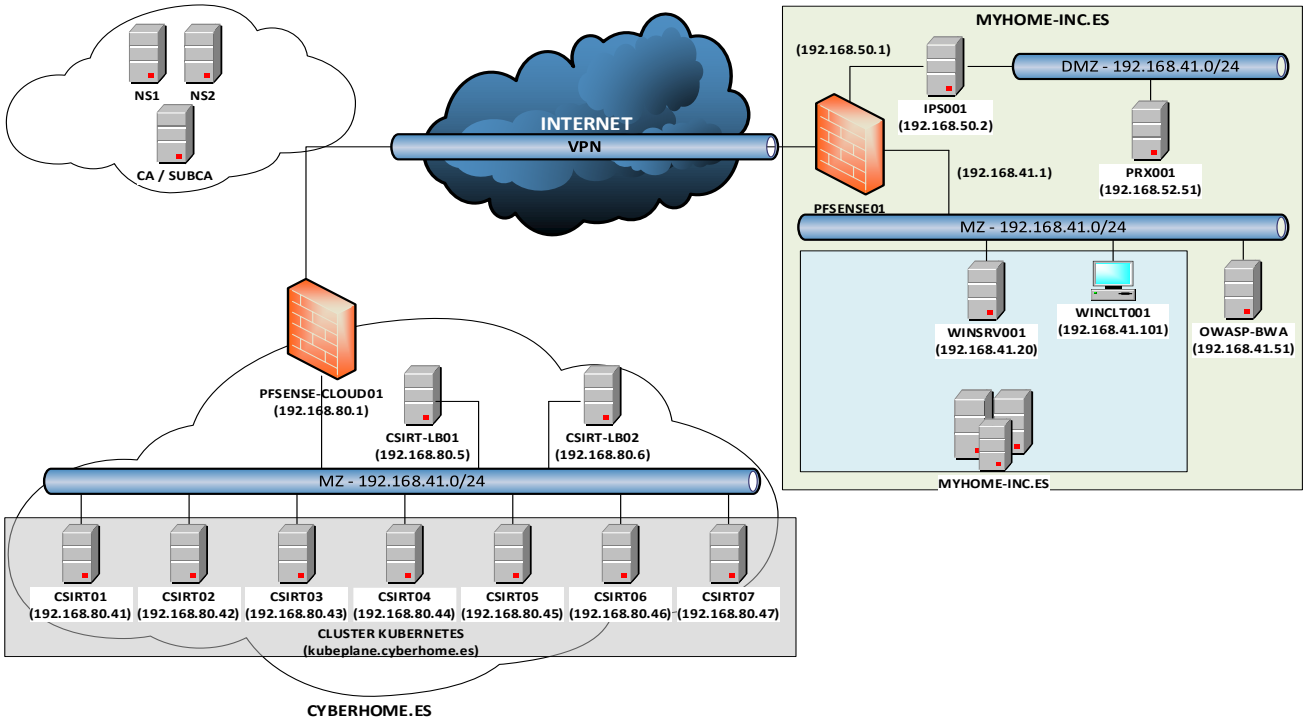


Ilustración 8 - Entorno de simulación

3.5 Dimensionamiento de los componentes

Aunque se trata de un entorno simulado para poder hacerlo lo más real posible y mostrar las posibilidades de escalabilidad de los componentes de nuestra plataforma SIRP se ha realizado el siguiente dimensionamiento.

- 1 nodo wazuh-master y 2 nodos wazuh-workers
- 3 elasticsearch nodes
- 1 kibana
- 1 TheHive
- 1 Cortex
- 1 MISP
- 1 Praeco-Elastalert

Esto en un entorno real, si no fuera suficiente, se puede solucionar con la adición de nodos si el problema es de capacidad de CPU o memoria, o añadiendo espacio en disco (el cual depende del número de eventos a registrar en Elastic, número de agentes, etc.).

Al estar instalado en kubernetes podemos “jugar” con los replicasets y añadir más pods en cada deploy.

En el caso de TheHive, Cortex y MISP no se han levantados réplicas de los productos, dado que según indica la documentación de los diferentes proyectos, sobre todo el caso de The Hive y Cortex requieren bastantes recursos y por lo tanto ya que no se dispone de más hardware, se ha realizado una instalación de mínimos para poder utilizar los productos.

3.6 Fuentes de los componentes

Como ya se ha comentado en el entorno Docker y kubernetes se trabaja con pequeñas imágenes con lo imprescindible para poder realizar la tarea (microservicio) que tienen que implementar.

A continuación, se citan las fuentes / repositorios de donde se ha obtenido el software o bien las imágenes y/o ficheros de configuración para cada producto.

Wazuh

Este repositorio contiene los ficheros de configuración para deployar Wazuh con Elasticsearch Opendistro y Kibana, viene preparado principalmente para funcionar en EKS de Amazon y tiene una instalación más genérica para un entorno local.

```
https://github.com/wazuh/wazuh-kubernetes.git
```

Para la instalación hemos seguido la versión de instalación en un entorno local (como es el caso), aunque requiere customización de los ficheros sobre todo para los temas de storage.

En el entorno se ha instalado con disco local, mapeando directorios afines a los diferentes nodos, pero que están visibles desde todos ya que se ha montado un filesystem distribuido con GlusterFS.

Existe la opción de utilizar un storage class provider directamente con GlusterFS, pero requería la instalación de más componentes ya que se requiere una API para poder hacer la petición de espacio (PVC: Persistent Volume Claim).

Los certificados creados en el proceso de instalación han sido firmados por una CA y SubCA propias no creada explícitamente para este proyecto, sino preexistente.

Los nombres de servicios y puertos expuestos son:

- wazuh-master.cyberhome.es (API Server TCP/55000 y authd TCP/1515)
- wazuh-manager.cyberhome.es (reporting service TCP/1514)

- wazuh-kibana.cyberhome.es alias de kibana.cyberhome.es (dashboards kibana y plugin wazuh TCP/443)
- elasticsearch.cyberhome.es (ingestar en Elastic y consultas TCP/9200)

The Hive / Cortex

En la instalación de The Hive y Cortex tenemos los templates de instalación en el siguiente repositorio de GIT:

```
https://github.com/TheHive-Project/Docker-Templates
```

De las múltiples versiones disponibles de templates en la que se contemplan instalaciones de diferentes versiones de TheHive, Cortex e incluso Elasticsearch, se ha escogido la opción con etiqueta **thehive4-cortex31-nginx-https**

En este repositorio se dispone del template para utilizar con Docker-compose. En nuestro caso y aunque tenemos Docker como plataforma de contenedores y podría utilizarse de forma directa, dado que tenemos por encima kubernetes para hacer la gestión de despliegues no lo hacemos directamente, sino que generamos un fichero “template” para kubernetes con **Kompose**, realizando algunas modificaciones, así como eliminando la parte de Elasticsearch, ya que disponemos de este componente vía la instalación anterior del producto Wazuh, a priori es compatible al tratarse de una versión 7.x

La instalación incluye también la BBDD Casandra para dar soporte a TheHive.

Respecto al espacio en disco hemos utilizado el espacio compartido por el GlusterFS (así la aplicación puede arrancar en cualquier nodo worker del kubernetes).

Los certificados creados en el proceso de instalación han sido firmados por la misma CA y SubCA que los certificados de la instalación Wazuh, Elasticsearch y Kibana.

Los nombres de servicios y puertos expuestos son:

- thehive.cyberhome.es (TCP/443)
- cortex.cyberhome.es (TCP/443)

MISP

Para la instalación de MISP disponemos de un repositorio GIT accesible desde la web del proyecto open source MISP y es el siguiente:

```
https://github.com/misp/misp-docker
```

Aquí podemos encontrar un template para realizar la instalación directamente sobre plataforma Docker, al igual que en el caso anterior hay que transformar

nuestra plantilla de Docker-compose en una plantilla para poder instalar sobre kubernetes.

En este caso para el funcionamiento de MISP es necesario disponer también de un MYSQL versión 5.7 como nos indica la versión de la imagen a descargar.

Aunque en el repositorio de Docker Hub existen algunas imágenes de MISP en este caso utilizando el procedimiento de instalación de este repositorio es necesario crear la imagen de MISP (que a su vez se basa en una mini imagen de Ubuntu).

Los nombres de servicios y puertos expuestos son:

- misp.cyberhome.es (TCP/443)

4. Integración

Hasta ahora se han instalado los componentes que conforman nuestra plataforma SIRP, ahora se trata de configurarlos e integrarlos.

La configuración de cada uno de los productos, utilizando todas sus posibilidades nos podría llevar mucho tiempo, por lo tanto, se han realizado las configuraciones mínimas necesarias para poder tenerlos operativos y realizar las tareas necesarias para el caso de uso que expondremos más adelante.

4.1 Wazuh-Elasticsearch-Kibana

Una vez realizada la instalación el producto ya está operativo, es decir, una se han deployado los pods, se han aplicado las configuraciones y se han inicializado los datos, la configuración básica para su funcionamiento ya está implementada, dado que es una suite de productos que está configurada de base para integrarse en el proceso de instalación y activación de los componentes.

Wazuh podría funcionar solo, pero no tendríamos donde visualizar la información que nos reporta sin elasticsearch y kibana, puede operarse a través de una API, pero esto es más útil para integraciones con otras aplicaciones o para desarrollos propios. Con kibana lo podemos explotar de una forma más práctica y visual vía web.

Para ello durante la instalación se configura una plantilla para poder usar elasticsearch y se instala un plugin que es utilizado desde kibana para poder visualizar y administrar nuestro Wazuh.

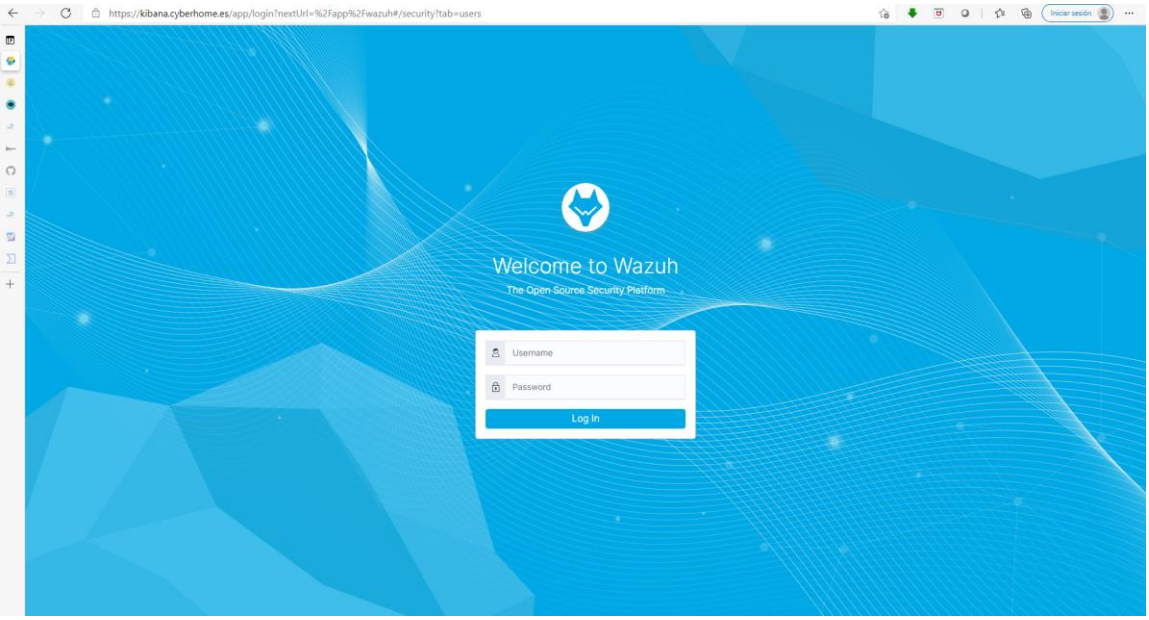


Ilustración 9 - Pantalla de Login Wazuh-Kibana

Como se puede ver en la imagen anterior, la pantalla de Login de kibana esta personalizada por Wazuh, una vez se introduce el Login y Password que se crea durante el proceso de instalación (en este caso en el Elasticsearch), dado que como se ha comentado visualiza información almacenada en dicho software y también opera contra el Wazuh Server a través de la API, para lo cual utiliza otro usuario este sí, dado de alta también durante el proceso de instalación en el producto Wazuh.

El look & feel de las pantallas puede variar según las versiones de los productos utilizadas.

Cuando se realiza el login en nuestro Wazuh-Kibana se realizan comprobaciones para ver si tiene conectividad con Elasticsearch, si están los índices necesarios, así como la conectividad antes mencionada y login contra el Wazuh-API Server.

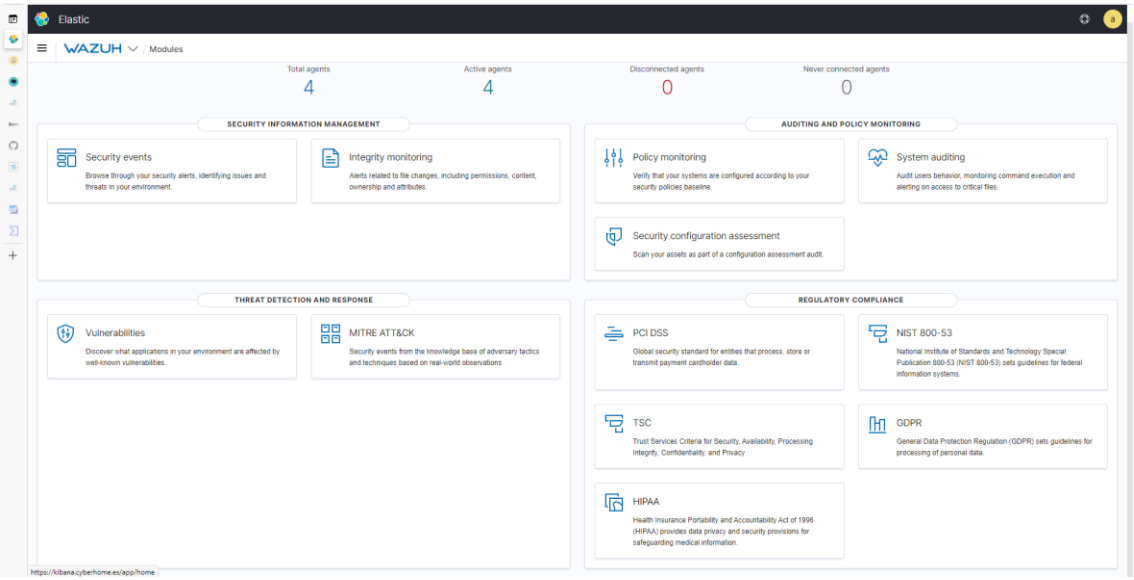


Ilustración 10 - Pantalla Wazuh Modules

La pantalla anterior muestra la página principal una vez autenticado con un usuario válido, donde se muestran los módulos de Wazuh, aunque reporta datos de funcionamiento, no deja de ser información de los propios contenedores donde están ubicados los diferentes componentes, no empieza a ser útil hasta que no tenemos agentes reportando información.

Por defecto no están activados todos los módulos, sin entrar en detalles, por ejemplo, el de vulnerabilidades es uno de los no activos, este comportamiento se puede modificar cambiando la configuración del agente, activando o desactivando los módulos que nos sean necesarios.

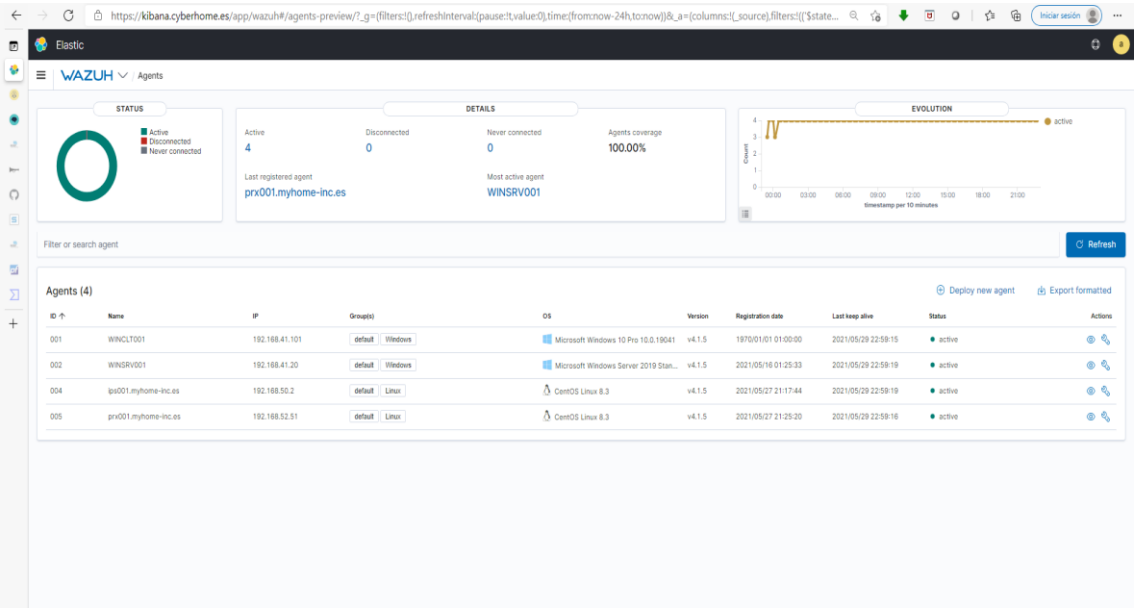


Ilustración 11 - Pantalla Wazuh Agents

En nuestro caso como se puede observar tenemos 4 agentes disponibles y conectados, dos sobre equipos Linux y otros dos sobre equipos Windows, se indica entre otras cosas la versión de SO y su estado. Una de las funciones de los agentes por ej. es obtener el inventario de software de los equipos gestionados.

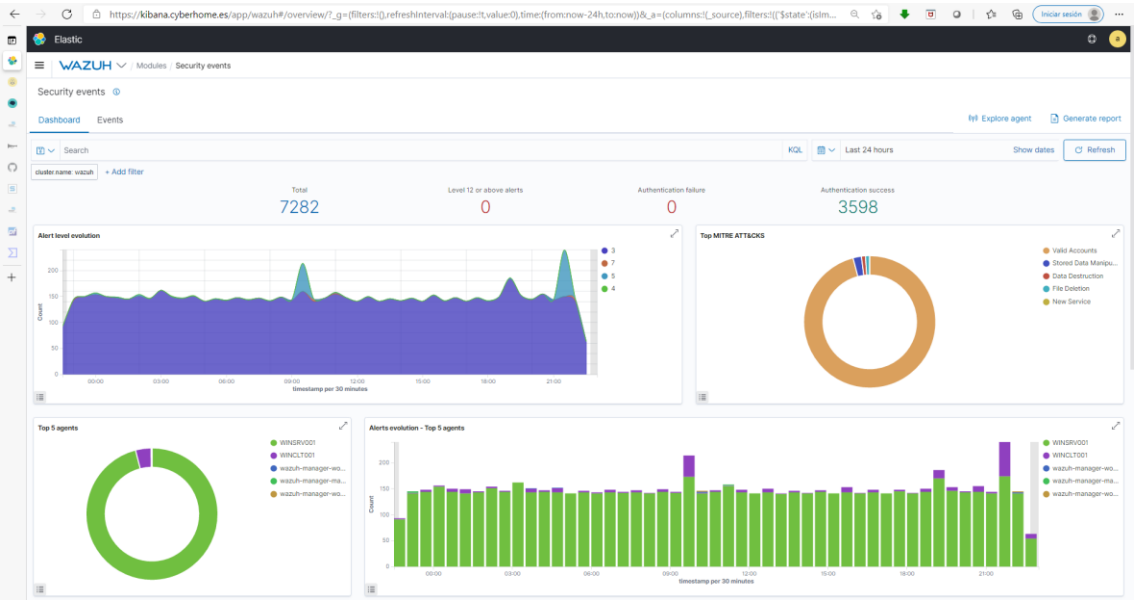


Ilustración 12 - Pantalla Wazuh Modulo Security events

En Security Events podemos ver que el agente que más información de eventos reporta con diferencia es el servidor Windows que es un Domain Controller, evidentemente no quiere decir que tengamos un problema, simplemente que tiene más actividad a nivel de eventos de seguridad.

Es trabajo del analista de seguridad, realizar las búsquedas y definir las alertas sobre los comportamientos que puedan constituir una amenaza o incluso producir un incidente.

Kibana permite visualizar información mediante dashboards o realizar búsquedas sobre los datos que contiene Elasticsearch, para Wazuh y sus módulos ya tiene unos dashboards creados, pero existen opciones para crear todos los dashboards que se necesiten, como ya se sabe pueden ser muy útiles, dado que las personas en general procesamos mejor la información de forma visual.

4.2 TheHive

Para TheHive nuestra herramienta de gestión de incidentes, la primera vez que entremos aparecerá una opción para actualizar la BBDD del producto, en nuestro caso Cassandra, para crear las estructuras necesarias y datos mínimos para poder empezar a trabajar con él producto.

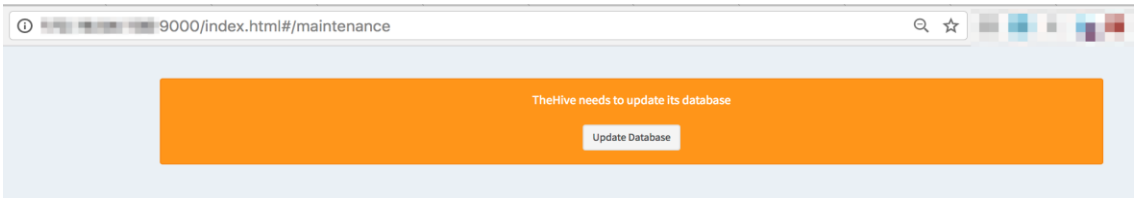


Ilustración 13 - Pantalla TheHive Maintenance

La pantalla anterior esta extraída de la documentación del producto, ya que no guardé un pantallo, como se puede observar es un acceso directo al producto TheHive, en nuestro caso accedemos a través de un Reverse Proxy NGINX para realizar el acceso usando TLS.

En nuestro caso, accedemos a nuestra consola vía web a través de la URL https://thehive.cyberhome.es lo que nos lleva a la pantalla de login del producto, la primera vez que se entra hay que utilizar el usuario y password por defecto que esta en la documentación de TheHive.

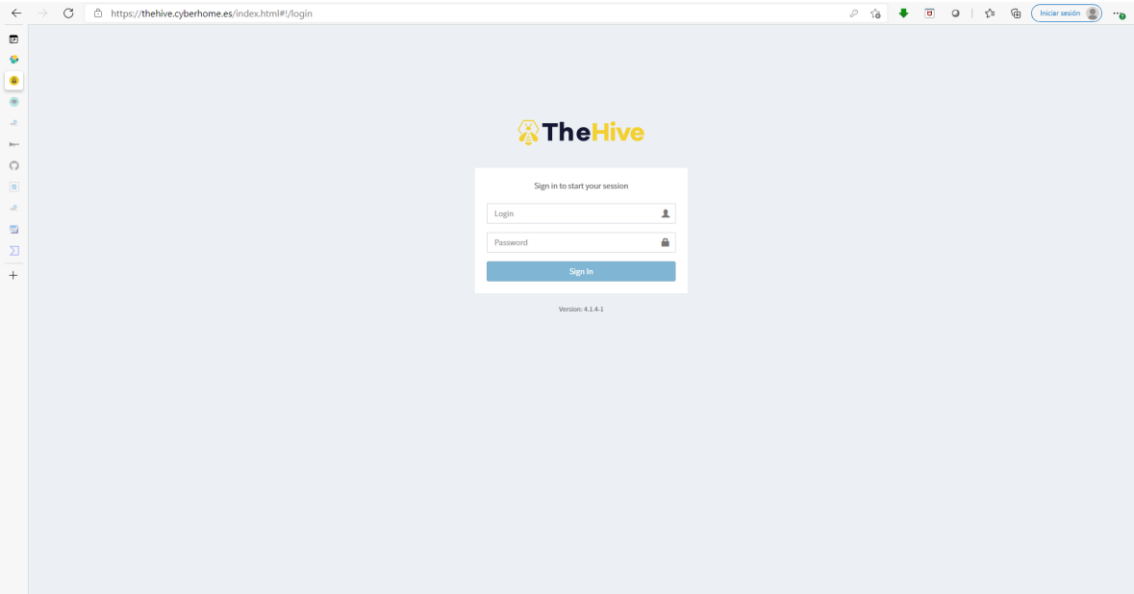


Ilustración 14 - Pantalla Login TheHive

Una vez nos autenticamos en la herramienta tendremos una organización por defecto que es la de Administración, podríamos utilizarla posiblemente, pero no tiene mucho sentido, de ella cuelga el usuario de administración con el que nos hemos autenticado en la aplicación y podemos (deberíamos) cambiar la contraseña por defecto para disponer de una más segura.

Lo siguiente debería ser crear una organización, en nuestro caso la empresa ficticia a la que vamos a ofrecer nuestros servicios de SOC o de CSIRT, en nuestro caso la empresa se llama MyHome Inc.

Solo es necesario darle un nombre y una descripción.

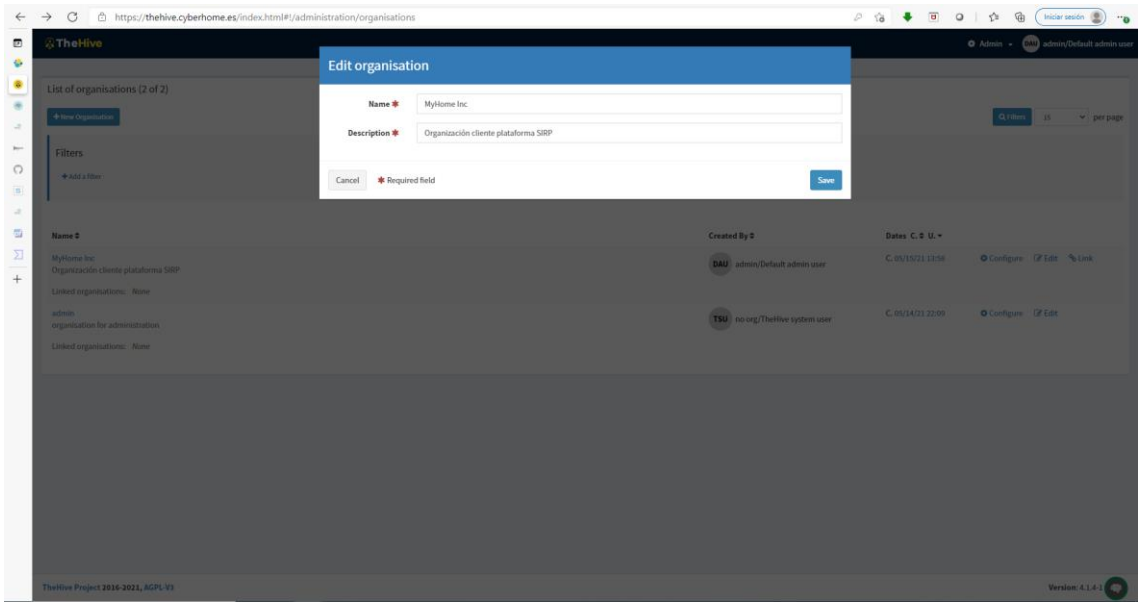


Ilustración 15 - Pantalla de creación Organización TheHive

Una vez creada la organización ya podemos crear usuarios dentro de dicha organización.

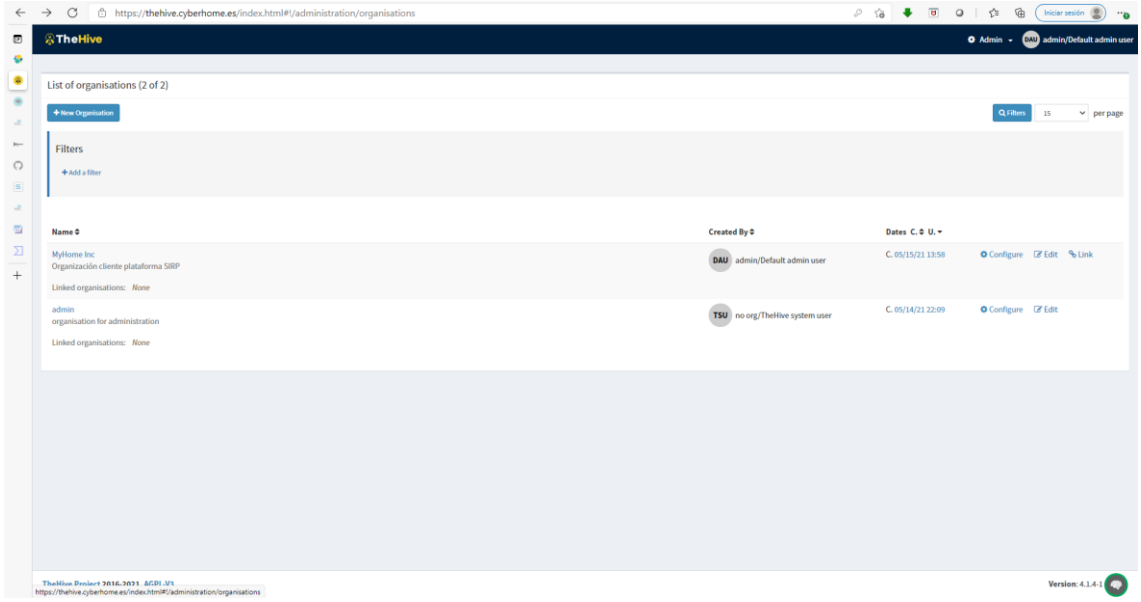


Ilustración 16 - Pantalla TheHive Organizaciones

Para ello basta con seguir el enlace que hay sobre el nombre de nuestra reciente nueva organización creada donde nos aparecerá la opción para hacerlo

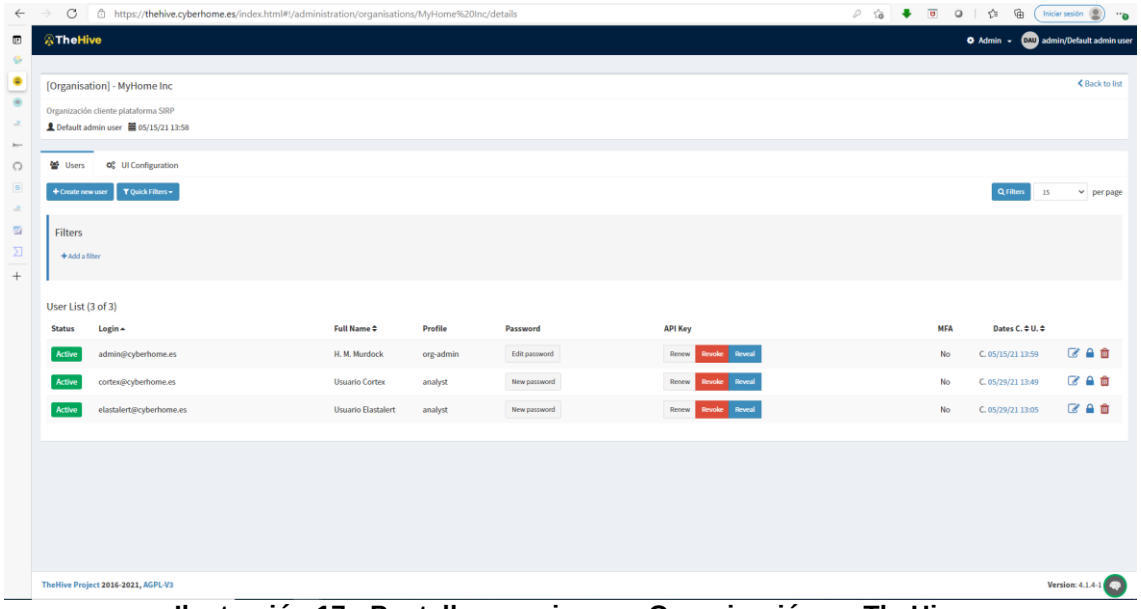


Ilustración 17 - Pantalla usuarios por Organización en TheHive

En nuestro caso hemos creado un usuario para administrar la Organización, se tienen que utilizar direcciones de correo electrónico para el nombre, así que hemos creado una primera cuenta admin@cyberhome.es con el rol orgAdmin y hemos creado dos usuarios más para cortex y elastalert que utilizaremos más adelante, a todos ellos les hemos habilitado una API Key, solo al usuario admin de la organización le hemos asignado contraseña ya que la necesitaremos para acceder a TheHive a esta organización.

Los usuarios para cortex y elastalert los he creado dentro de la organización dado que en este caso solo tengo una, pero podría tener sentido que se creen en la organización de administración a no ser que tengamos diferentes instancias para diferentes organizaciones. Son opciones de configuración que hay que tener en cuenta según la casuística que se quiera tratar.

Si esto no fuera un entorno simulado tendríamos que crear usuarios para todos nuestros analistas que contribuyan en nuestros casos y también se pueden utilizar perfiles de tipo read-only por ejemplo para auditores, directores, etc., perfiles que no participarán en los casos pero que pueden necesitar realizar consultas.

Una vez hemos creado este usuario salimos del usuario admin por defecto con el que estábamos autenticados en la herramienta “clickando” arriba en la esquina derecha sobre el nombre el usuario aparece la opción de Logout.

Antes de ello también vamos a la opción de UI Configuration, esto al igual que algunas de las acciones que estamos realizando es opcional, pero allí podemos cambiar el formato de fecha (por defecto viene en él formato MM/DD/YY y la he cambiado a DD/MM/YY que es como habitualmente se utiliza en nuestro país, pero como digo es cuestión de gustos), así como la opción al entrar en el Portal con nuestro usuario de organización, la pantalla de como se muestran los casos, he dejado la opción por defecto.

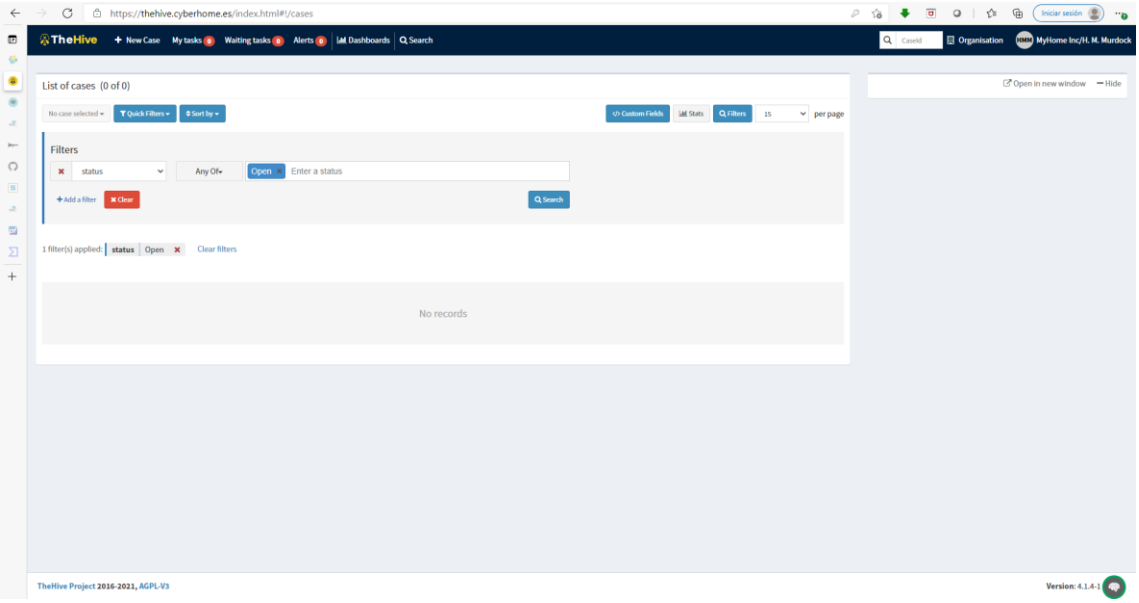


Ilustración 18 - Pantalla Cases en TheHive

Una vez autenticado con el usuario admin de nuestra nueva organización creada, tenemos una pantalla como la anterior en la que aparece la vista de casos, como no tenemos ninguno pues está vacía, se pueden aplicar filtros (para cuando tengamos muchos).

Arriba a la izquierda tenemos las opciones que usaremos habitualmente para trabajar con la herramienta:

- ir a esta página de inicio con el icono TheHive
- crear nuevos casos
- mis tareas asignadas
- tareas en espera
- alertas
- Dashboards (podemos crearlos o importarlos, pueden ser privados o compartidos)

Algunas de ellas las utilizaremos más adelante en nuestro caso de uso.

También tenemos un menú “Organization” arriba a la derecha junto a nuestro nombre de usuario autenticado en la aplicación. Desde este menú accedemos a la configuración de usuarios, templates, custom tags y UI Configuration.

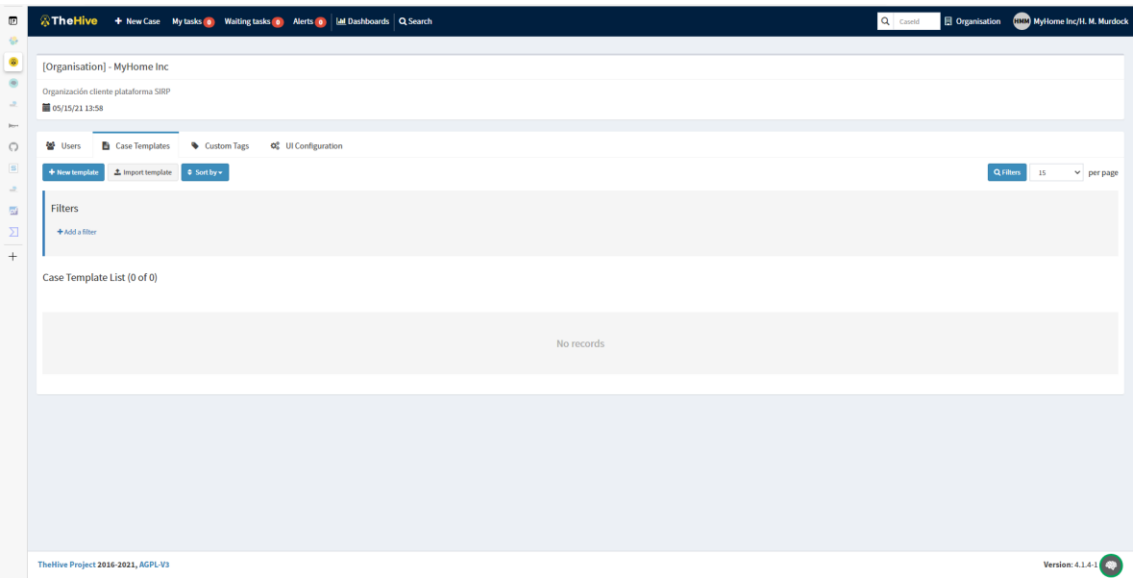


Ilustración 19 - Pantalla Case Templates en TheHive

La parte de usuarios ya la hemos visto anteriormente, así que no entraremos de nuevo, pero si puede ser interante las opciones de “Templates” en la cual podemos crear plantillas con campos personalizados para esta organización o importar plantillas ya confeccionadas por terceros (por ejemplo por la comunidad).

Finalmente, un detalle importante que enlaza con nuestro siguiente apartado, tenemos configurado TheHive para conectarse con nuestro Cortex, también podríamos conectarlo con MISP pero para esta demo de funcionalidad no lo hemos considerado necesario.

Para configurar el acceso de TheHive a Cortex o MISP se tiene que añadir al fichero de configuración application.conf la URL y API KEY de acceso a dichos componentes, añadiendo las secciones correspondientes tal como se informa en la documentación de TheHive.

En el caso de Cortex, nuestra imagen Docker permite configurar la conexión con Cortex mediante parámetros y es la opción que hemos seguido, por ello

como se puede ver en el siguiente pantallazo, abajo a la derecha tenemos un icono donde nos indica de color verde que tenemos conectividad con Cortex (en rojo cuando hay algún problema). También podemos verlo clickando sobre el usuario con el que estamos autenticados en la aplicación en la opción “About”, aparte de darnos las versiones de software podemos ver el OK en la conexión con Cortex.

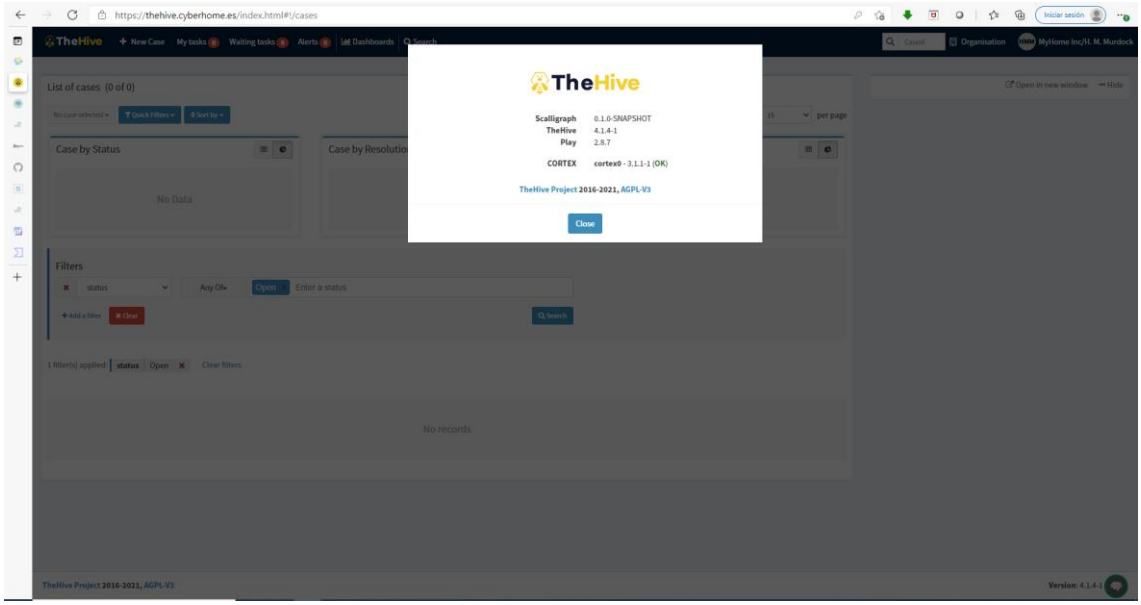


Ilustración 20 - Pantalla About y conexión TheHive - Cortex

4.3 Cortex

Una vez inicializada la BBDD como de forma similar a como se ha hecho en TheHive, en este caso dispone de una instancia de elasticsearch para almacenar sus datos ya podemos acceder a nuestro Cortex.

Accedemos a través de la URL https://cortex.cyberhome.es donde nos aparecerá ahora la pantalla de Login.

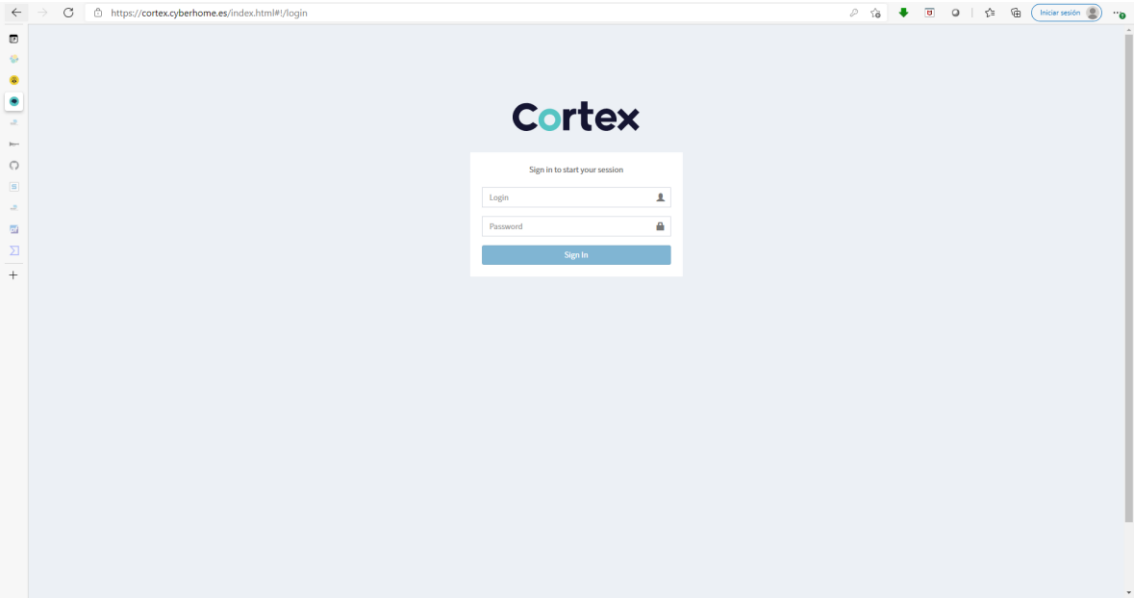


Ilustración 21 - Pantalla Login Cortex

Al igual que hemos hecho con TheHive, el primer paso será crearnos una Organización, en nuestro caso volvemos con la empresa ficticia MyHome Inc.

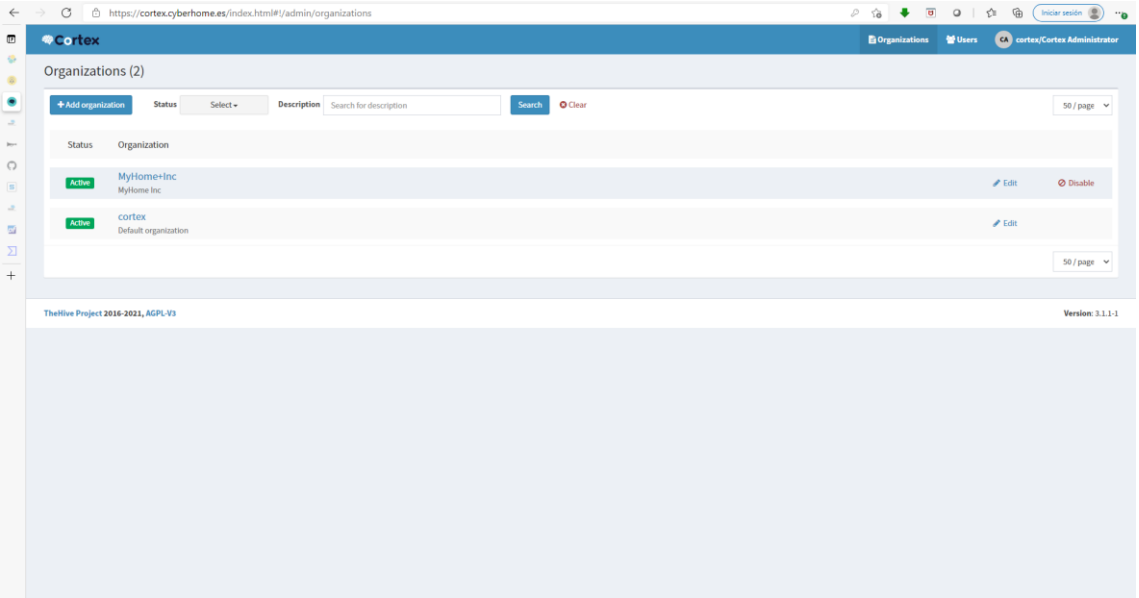


Ilustración 22 - Pantalla Organizaciones en Cortex

Ahora que tenemos la Organización creada, si clickamos sobre ella y seguimos el enlace nos aparecerán los usuarios, por defecto ninguno, pero vamos a crear dos usuarios un usuario con todos los roles que será nuestro administrador, se llama murdock, será nuestro usuario para interaccionar con el producto Cortex vía consola web.

También creamos un segundo usuario con los roles read y analyze para que nuestro TheHive pueda utilizar Cortex, en este caso no necesita contraseña, le bastaría solo con la API KEY que es lo que se ha de pasar como parámetro a Cortex para inicializar el contenedor o bien para configurarlo en el fichero application.conf (como ya se comentó en el apartador anterior).

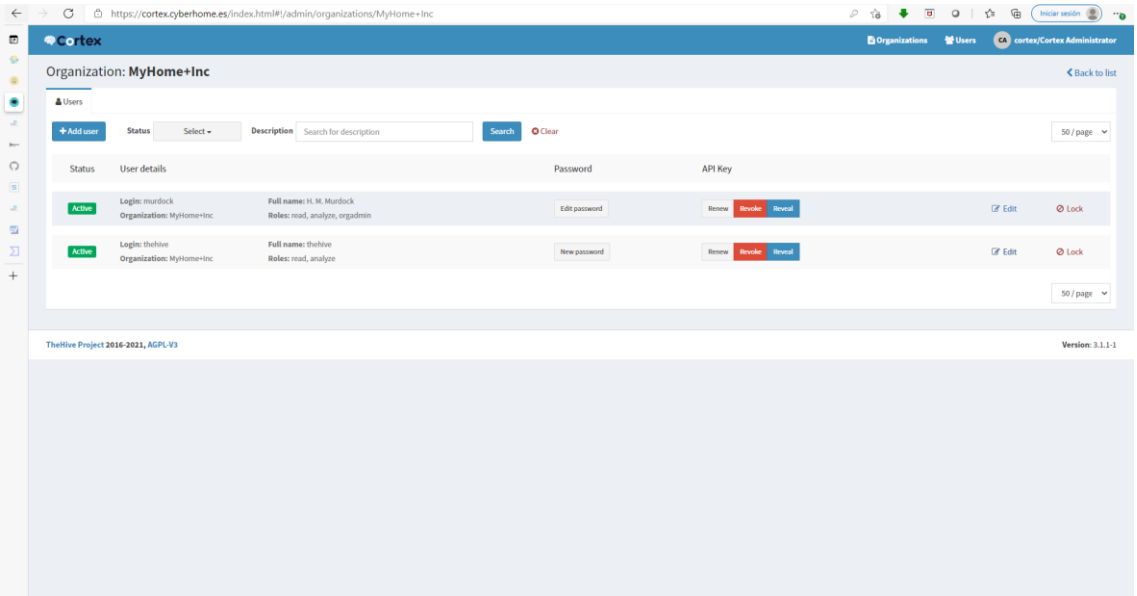


Ilustración 23 - Pantalla Usuario por organización Cortex

Ahora hacemos Logout y entramos en la aplicación con nuestro nuevo usuario administrador de Organización murdock, hay que configurar Cortex.

En Cortex tenemos dos tipos de Jobs, los analizadores (analyzers) y los de respuesta (responders), empezaremos por configurar los analizadores. En ambos casos hay servicios que son gratuitos y otros que son de pago para disponer de una API KEY que es lo que se necesita en la mayoría de los casos para poder configurar y activar los analizadores y algunos responders.

En mi caso, dispongo de algunos API KEY (por ej. SHODAN) y algunos otros los he dado de alta a modo Trial o Free Use, es decir, que tienen limitaciones de uso, pero dado que es para un entorno simulado el volumen de consultar no será alto en ningún caso, así que espero que sea suficiente.

Contra más fuentes de inteligencia y de mejor calidad sean más información fiable podremos obtener y por tanto podremos aportar información útil para los observables de nuestros casos en TheHive.

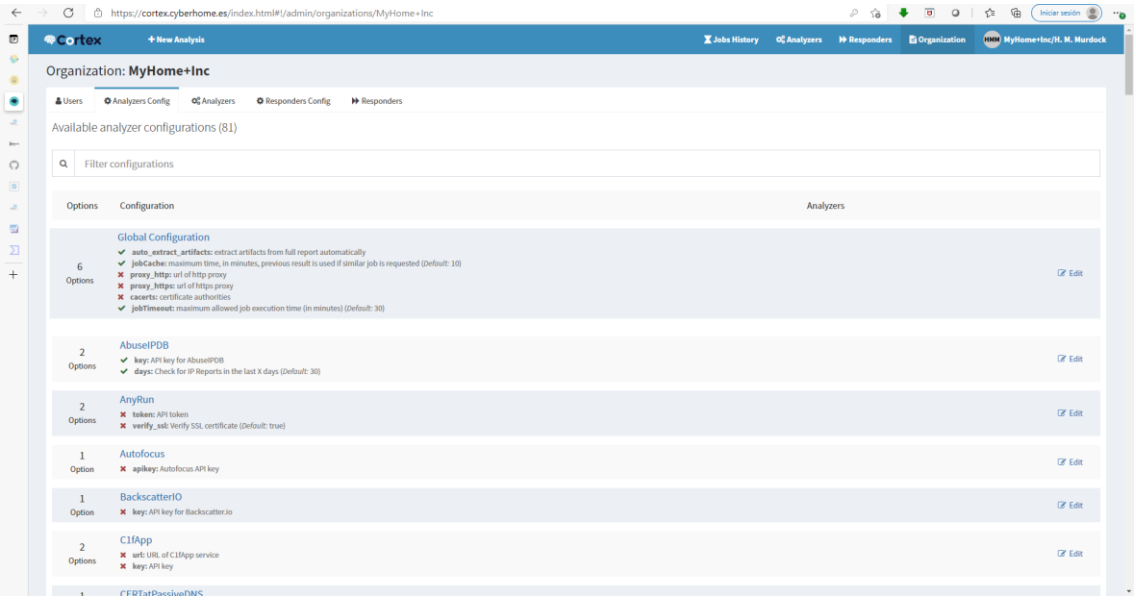


Ilustración 24 - Pantalla Configuración Analizadores Cortex

Si entramos en “Organizacion” podemos ver “Users”, ya lo hemos visto antes, así que vamos a “Analyzers Config” que es donde podemos configurar los analizadores, como se puede observar en la imagen hay hasta 81 disponibles (son las configuraciones disponibles, ya que hay varios analizadores que usan los mismos datos de configuración).

Como ya se ha comentado en nuestro caso activaremos algunos para ello basta con clickar en “Edit” y añadir la información solicitada en el formulario, de hecho en esta pantalla ya se puede ver por cada analizador que información nos va a solicitar con una X se muestra la información que no se ha suministrado y con una especie de V los valores suministrados.

Entre ellos tenemos uno que veremos en el siguiente apartado y es MISP, aquí es donde ponemos en juego nuestro componente de la plataforma SIRP.

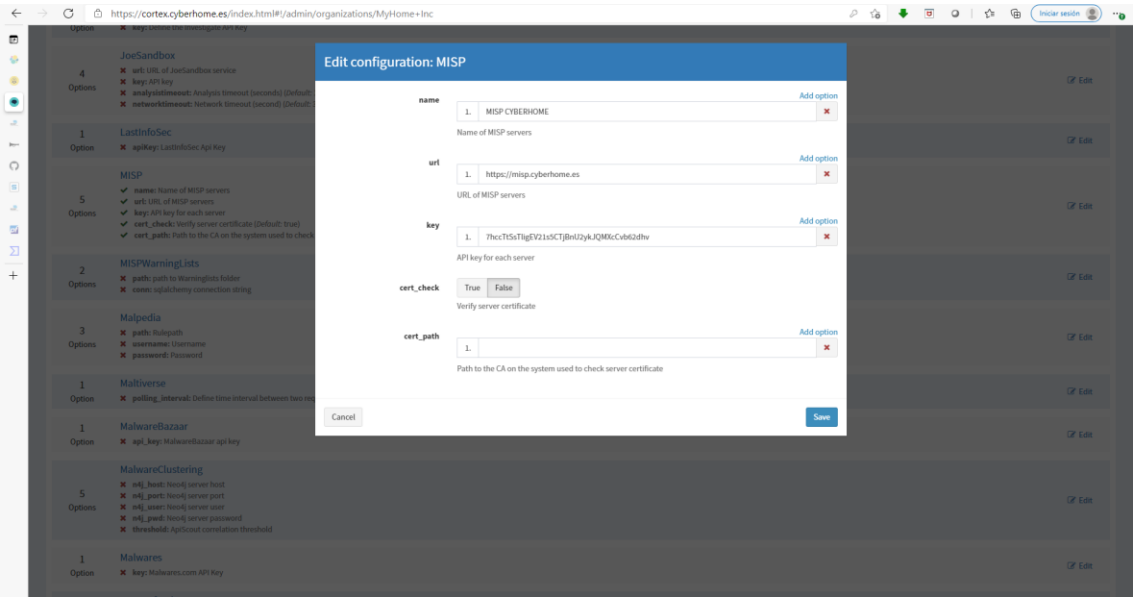


Ilustración 25 - Pantalla Configuración Analizador MISP en Cortex

Una vez configurados aquellos módulos o plugins que vamos a utilizar pasaremos a la siguiente sección “Analyzers”. Tenemos 164 disponibles (que hacen uso de las 81 configuraciones de parámetros comentada anteriormente).

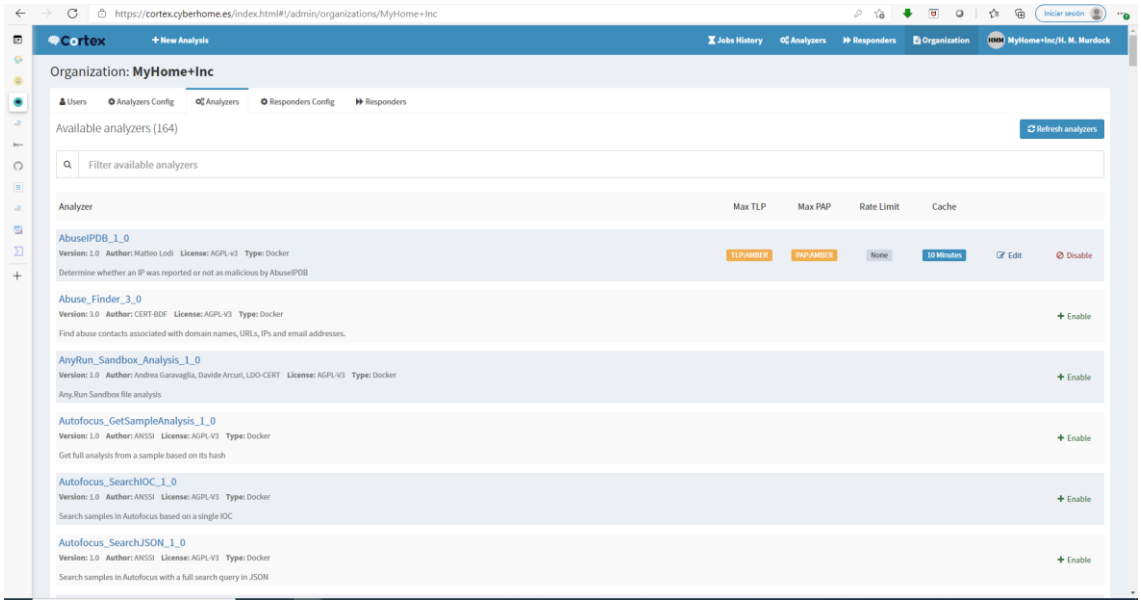


Ilustración 26 - Pantalla Activación/Desactivación Analizadores Cortex

Para activar basta con “clickar” sobre el “Enable” y aparecerá un formulario donde se solicitan datos de configuración, tiempo de caché, timeout para las petciones, si se descargan o no los observables, etc. En mi caso he dejado las opciones por defecto a esperas de realizar pruebas y ver si hay que modificar algún valor.

A continuación, podemos ver la configuración para nuestro MISP, como se puede observar en la imagen, aunque incompleta en la parte superior aparece los parámetros que ya hemos introducido anteriormente en la configuración y los nuevos parámetros comentados en el párrafo anterior.

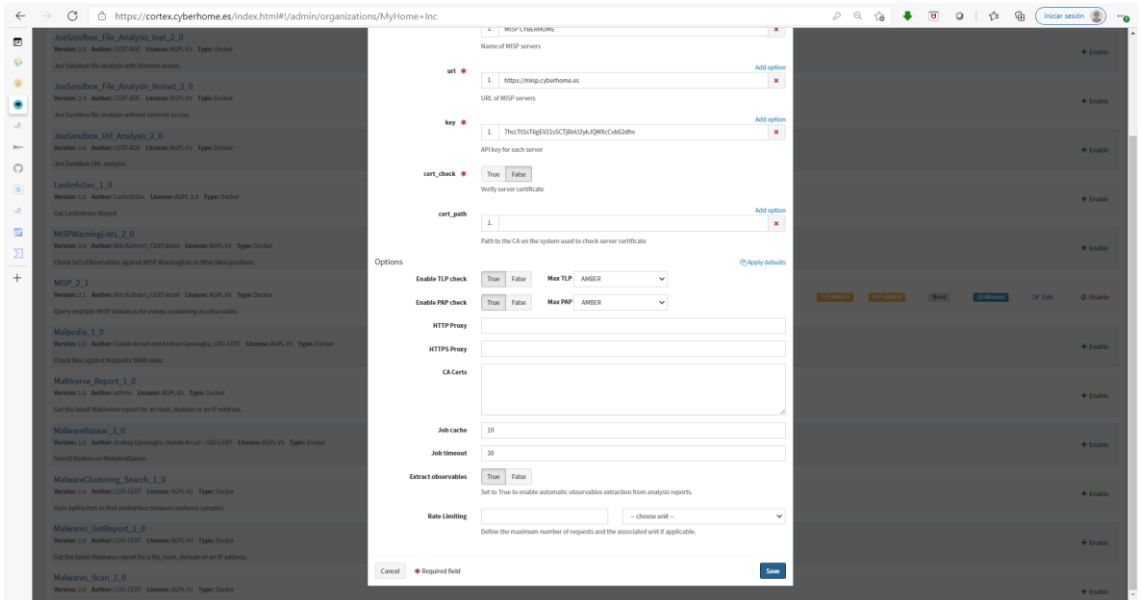


Ilustración 27 - Pantalla de activación MISP Analyzer en Cortex

En el momento de redactar este documento tengo activos los siguientes módulos de Responder, a la espera de confirmar algunas solicitudes de acceso a otros servicios:

- AbuseIPDB_1_0
- Shodan_DNSResolve_1_0
- Shodan_Host_1_0
- Shodan_Host_History_1_0
- Shodan_InfoDomain_1_0
- Shodan_ReverseDNS_1_0
- Shodan_Search_2_0
- SpamhausDBL_1_0
- TalosReputation_1_0
- TeamCymruMHR_1_0
- Threatcrowd_1_0
- TorBlutmagie_1_0
- TorProject_1_0
- URLhaus_2_0
- UnshortenLink_1_2
- Urlscan_io_Scan_0_1_0
- Urlscan_io_Search_0_1_1
- VirusTotal_GetReport_3_0
- VirusTotal_Scan_3_0
- Virusshare_2_0

Cada analizador se especializa en temas diferentes, algunos de reputación de IPs, nombres DNS, otros son para búsquedas de observables de correo electrónico, otros para la red TOR, otros para investigar URLs y otros para analizar ficheros de observables de posible malware, etc.

En general nuestros analizadores suelen ser programas escritos por lo general en Python, en nuestro caso algunos de ellos se ejecutan como contenedores Docker, esto es común tanto para analizadores como responders, que es lo que veremos a continuación.

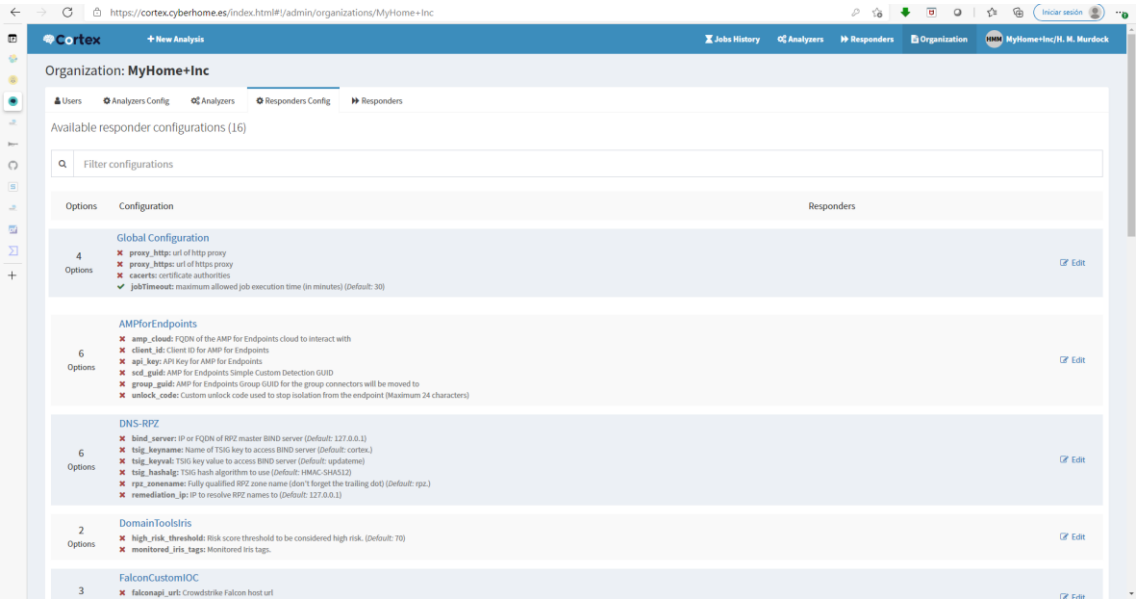


Ilustración 28 - Pantalla de configuración Responders en Cortex

Lo mismo, tenemos los apartados de configuración de parámetros que nos permitirán activar o no los responders, la dinámica es la misma que para los analizadores.

A nivel de responders el producto Cortex tiene muchos menos disponibles (actualmente 22), también es lógico porque depende de la infraestructura que disponga la empresa u organización a defender. Por lo que es un punto de mejora tanto para Cortex a nivel de soportar productos (firewalls, IPSs, etc.), es un reto para la comunidad y para los usuarios de este producto que como es lógico también pueden colaborar con sus desarrollos.

En nuestro caso tenemos activos los siguientes responders:

- Virustotal_Downloader_0_1
- Wazuh_1_0

El primero es bastante conocido por cualquiera que trabaje en el mundo de la seguridad, el segundo no es otro que nuestro Wazuh que como ya se ha explicado en este documento es un EDR, la R es de respuesta.

Así que se ha creado un usuario en Wazuh para este tipo de respuestas sin son necesarias, para ello vamos al Wazuh Kibana y creamos el usuario “cortex”.

En este caso y tratandose de un entorno de simulación se ha configurado con el rol admin, pero esto no sería la opción a escoger en un entorno real, dado que hay que restringir los permisos al mínimo necesario. En este caso me he tomado esta licencia, posiblemente hubiera bastado con un agent_admin, pero he querido asegurar que funcionaba en lugar de hacer pruebas para verificar si eso es suficiente.

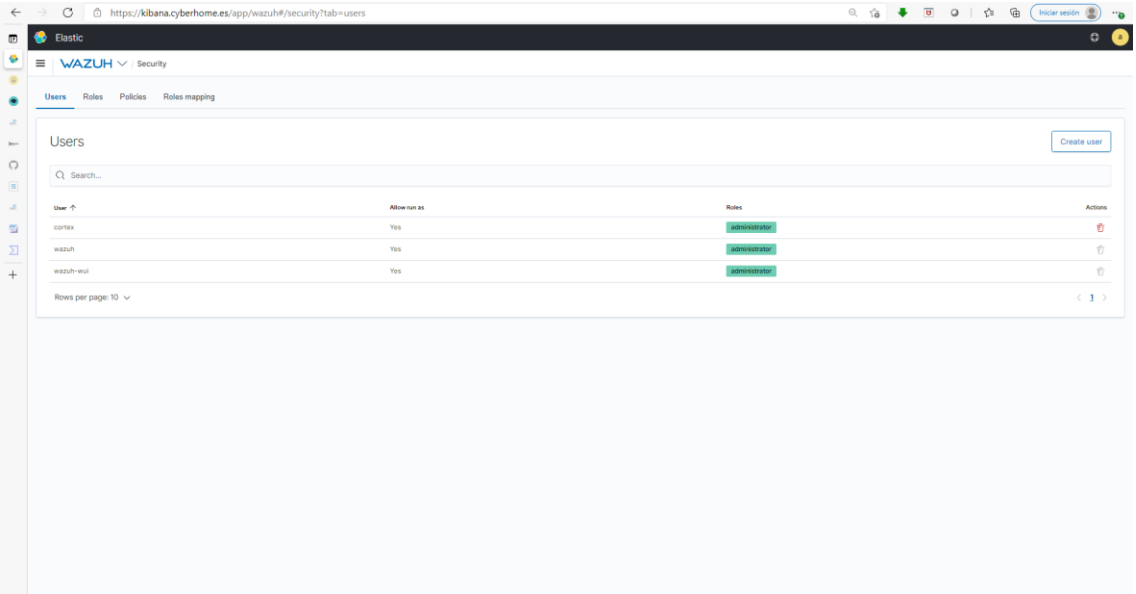


Ilustración 29 - Pantalla usuarios Wazuh

Para finalizar comentar que hemos tratado siempre a Cortex como el aliado de TheHive y lo es, pero también se puede utilizar de forma individual, en la parte superior izquierda podemos ver la opción “New Analisys” que nos permite mediante un pequeño formulario muy simple donde poder pasar los diferentes tipos de datos (observables) para ser analizados por nuestra herramienta Cortex.

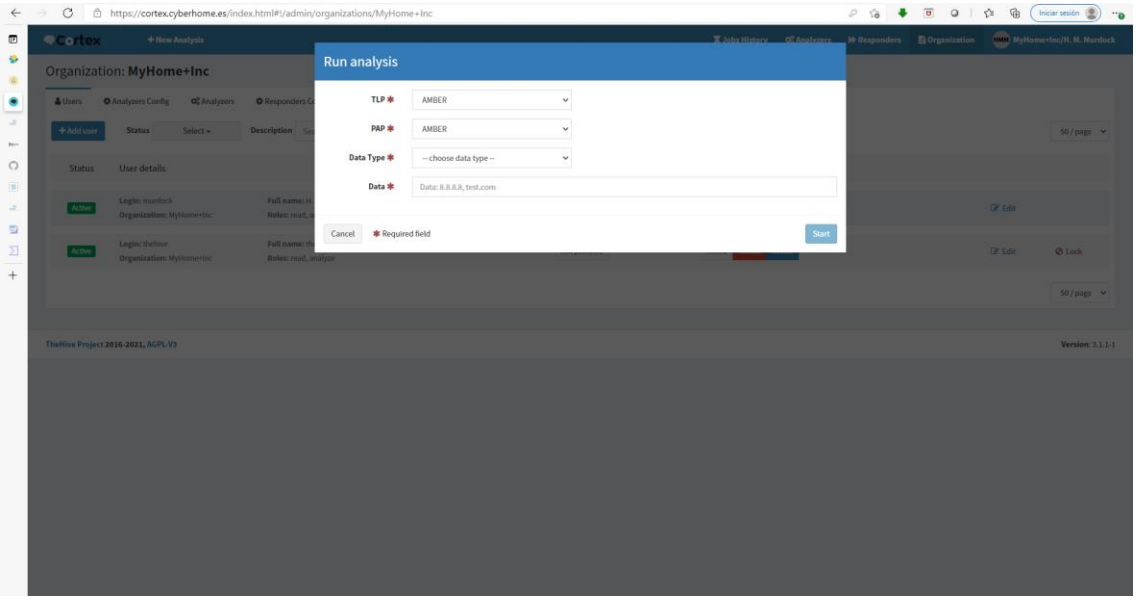


Ilustración 30 - Pantalla Run Analysis en Cortex

4.4 MISP

Una vez instalado el producto y operativo, tenemos un producto vacío como en los casos anteriores, no tiene información de ningún tipo, este producto es nuestra BBDD local de IOCs.

Procedemos como en los casos anteriores a acceder al servicio a través de su consola web siguiendo la URL: https://misp.cyberhome.es apareciendo la pantalla de Login del producto.

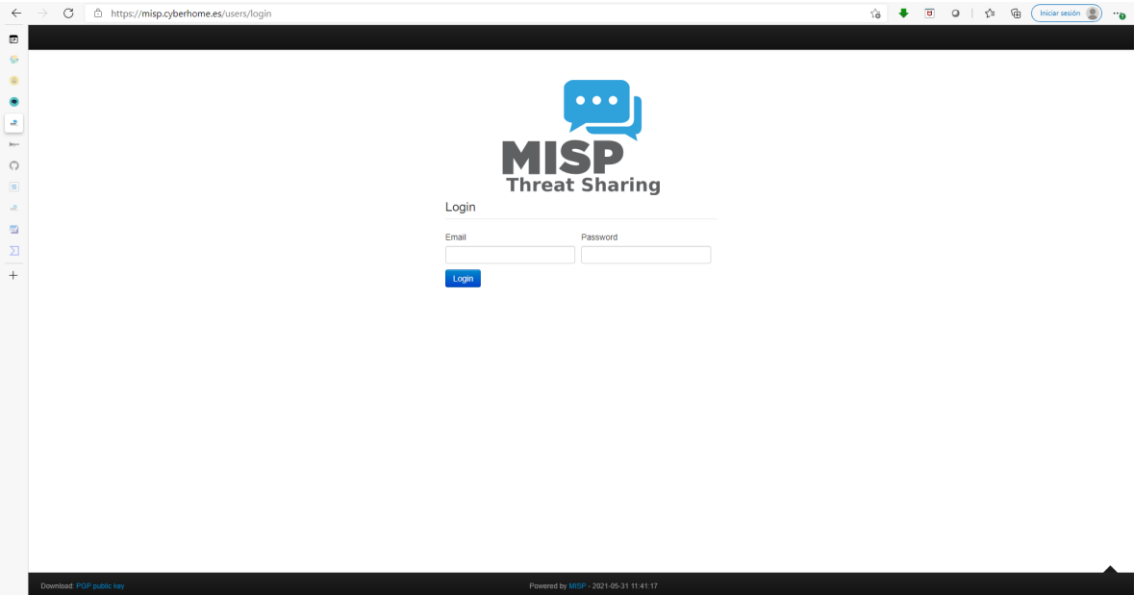


Ilustración 31 - Pantalla de Login de MISP

Accedemos con nuestro usuario administrador por defecto admin@admin.test y vamos a proceder como en los casos anteriores a crear nuestra organización ficticia, la empresa cliente a la que da servicio nuestra plataforma SIRP.

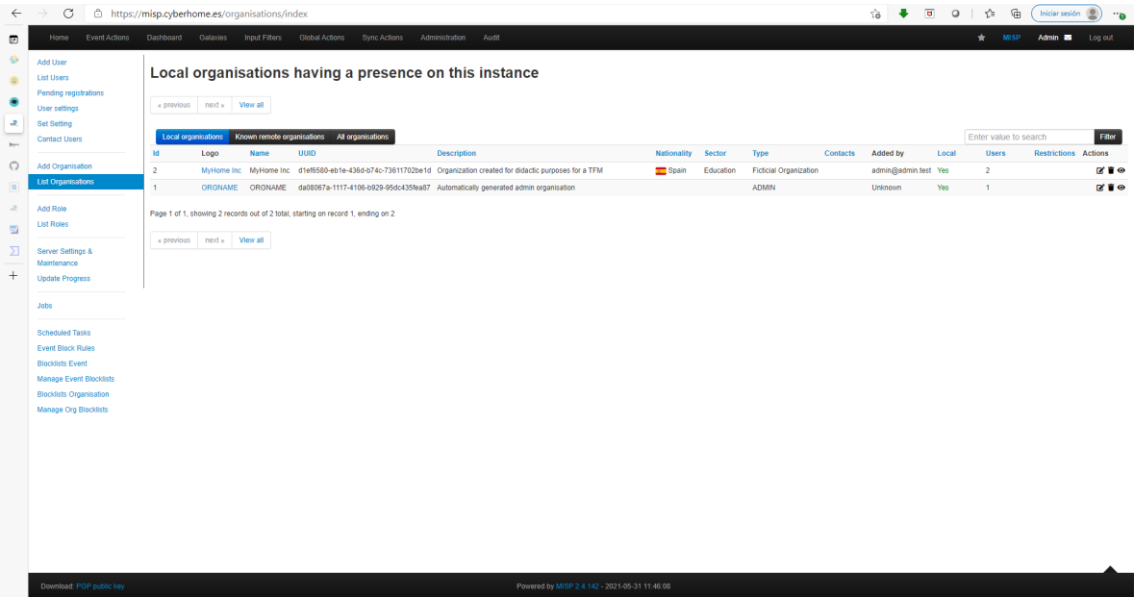


Ilustración 32 - Pantalla de Organizaciones en MISP

En esta pantalla que se ha mostrado tenemos nuestra organización por defecto y la que hemos creado MyHome Inc., para crearla basta con acceder al menú “Administration” y seleccionar “Add Organization” o bien clickar sobre Administration y se fija el menú en la margen izquierda de la pantalla donde entre otras aparece la opción indicada “Add Organization”.

Basta con rellenar algunos campos como el nombre, un identificado único para el cual se dispone de un botón de autogeneración, descripción de la organización, nacionalidad, sector, etc.

Esta aplicación dispone de muchas opciones de menú para realizar configuraciones y diferentes acciones sobre el producto, como es lógico aquí solo veremos unas pocas de dichas opciones que son a las que se ha dado uso en el transcurso de este TFM.

A continuación, como se ha hecho en los otros componentes vamos a crear un usuario de la nueva organización y un usuario para nuestro Cortex para que pueda trabajar con nuestra plataforma MISP vía API.

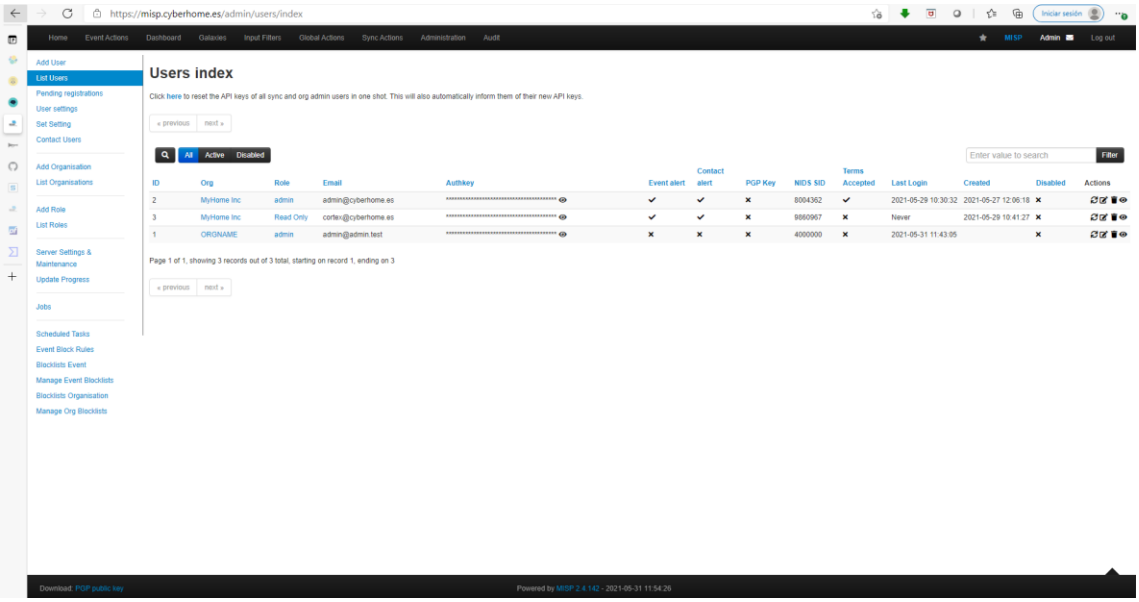


Ilustración 33 - Pantalla de Usuarios en MISP

Como en casos anteriores se requiere una dirección de correo, así que hemos usado admin@cyberhome.es para el administrador de la organización y cortex@cyberhome.es para el usuario con permisos de read-only (a priori deberían ser suficientes según se entiende la función de cortex).

Con el primer usuario nos autenticaremos en esta aplicación web con el usuario y contraseña, con el segundo usaremos su authkey que se encuentra en la definición del usuario, podemos verla y copiarla con la opción “Edit” o con la opción “View”.

Hacemos un “Logout” y nos conectamos a la aplicación MISP con nuestro nuevo usuario admin@cyberhome.es.

Después de esto seguimos teniendo una BBDD vacía, así que ahora tenemos que hacer que tenga contenido y para ello vamos a ver nuestra lista de “Feeds”.

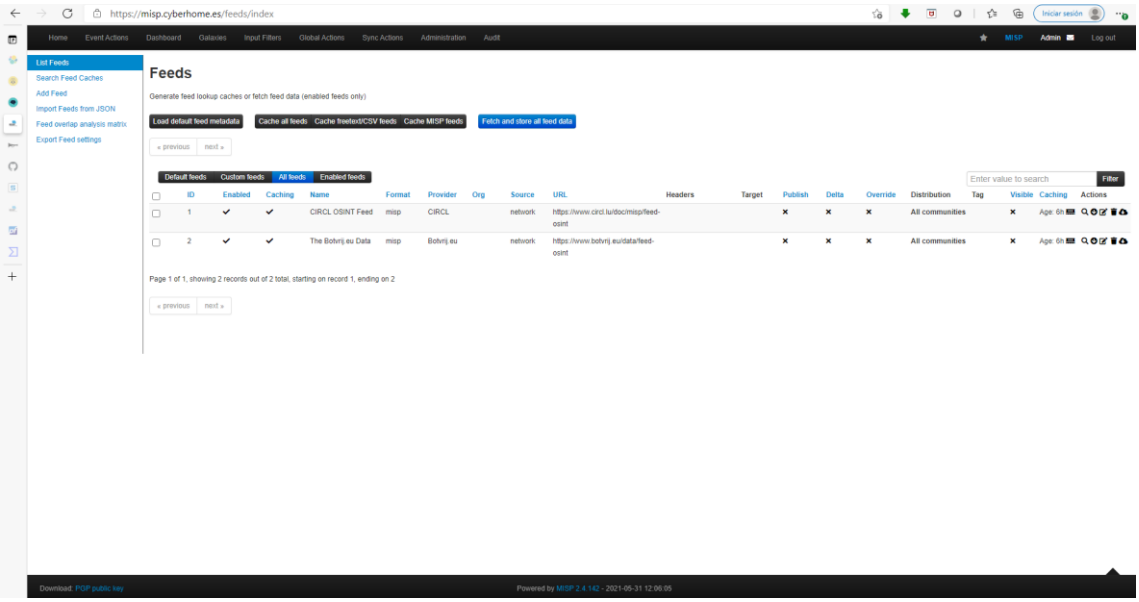


Ilustración 34 - Pantalla Listado Fuentes Feeds en MISP

Por defecto vienen estos dos dados de alta en el producto MISP, al menos en la instalación que yo he realizado utilizando el repositorio MISP de Github. Para verlos y activarlos (por defecto vienen inactivos) hay que ir a “Sync Actions” y “List Feeds”. En las actions de la derecha encontraremos la opción de activar.

Una de las funciones de MISP y que parece que ha alcanzado es conseguir un estándar para la compartición de información, en la página [MISP Default Feeds (misp-project.org)](#) del proyecto MISP podemos encontrar una lista de Feeds que podemos añadir a nuestra instalación.

Aparte de esta lista de Feeds, tendría sentido como plataforma de compartición de información que nos conectemos e intercambiemos información con otras plataformas MISP de nuestro entorno, ya sean de organizaciones de nuestro sector u organismos oficiales CERT.

Estos feeds se pueden descargar en el momento como se puede ver por los actions que hay al lado de cada fuente de eventos de la lista, pero hay que activar que estas actualizaciones se hagan de forma periódica para estar actualizados con la información que se mueve por el mundo.

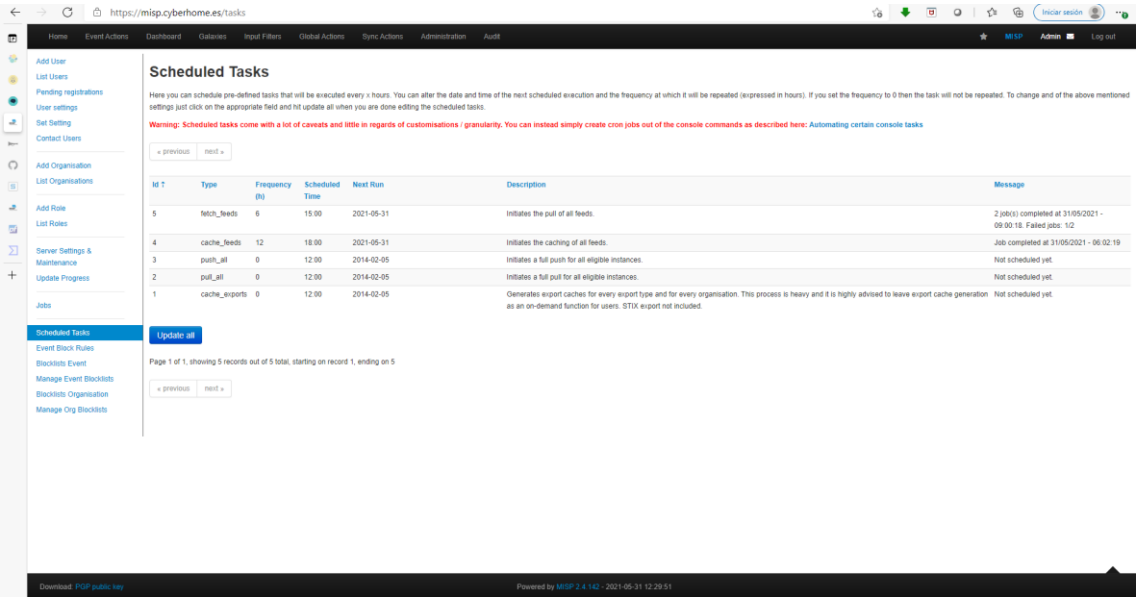


Ilustración 35 - Pantalla Programación de Tareas en MISP

En el menú de “Administration” tenemos “Schedules Tasks” para programar la actualización de nuestro feeds (los de la lista anterior) y de la caché que se mantiene dentro de nuestro MISP.

Se puede programar la frecuencia de ejecución y la hora a la que se ejecuta la primera vez (después lo marca la frecuencia), el campo es configurable “clickando” sobre él. El “Next Run” es la próxima ejecución, aunque en mi opinión el hecho de que solo muestre la fecha lo hace poco útil, dado que para saber la hora hay que hacer el cálculo de la hora inicial y la frecuencia configurada.

En nuestro caso, podemos ver que se ha completado una de dos. Tenemos la opción de ver cómo ha estado trabajando esto accediendo a la opción “Jobs”.

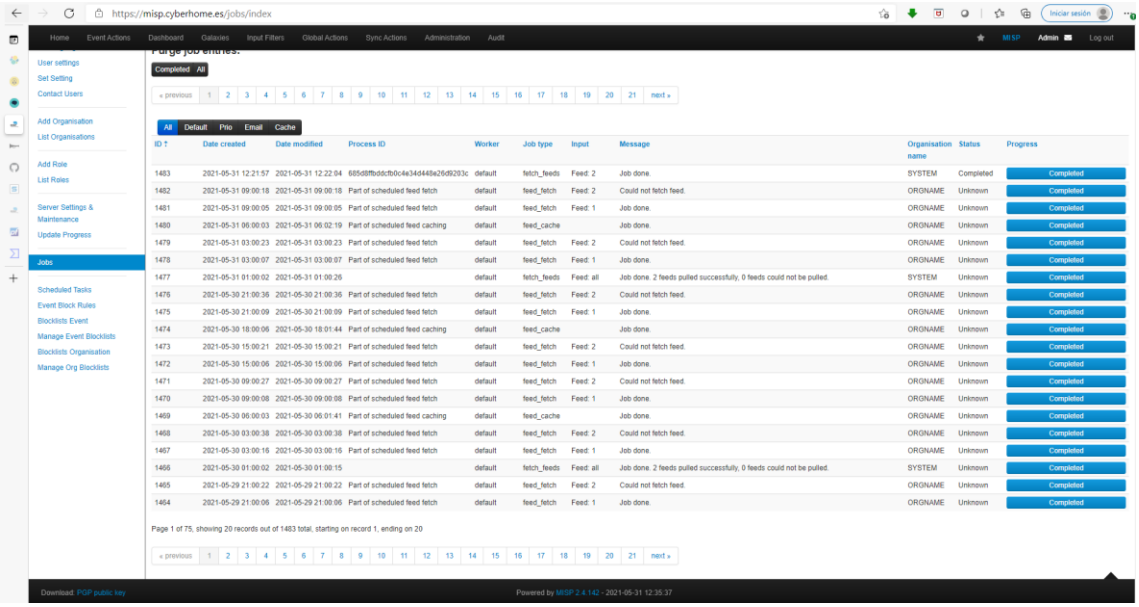


Ilustración 36 - Pantalla Jobs en MISP

Al parecer la ejecución sobre el Feed 1 parece que funciona correctamente, pero para el Feed 2 (número asignado en la lista de “Feeds”) siempre falla la descarga (tema a revisar, si hay algún problema con la URL o el acceso, etc.), aunque aparentemente con una descarga manual que es la primera línea de la pantalla anterior parece que ha funcionado (o como mínimo no ha dado ningún error).

Hablando de errores, creo que es interesante también revisar la opción “Server Settings and Mantenance” del mismo menú de “Administration”.

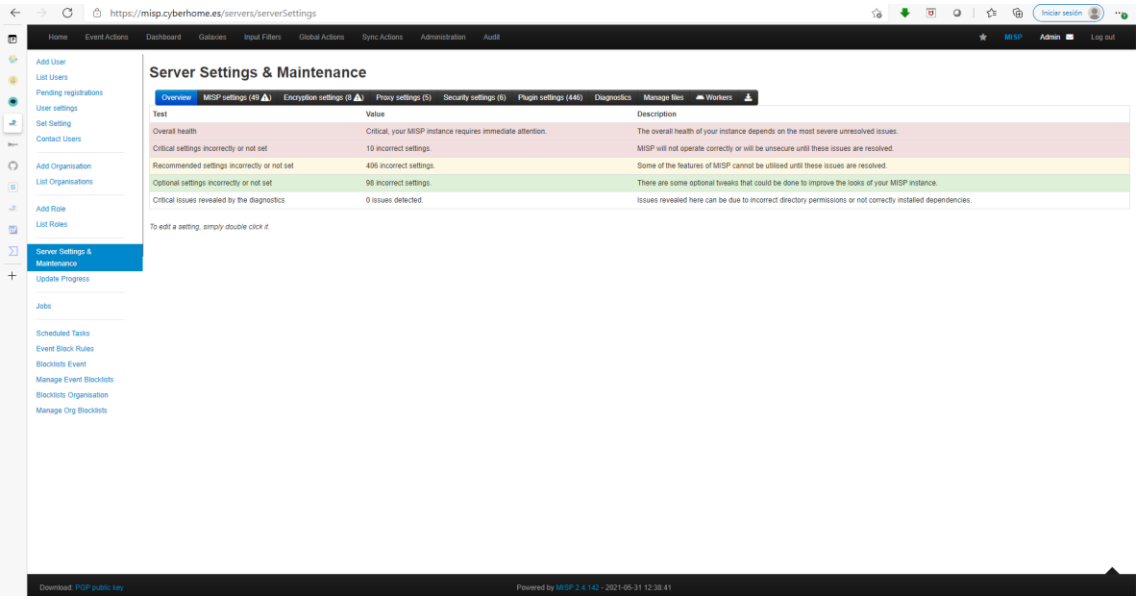


Ilustración 37 - Pantalla Server Settings & Maintenance MISP

Es como una opción de autodiagnostico del software MISP y también se permite hacer modificaciones de algunos parámetros desde estas pantallas, lo ideal sería tenerlo todo en verde y he dedicado algo de tiempo sobre todo a reducir los “Critical Settings” en la mayoría de casos suele ser porque algún parámetro no se ha seteado y se ha dejado por defecto (ej. el idioma).

Por último, para finalizar este apartado, también comentar como ya se ha dicho anteriormente que se trata de compartir información con nuestro entorno (es la forma de contribuir a nuestra protección y a la de los demás), en esta herramienta podemos añadir nuestros propios eventos (campos y otras configuraciones).

Esta pantalla que se muestra a continuación la podemos ver a través del menú “Event Actions” en la opción “Lista de Eventos”, esta es la información que se ha descargado de nuestras fuentes de Feeds.

Como se puede observar los eventos tienen unas etiquetas que son las que tienen asignadas por terceros, pero también podemos añadir nuestros propios eventos con información de IOCs de nuestra organización, podremos usar estas etiquetas ya creadas si tiene sentido (podemos aparte de verlas asignadas a los Eventos cargados, verlas en “List Tags”) y podemos crear las

nuestras para ver claramente aquellos datos que hemos introducido desde nuestra organización.

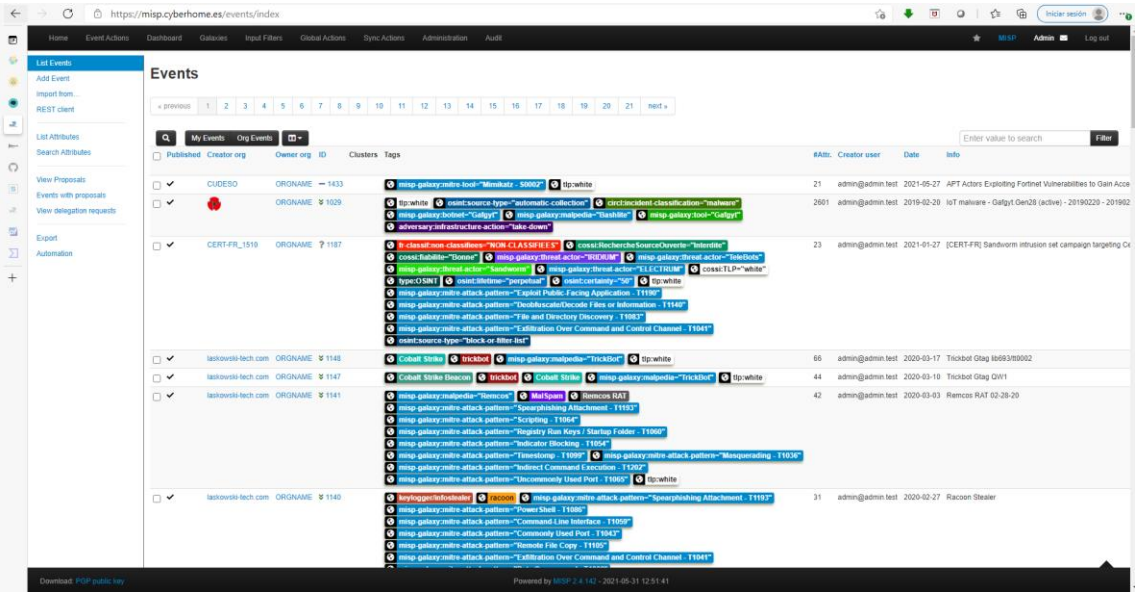


Ilustración 38 - Pantalla Listado de Eventos MISP

Para ello podemos desde el menú “Event Actions”, donde esta opción “List Tags” también podemos hacer “Add Tags”, en nuestro caso añadimos una etiqueta, pero puede tener sentido añadir más según queramos tener bien localizados nuestros eventos, se ha añadido “Manual” para cuando un analista añada información a nuestro MISP de forma manual.

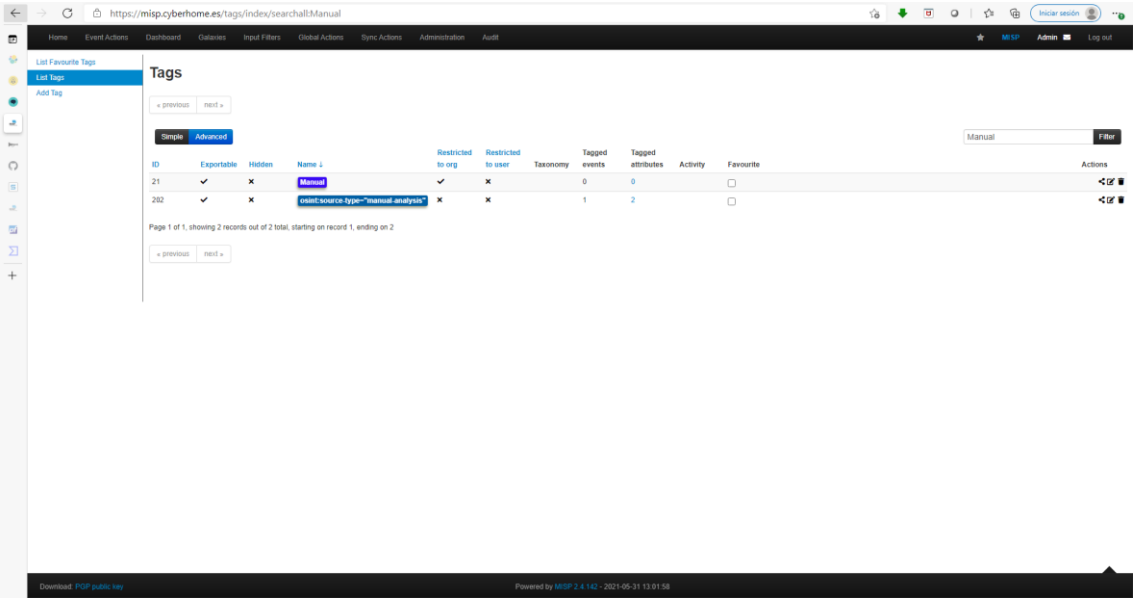


Ilustración 39 - Pantalla Listado Etiquetas MISP (filtrado)

Aquí las tenemos filtradas, ya que hay miles de etiquetas.

4.5 Praeco-Elastalert

Aquí tenemos dos componentes que una vez instalados ya están integrados entre ellos de forma similar a como teníamos la integración entre Wazuh – Elasticsearch y Kibana, salvando las diferencias.

En este montaje Elastalert sería el backend y Praeco un frontend vía web, dado que Elastalert funciona a nivel de ficheros para generar las reglas (alertas).

Elastalert se configura para hacer consultar sobre nuestro Elasticsearch (esta vez sí, el de Wazuh, no tendría sentido trabajar con uno propio que no tendría datos). Elastalert también guarda sus metadatos y auditoría en el propio Elasticsearch para lo cual también genera sus propios índices que están en el fichero de configuración que se suministra durante la instalación (al igual que el usuario/password o el token de usuario) que son las opciones que se pueden usar para acceder a Elasticsearch.

Esto sería en cuanto a la entrada, respecto a la salida de las alertas son múltiples, permite trabajar con JIRA, con Slack, un simple correo y por supuesto y en este caso lo que nos interesa con TheHive (entre otras opciones) para hacer llegar la información a los analistas de seguridad de la organización.

Praeco como ya se ha comentado es la “interface” web que permite hacer la configuración más sencilla de las alertas mediante formularios, además de poder visualizar los metadatos de elastalert.

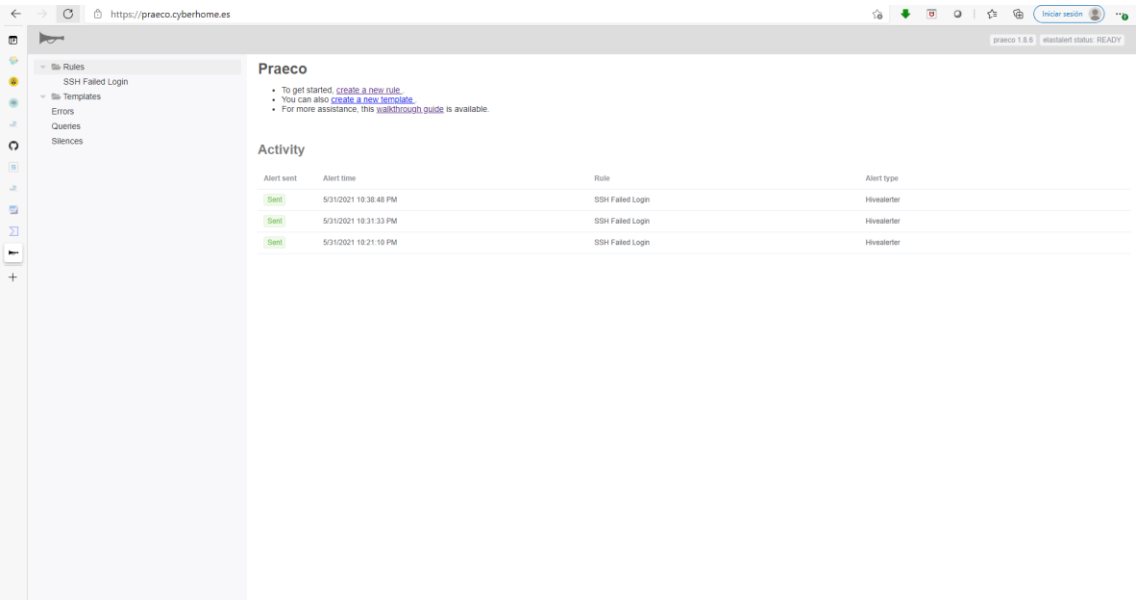


Ilustración 40 - Pantalla principal de Praeco

Esta es la “pinta” que tiene Praeco, como se puede comprobar es bastante austera, hay que decir que realiza las funciones estrictamente necesarias a diferencia de los otros componentes que tienen un look&feel más desarrollado.

Otro defecto que tiene el producto es que permite el acceso a cualquiera que tenga visibilidad de la URL dado que no implementa autenticación, lo cual

podría ser un poco peligroso dado que permite hacer consultas sobre nuestro Elasticsearch que contiene información sensible sobre la organización, para paliar este defecto en mi opinión bastante grave, en la configuración de NGINX aparte de darle cifrado TLS, al menos se le ha añadido un basic authentication.

Como se aprecia en la consola web de Praeco tiene un menú en la parte izquierda de la pantalla, donde se pueden ver como dos carpetas “Rules” y “Templates” de entrada estas están vacías, una vez instalado salvo que importemos contenido de la comunidad no hay ningún tipo de regla.

Para esta prueba hemos generado una “Rule” llamada “SSH Failed Login” que realiza una consulta a Elasticsearch y si encuentra más dos Failed Login en conexión SSH (es una prueba el número es muy bajo para generar una alerta real) pues nos genera una alerta en nuestro TheHive que podemos tratar añadiéndola a un caso o el tratamiento que creamos más conveniente.

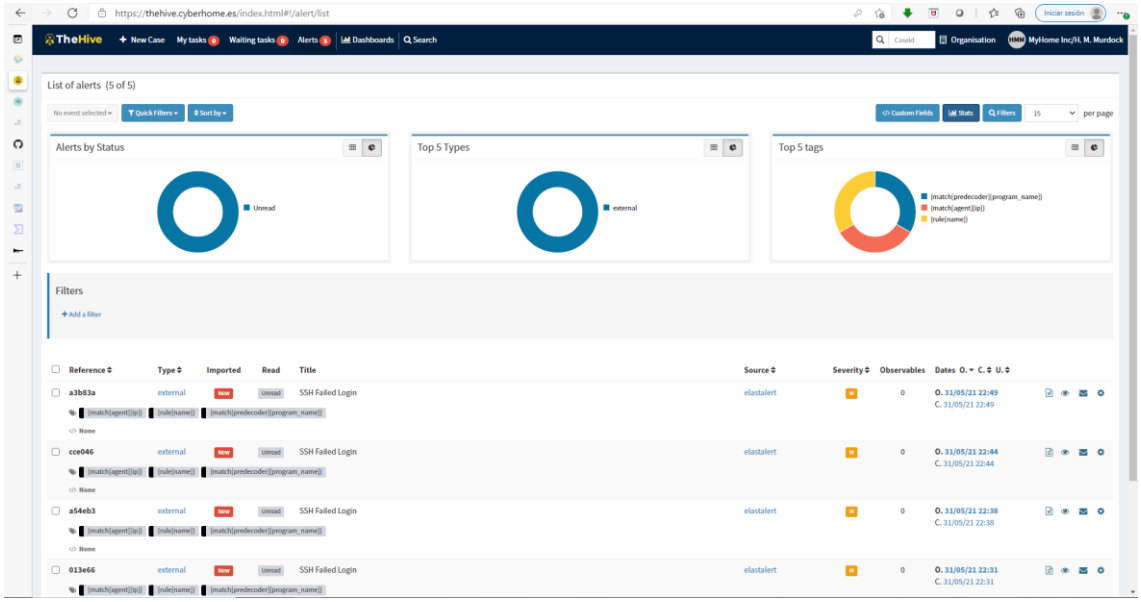


Ilustración 41 - Pantalla de Alerts TheHive (alertas creadas por Elastalert)

Para generar alertas útiles hay que jugar con los contadores de eventos, la frecuencia, etc. etc., toda una serie de parámetros que como indico son los que le dan utilidad o no a una alerta, se trata de evitar en la medida de lo posible los falsos positivos y detectar conductas que constituyan amenazas reales y/o posibles incidentes.

Volviendo a Praeco a continuación se muestra la pantalla donde está definida la regla que a su vez define cuando se ha de generar una alerta. Para ello se indica el índice que se ha de monitorizar y una especie de lenguaje SQL nos permite generar consultas, determinar que campos visibilizar, el título de la alerta, etc.

Para cada una de las reglas estos datos se pueden diferenciar e incluso se podría diferenciar la fuente, podríamos usar diferentes Elasticsearch si fuera el caso, por defecto utiliza el que tenemos en el fichero de configuración, pero por

cada regla se podría llegar a indicar otras fuentes Elastic (aunque para ello ya habría que hacer la configuración a mano, desde Praeco no es posible).

En nuestro caso la configuración de TheHive también va sobre un fichero ubicado en el directorio “Rules” y que es un tanto especial, permite guardar allí la URL y credencias de acceso y luego puede ser utilizado por todas las reglas, usa la API KEY del usuario “elastalert” que vimos en el apartado de configuración de TheHive.

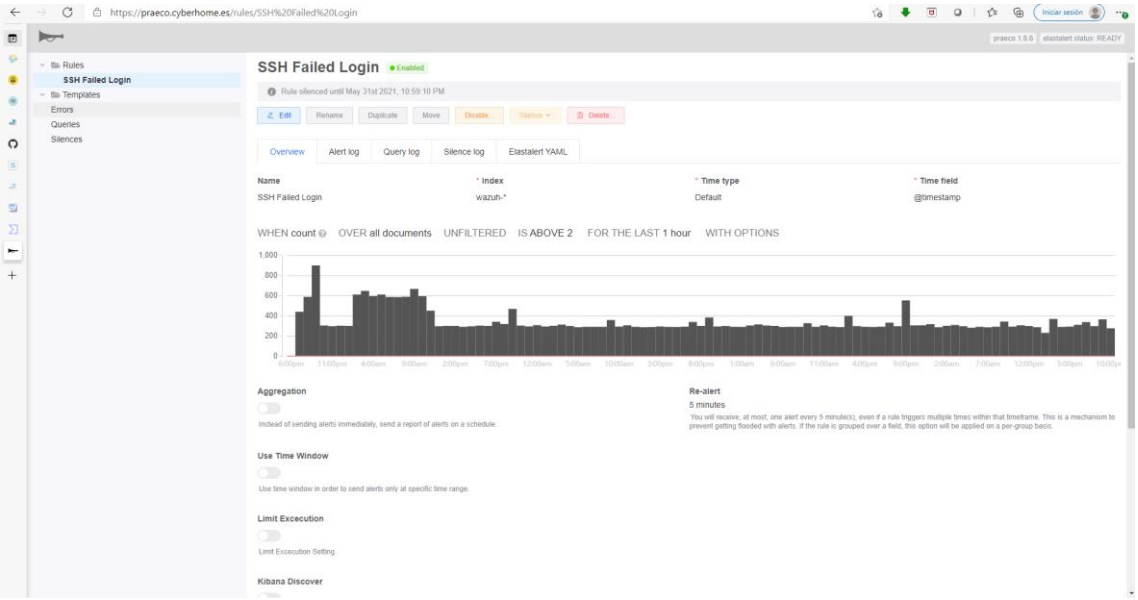


Ilustración 42 - Pantalla Praeco Regla (definición Alerta)

También es necesario crear un usuario en nuestro Elasticsearch y asignarle permisos para que pueda crear y mapear los índices necesarios para usar con Elastalert y poder consultar los datos sobre los que se implementan las alertas.

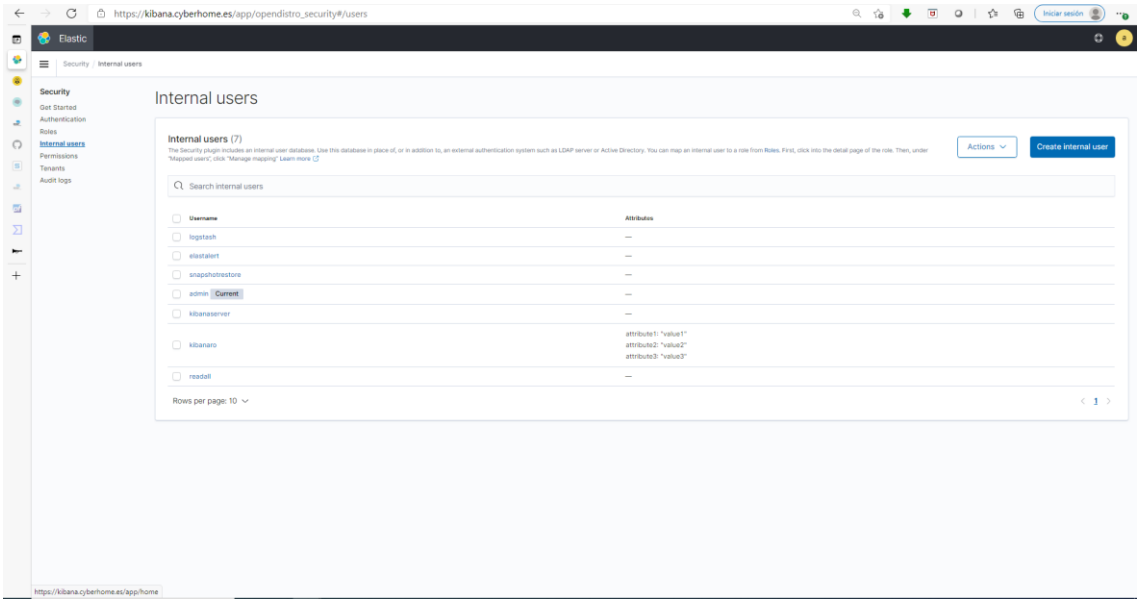


Ilustración 43 – Pantalla usuarios Elasticsearch (usuario Elastalert)

En Internal Users creamos otro usuario “elastalert” de forma similar a como hicimos en TheHive, aunque aquí le asignaremos usuario y contraseña.

Todo esto como se puede observar lo hacemos vía web a través de nuestro Wazuh-Kibana desde donde también podemos crear un rol para asignarle los permisos necesarios a este nuevo usuario.

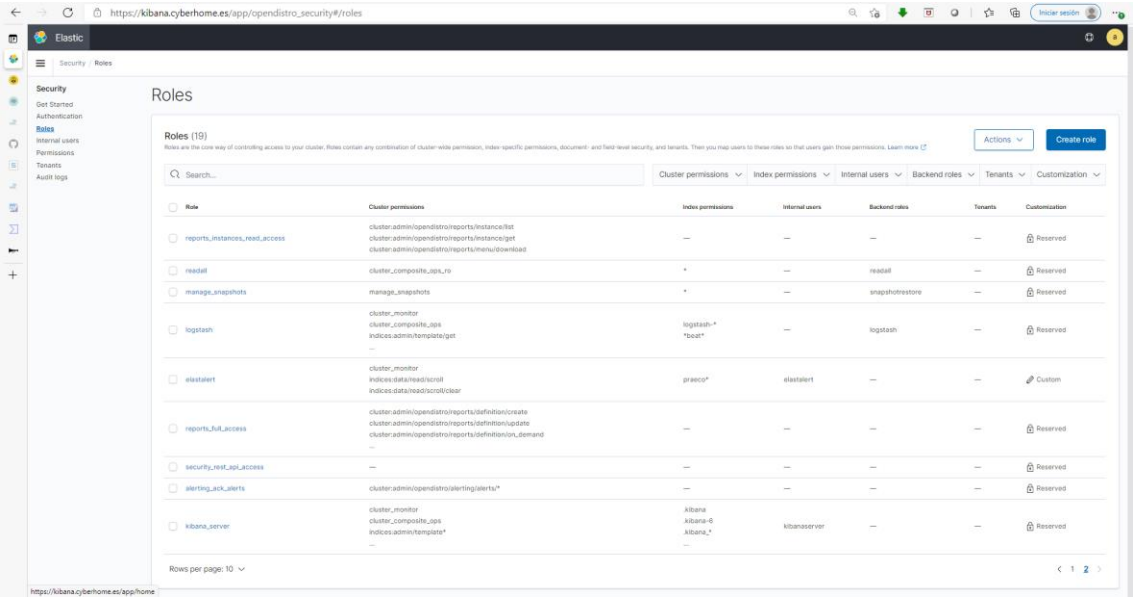


Ilustración 44 - Pantalla Roles Elasticsearch (permisos Elastalert)

Con todos estos productos cuya configuración se ha realizado durante este TFM es con lo que hemos creado nuestra plataforma SIRP.

5. Caso de uso

5.1 Descripción

Hay múltiples casos de uso en una plataforma SIRP, pero para este trabajo nos centraremos en uno que haga trabajar todos los componentes de nuestro entorno de simulación.

1- Evento o eventos de seguridad (repetidos) que Wazuh sea capaz de detectar en base a sus reglas, en este caso estamos haciendo uso de los agentes de Wazuh y el Wazuh Server, así como nuestra plataforma Elasticsearch que forma parte de la solución de Wazuh dado que allí se guardan todos los eventos
2- Podemos crear una alerta que directamente detecte el evento o grupo de eventos elegido y se integre con TheHive, de forma similar al ejemplo mostrado en la configuración del apartado 4.5
3- A partid de aquí entra en juego TheHive y se trata de coger la alerta y tratarla con nuestra herramienta de gestión de incidencias, crear un caso, añadir observables si procede, etc.
4- Una vez tenemos observables estos pueden ser analizados con nuestros Analyzers disponibles en Cortex y así ayudar a determinar si constituye un problema, se trata de un falso positivo, etc.
5- Entre los Analyzers disponibles tenemos nuestra plataforma MISP que sería una de las fuentes de información para enriquecer el caso si encuentra algún resultado.
6- En función de los hallazgos podemos decidir si dar respuesta mediante nuestros módulos de Cortex o bien la respuesta debe ser de otro tipo o simplemente terminar la documentación del caso y cerrarlo.

Con un caso de este tipo estaríamos utilizando todos los componentes de nuestra solución y podría ser un caso real que se encuentre un analista de seguridad de cualquier empresa u organización.

5.2 Test realizados

Previo al caso de uso se han realizado pruebas unitarias de cada uno de los productos y se ha validado su correcto funcionamiento para poder asegurar que el caso de uso tuviera éxito en su ejecución.

Se ha elaborado un caso simulado como se ha indicado para hacer uso de todos los componentes del entorno que se ha construido durante el transcurso de este TFM.

El caso en cuestión es un posible ataque de fuerza bruta realizado por un “insider” (por personal interno de la empresa).

Pasos que se han seguido:

1- Se trata de forzar un número ingente de Logins Fallidos con un script desde la máquina cliente Windows (WINCLT001) contra el usuario root de la máquina PRX001.

2- Esto como es lógico hace saltar eventos y además se detecta como posible TTP de Mitre, como ataque de fuerza bruta, que podemos observar en nuestra consola web Wazuh-Kibana.

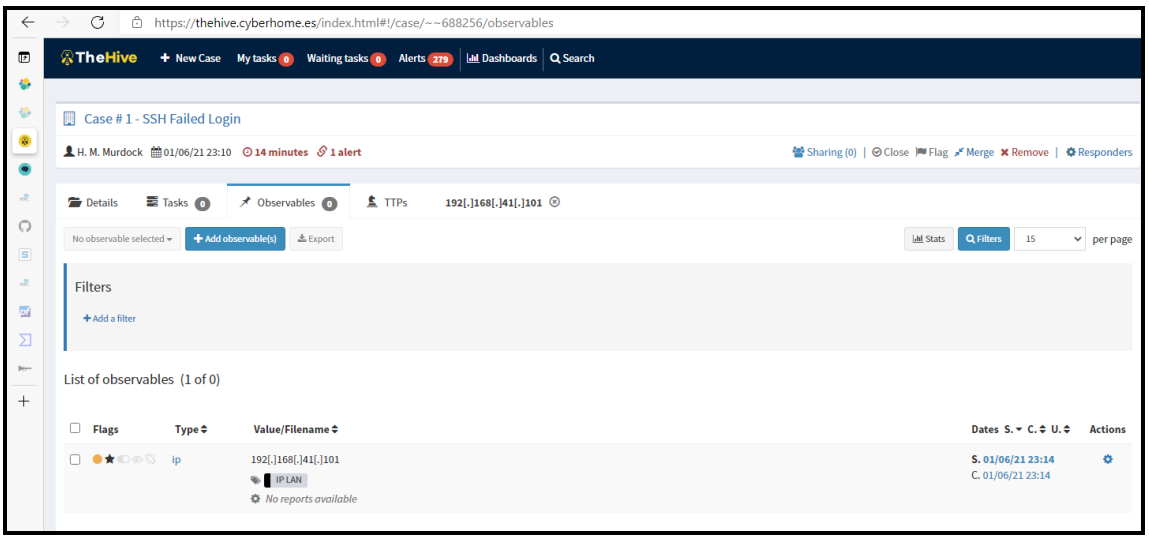
3- Se ha creado una alerta para que nuestro Praeco-Elastalert detecte los accesos fallidos en un corto periodo de tiempo, en la alerta se ha definido más de 10 eventos por minuto (algo que un humano no puede hacer).

4- Con estas alertas se recibe en TheHive, más de una alerta, ya que en la definición de la alerta se indica que si repite el comportamiento x minutos se genere otra alerta nueva.

5- En TheHive con nuestro usuario que hace las veces de administrador y analista (tenemos poco personal) pues revisamos la alerta y vemos que hay varias con el mismo origen, ya que hemos configurado que en los datos envíe la línea de “full_log”, dado que esta alerta en Wazuh no tiene un campo identificando la IP que podría ser tratada directamente como un observable.

6- Abrimos un caso con una de las alertas, como no teníamos ningún caso aparecerá como “Empty”, si pulsamos el botón de “Yes, import” nos creará un caso número asignando un número y el título de la alerta por nombre, pero también sería posible abrir un caso directamente con el título que queramos y entonces hacer un “Merge” con la alerta o alertas (serán varias) en el mismo caso.

7- Añadimos un nuevo observable e introducimos la dirección IP origen que viene en la alerta, añadimos un tag y una descripción.



8- Aunque como sabemos es una IP interna, así que no obtendremos resultados, pero probamos a pasarlo por nuestros “analyzers” de Cortex. Para ello pulsamos sobre la IP del observable y en la parte de abajo nos aparecen todos los “Analyzers” de Cortex disponibles.

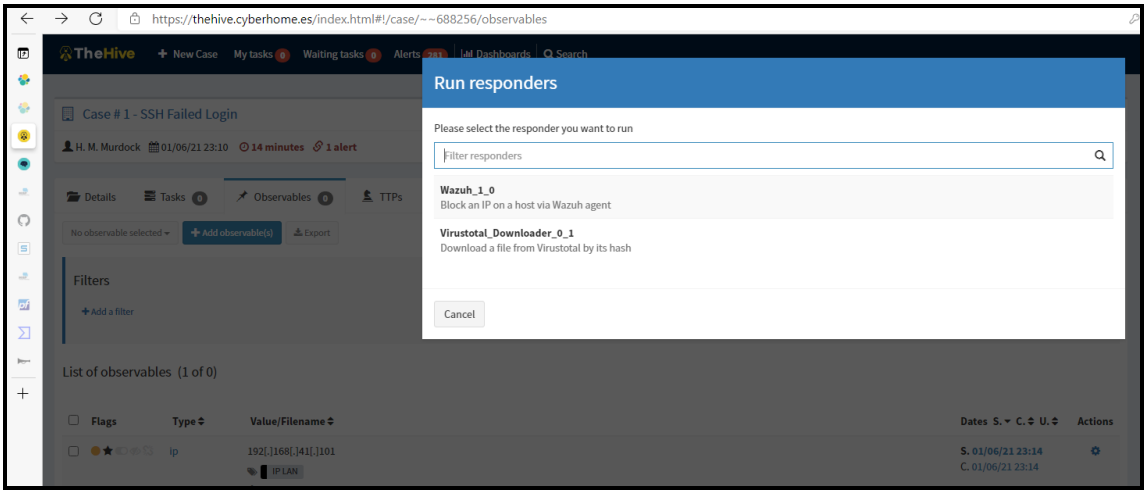
9- Podemos hacer un “Run All”, aunque como ya se ha comentado no encontraremos nada podemos ejecutar aquellos que sabemos que utilizan IPs, para nuestra prueba he utilizado MISP, Shodan (varios) y AbuseIPDB.

10-Como era de esperar tenemos resultados fallidos, dado que es poca información y que además es una IP privada de nuestra red empresarial simulada.

11- Aunque no hemos encontrado nada que avale que se trata de algo malicioso, el caso es que es una actividad ilícita, dado que como sabemos no es posible que se puedan generar tantas conexiones por minuto.

12- Así que se decide utilizar las capacidades de nuestro Cortex con el Wazuh como Responder (uno de los pocos configurados como se mostró en el apartado 4.3 cuando hablamos de configuración de Cortex).

13- Desde el caso en la pestaña “Observables” aparece a la derecha un icono con una rueda dentada, bajo el Título “Actions”, así que lo clickamos y seleccionamos “Wazuh”.



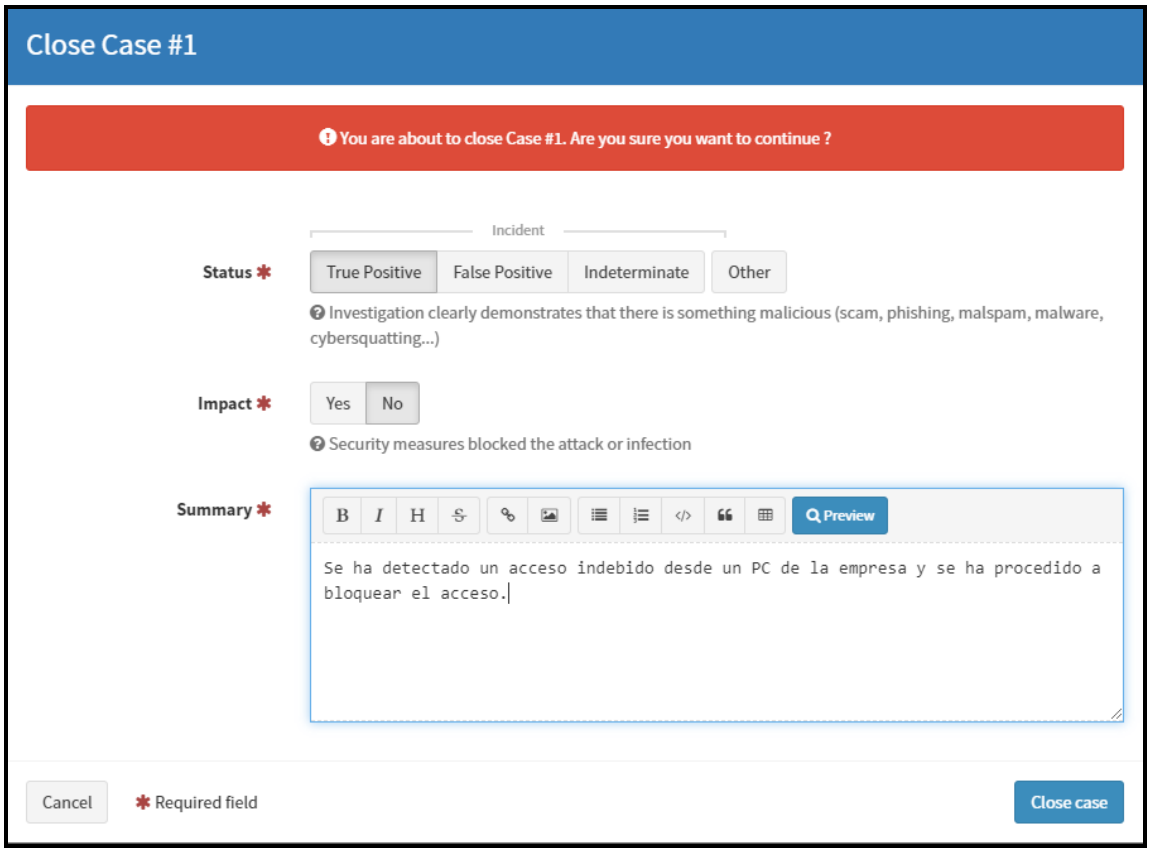
14- Nos aparecerá una pantalla de confirmación y pulsamos “Yes, run it”.

15- Así que ahora ya no debería poderse acceder desde ese equipo a esta máquina, una solución temporal mientras se hace otro tipo de investigación que a priori no podemos llevar a cabo con las herramientas (revisar el PC del usuario), podemos saber quien estaba logado antes de ir a verlo, porque tenemos el detalle en Wazuh, etc.

16- En mi caso he añadido además otro observable de tipo hostname, ya que sabemos a qué máquina corresponde la IP.

17- A partir de aquí ya habría que ver cuál es la política de la empresa u organización, si dejar el caso abierto hasta que se complete con la información adicional (que se estaba lanzando desde un script) o se podría finalizar el caso.

18- Para nuestro ejemplo de caso de uso procedemos al cierre con la información que se muestra.



6. Conclusiones

Tras el trabajo realizado se cumple el objetivo de disponer de una plataforma de Gestión de Incidentes plenamente operativa con capacidad de detección, análisis y respuesta.

Se ha construido con productos y herramientas de código abierto y utilizando herramientas que garanticen su alta disponibilidad y escalabilidad, ya sea vía kubernetes, balanceador software y/o el filesystem distribuido y replicado.

Se ha implementado un entorno de simulación, en el cual se representa por una parte una empresa u organización cualquiera con varios elementos a proteger y un entorno que simula un cloud donde se ejecuta los componentes de la plataforma SIRP.

Uno de los objetivos al iniciar este TFM es que esta plataforma con algunas lógicas modificaciones, dado que su implementación se ha realizado con carácter pedagógico pudiera dar servicio a una empresa u organización a priori de cualquier tamaño, dado que como se ha mencionado es escalable y altamente disponible, solo requiere de los recursos hardware necesarios para hacerla crecer y adaptarse al tamaño de la empresa u organización a la que tendría que dar soporte en el ámbito de gestión de incidentes de su SGSI.

Como se puede apreciar y teniendo en cuenta que se tocaban múltiples productos, solo se ha podido “rascar” la superficie de sus funcionalidades y no se ha podido probar y validar, tanto por temas temporales, como por recursos, dado que es un entorno de simulación.

Considerando el trabajo realizado, los siguientes pasos serían mejorar algunos aspectos de la instalación y configuración de los productos. Entre algunas de las mejoras que se deberían realizar en el entorno se encuentran las siguientes:

- Revisar y mejorar la instalación de los componentes en kubernetes:
    - Como por ejemplo homogeneizar la configuración de volúmenes, secretos y certificados
    - Utilizar servicio balanceador integrado en kubernetes, visibilidad de los microservicios al exterior a través de componentes Ingress
    - Integración de DNS externo con kubeDNS (DNS interno kubernetes)
    - Mejorar la seguridad con políticas de red dentro del cluster kubernetes
    - Limitar los recursos que pueden utilizar los componentes, para Wazuh está hecho porque la configuración recomendada ya incluida dichos controles, pero el resto de componentes no

- Añadir monitorización del propio kubernetes para determinar que el pod está funcionando correctamente, en caso contrario lo reinicie

- Etc.

- Mejorar la seguridad de algunos componentes, por ej. en la gestión de certificados, aunque los expuestos al usuario se han gestionado con una PKI externa, hay algunos certificados autogenerados y/o desactivada la verificación de los mismos.

- Siguiendo con temas de seguridad de la solución, modificar todas las passwords por defecto de los componentes (aquellas que no han sido configuradas explícitamente)

- Incluso hay un componente que no dispone de sistema de autenticación como es Praeco, al menos se debería implementar autenticación básica a través del NGINX.

- Sobre la autenticación en general una mejora sería implementar la gestión de usuarios a través de una fuente externa, léase un AD o un servicio LDAP, incluso se podría pensar en alguna gestión integrada mediante los componentes SSO que incluyen algunos productos (por ej. TheHive). El cual también dispone de una opción muy necesaria hoy en día MFA.

- Sobre mejoras en la funcionalidad, sería conveniente utilizar Logstash que finalmente no se ha implementado en esta solución y agentes Beat en los sistemas a proteger de cara a recoger logs de otros productos aparte del SO.

- También sería necesario mejorar la administración y gestión del Elasticsearch para convertirlo en un verdadero SIEM, según lo comentado en el punto anterior, optimizar la implementación de índices, shards, réplicas, etc.

- Sobre Elasticsearch también sería una mejora utilizarlo para almacenar datos de TheHive, así como de Cortex, en nuestra instalación tiene una instancia de elastic propia que se incluía en el repositorio de instalación y que no se ha querido modificar por si había algún problema de compatibilidad de versiones.

- Mejorar las capacidades de análisis y respuesta del componente Cortex, recurriendo a las fuentes de inteligencia externa y que normalmente son servicios de pago

- En cuanto al componente MISP sería una buena opción compartir información con otras organizaciones y/o entidades del entorno, ámbito empresarial u organizativo, así como organismos oficiales como por ej.

el INCIBE, CSICAT, etc. de tal manera que podamos enriquecer nuestras capacidades y también colaborar con otros equipos SOC, CERT o CSIRT tanto públicos como privados.

- En los capítulos iniciales hablábamos de Suricata como elemento de detección y respuesta, de echo nuestra simulación de red empresarial dispone de uno en modo IPS, pero no lo hemos llegado a integrar con nuestra solución SIRP con lo que sería otra de las mejoras para el entorno.

7. Glosario

CSIRT se trata del equipo de respuesta ante incidentes de seguridad, normalmente formado por expertos en diferentes áreas que permiten realizar acciones tanto preventivas como reactivas ante incidentes de seguridad. Esta formado principalmente por expertos en seguridad, pero cada vez es más habitual que se integre personal de TI y de otras áreas como legal o comunicación.

EDR son las siglas de Endpoint Detection & Response y se corresponde con un software que se instala en equipos tanto cliente, como servidor para poder detectar y responder ante amenazas (por ej. malware). Algunos EDR tienen la capacidad de aprender el funcionamiento normal de un equipo y detectar actividad anómala con respecto a ese funcionamiento base.

HIDS son las siglas de Host Intrusion Detection System, es un software que permite detectar posibles ataques en base a patrones (firmas).

IDS siglas de Intrusion Detection System, generalmente se refiere a un sistema de red que realiza esta función, pero se puede aplicar el termino tanto a HIDS, como NIDS. Este tipo de productos suelen detectar posibles ataques y emitir alertas.

IOC es el acrónimo de indicador de compromiso, se podría decir que son evidencias que forman parte de la inteligencia de amenazas y permite prevenir, detectar y actuar (tendríamos por ej. direcciones IP, hash de ficheros, patrones, etc.)

IPS a diferencia de los IDS, el Intrusion Prevention System permite no solo detectar posibles ataques y alertar, sino tomar acciones, básicamente cortar una conexión o en algunos casos alterar la respuesta por ejemplo para ocultar un sistema vulnerable, también existen como elementos de red (inline) o como productos de Host (a veces forman parte de una solución EDR).

K8S es la denominación abreviada de kubernetes, en este documento se utiliza indistintamente kubernetes o k8s, que es lo mismo.

MFA abreviación de Multi Factor Authentication, hoy en muchos entornos ya no se considera seguro usar un solo factor de autenticación para demostrar nuestra identidad, por ej. una contraseña, por lo que es necesario recurrir a más factores de autenticación, como puede ser usar OTP ya sea por SMS o una aplicación como Google Authenticator o Microsoft Authenticator, usar certificados o biometría (huella dactilar, Face-ID, etc.).

NIDS de Network Intrusion Detection System, se trata de un producto software o hardware (appliance) que permite detectar patrones de ataque y alertar de ello, a diferencia de los IPS no interfiere en el tráfico de red (normalmente se

configuran para recibir una copia del tráfico para poder analizarlo en tiempo real).

PKI o Infraestructura de Clave Pública, se trata de una infraestructura de CA (Certificate Authority) que permite la firma de certificados.

OTP siglas de One Time Password, es un tipo de factor de autenticación que se basa en disponer de una contraseña de un solo uso y que suele ser válida durante un tiempo restringido.

OWASP es un proyecto que se encarga de velar por la seguridad básicamente de aplicaciones web, elabora la lista Top Ten de vulnerabilidades conocidas y más explotadas por tipología en las aplicaciones web, detrás esta la Fundación OWASP en la que se puede participar y contribuir.

SGSI son las siglas de Sistema de Gestión de Seguridad de la Información, sus siglas en inglés son ISMS y a falta de una definición más formal y completa se trata del conjunto de procedimientos y políticas de seguridad destinados a proteger los sistemas de información de una empresa u organismo.

SIEM de las siglas de Security Information & Event Management o lo que es lo mismo es un sistema que recibe la información de alertas y eventos que generan los diferentes productos de detección, así como logs de actividad, auditoría, etc. Es un sistema que permite reunir toda esta información de forma que se pueda correlacionar, buscar patrones, etc. y así generar alertas. Es una herramienta considerada básica hoy en día en un SOC o CSIRT.

SOAR es un tipo de producto que va un poco más allá del trabajo que realiza un SIEM y permite gestionar incidentes y su respuesta, las siglas provienen de Security Orchestation Automation and Response, es decir, se trata de un producto que permite ayudar a los SOC y CSIRT para poder automatizar acciones de respuesta ante posibles incidentes.

SOC, los Security Operation Centers se dedican a prevenir, monitorizar y controlar la seguridad de redes y sistemas. Es un tipo de operación normalmente especializada en seguridad como su nombre indica, aunque en ocasiones forman parte o trabajan conjuntamente con un NOC (Network Operation Center).

TOR es una red compuesta por varios nodos, que permite garantizar el anonimato mediante nodos que ocultarán nuestra dirección IP al destino de nuestra conexión, así como acceder a direcciones “.onion” de la Deep Web (esa parte de Internet que no está indexada por los buscadores) y en la cual se ampara la Dark Web (donde se suelen realizar operativas de dudosa reputación y/o legalidad).

VPN de las siglas Virtual Private Network, permite proporcionar un canal seguro entre dos puntos en la red, normalmente se utiliza para proporcionar confidencialidad sobre un canal no seguro como puede ser Internet.

8. Bibliografía

Referencias en el TFM:

[1] Web de “Wazuh · The Open Source Security Platform.”: https://wazuh.com/
(acceso Feb 27, 2021)

[2] Web de Microsoft: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
(acceso Mar 01, 2021)

[3] Web de Elastic: https://www.elastic.co/es/logstash
(acceso Feb 28, 2021)

[4] Web de Elastic: https://www.elastic.co/es/elasticsearch/
(acceso Feb 28, 2021)

[5] Web de Elastic: https://www.elastic.co/es/kibana
(acceso Feb 28, 2021)

[6] Web de Elastic: https://www.elastic.co/es/beats/filebeat
(acceso Feb 28, 2021)

[7] Web de Elastic: https://www.elastic.co/es/beats/packetbeat
(acceso Feb 28, 2021)

[8] Articulo Web “Open Source SIRP with Elasticsearch and TheHive – Overview” : https://arnaudloos.com/2019/open-source-sirp-overview/
Autor: Arnaud Loos
(acceso Mar 01, 2021)

[9] Web del “Proyecto IntelMQ”: https://intelmq.readthedocs.io/en/maintenance/
(acceso Mar 01, 2021)

[10] Artículo web “Deploying of infrastructure and technologies for a SOC as a Service (SOCasS)”: https://medium.com/@ibrahim.ayadhi/deploying-of-infrastructure-and-technologies-for-a-soc-as-a-service-socass-8e1bbb885149
Autor: Ibrahim Ayadhi
(acceso Mar 02, 2021)

[11] Web del Proyecto MISP “Features of MISP“: https://www.misp-project.org/features.html
(acceso Mar 15, 2021)

Principales referencias de consulta:

- Web del Proyecto TheHive “TheHive Project Documentation”: https://docs.thehive-project.org/thehive
- Web del Proyecto Cortex “CortexDoc”: https://github.com/TheHive-Project/CortexDocs
- Web del Proyecto Praeco: https://github.com/johnsusek/praeco
- Web de “Wazuh · The Open Source Security Platform.”: https://wazuh.com/
- Web del Proyecto MISP: https://www.misp-project.org/
- Web de kubernetes “Kubernetes Documentation”: https://kubernetes.io/docs/home/

Anexo 1 Entorno Kubernetes (k8s)

No entraremos en muchos detalles sobre el cómo instalar un cluster kubernetes en alta disponibilidad, ya se ha explicado algunos detalles en apartados anteriores, pero no es el objetivo de este trabajo, solo recordaremos algunos detalles de esta instalación en concreto que pueden ser útiles para entender cómo se han instalado los componentes de la plataforma SIRP.

En el entorno kubernetes que se ha preparado tenemos tres nodos, digamos de administración (aunque podrían ser operatives para desplegar aplicaciones, pero se han configurado para no hacerlo) ya que es un clúster en alta disponibilidad por lo tanto y sin entrar en excesivos detalles sobre el funcionamiento de kubernetes tenemos los componentes api-server, scheduler, etcd replicados en cada uno de los nodos que he denominado master, con la ayuda del software keepalived y haproxy integrado con kubernetes podemos tener un clúster kubernetes en el cual si deja de funcionar uno de los nodos cualquiera de los otros dos puedes mantenir el servicio de kubernetes operativo.

En cada nodo de administración se ha configurado para poder administrar kubernetes como **root** (usuario con el que hay que hacer la instalación de kubernetes) o bien con el usuario **kadmin**.el cual se ha configurado para no tener que utilitzar el usuario root sin ser necesario.

```
[kadmin@csirt01 ~]$ kubectl get nodes
NAME                    STATUS                      ROLES                  AGE   VERSION
csirt01.cyberhome.es    Ready,SchedulingDisabled    control-plane,master   31d   v1.21.0
csirt02.cyberhome.es    Ready,SchedulingDisabled    control-plane,master   31d   v1.21.0
csirt03.cyberhome.es    Ready,SchedulingDisabled    control-plane,master   31d   v1.21.0
csirt04.cyberhome.es    Ready                       worker1                31d   v1.21.0
csirt05.cyberhome.es    Ready                       worker2                31d   v1.21.0
csirt06.cyberhome.es    Ready                       worker3                31d   v1.21.0
csirt07.cyberhome.es    Ready                       worker4                31d   v1.21.0
```

Ilustración 45 - Nodos en Kubernetes

Además del clúster kubernetes se ha instalado también el GlusterFS para poder disponer de un espacio distribuido y compartido entre todos los nodos. Se han creado dos volumenes con las diferentes carpetas donde se ubicarán los datos persistentes de nuestras aplicaciones deployadas, ya que un entorno kubernetes utiliza contenedores y estos se basan en imágenes que son inmutables y por tanto cuando se crean de nuevo los datos no persistidos se perderían.

En esta instalación los nodos workers tienen configurados discos que como ya se ha indicado comparten en dos volúmenes distribuidos entre los 4 nodos workers (donde deployaremos aplicacions) y con número de réplica 2, es decir, cada dato se escribe dos veces en el filesystem compartido.

Los volúmenes compartidos se han configurado para montarse en:

/shared/vol01
/shared/vol02

```
[root@csirt01 ~]# gluster volume info vol01

Volume Name: vol01
Type: Distributed-Replicate
Volume ID: 33e39558-bf81-4613-b36c-cce0fbb25870
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: csirt04:/gfs/vol01/br
Brick2: csirt05:/gfs/vol01/br
Brick3: csirt06:/gfs/vol01/br
Brick4: csirt07:/gfs/vol01/br
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

```
[root@csirt01 ~]# gluster volume info vol02

Volume Name: vol02
Type: Distributed-Replicate
Volume ID: d8ae9ec6-ad0d-4514-89fc-5550aeb82f77
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: csirt04:/gfs/vol02/br
Brick2: csirt05:/gfs/vol02/br
Brick3: csirt06:/gfs/vol02/br
Brick4: csirt07:/gfs/vol02/br
Options Reconfigured:
cluster.granular-entry-heal: on
storage.fips-mode-rchecksum: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

Ilustración 46 - Info definición Volumenes GlusterFS

De esta forma cualquier nodo del cluster kubernetes accede a su disco como si fuera disco local, aunque en realidad no lo es.

Para dar acceso a los Servicios publicados por el clúster de kubernetes he utilizado unos balanceadores externos, hay otras formes de hacer la instalación en un entorno de cloud público normalmente se ofrecen estos Servicios, en nuestra instalación podríamos haber utilizado keepalived y haproxy también para esta funcionalidad de forma similar a como dan alta disponibilidad a los Servicios básicos de kubernetes. En mi caso para no añadir más carga a los nodos workers he preferido que la función fuera externa.

Anexo 2 Wazuh en k8s

Para realizar esta instalación del software Wazuh partimos de la base de que disponemos de un entorno k8s plenamente operativo y tenemos un usuario con permisos de administración en nuestro entorno de kubernetes.

Esta instalación se ha realizado siguiendo las instrucciones dadas en la documentación de Wazuh para un entorno kubernetes, aunque la instalación está muy pensada para ser utilizada sobre un cluster kubernetes EKS de Amazon y se han tenido que hacer adaptaciones, aparte de algunos cambios que he tenido que realizar al observar que la instalación no acababa de funcionar por ejemplo en la parte de elasticsearch.

Primero clonamos el repositorio GIT donde se encuentra “la receta” para deployar nuestras aplicaciones sobre el cluster k8s.

```
git clone https://github.com/wazuh/wazuh-kubernetes.git -b v4.1.1 --depth=1
```

A esto se debe que no tenga la última versión de Wazuh Server que actualmente es la 4.1.5, aunque existen imágenes Docker para dicha versión no sé si están testeadas para funcionar en kubernetes, así que he tomado una decisión conservadora.

El método de instalación es el recomendado por kubernetes, instalación declarativa en el cual se indica a kubernetes como queremos que estén nuestros componentes funcionando y k8s se encarga de arrancar los componentes para intentar alcanzar ese estado declarado en los yaml manifests.

La instalación se realiza en conjunto de todos los componentes usando kustomization (es el método que usaremos para instalar todos nuestros componentes), aunque los he troceado para cada grupo de componentes, dado que durante el proyecto esto ha sido una instalación progresiva, instalar, verificar que los componentes operaban como se esperaba y seguir adelante.

Para los certificados utilicé los scripts que ofrecía el producto, eliminando la creación de la CA y la firma de los mismos, este paso en mi caso fue un paso manual, ya que le pasaba los CSR creados a una PKI externa para ser firmados.

En el caso de Elasticsearch se ha utilizado la OpenDistro que es la versión 1.12 que equivale a un Elasticsearch 7.10, en la instalación también viene Kibana con el plugin instalado y las plantillas de Wazuh para Elasticsearch, así como Filebeat que corre en los mismos contenedores que nuestro Wazuh.

La modificación que comentaba que he tenido que realizar sobre la instalación de elasticsearch, en la instalación propuesta por Wazuh se instalar 3 réplicas de elasticsearch con el mismo fichero de configuración y esto aunque debería funcionar, en la práctica no funcionaba, las réplicas tenían que funcionar como

un cluster de elasticsearch, pero aunque había conectividad entre ellas no conseguían que uno de los nodos asumiera la función de master, por lo tanto en lugar de arrancar 3 réplicas, partí en dos partes la instalación forzando a tener un nodo master y que los otros dos nodos no lo fueran, es una parámetro en la configuración que según indica al arrancar esta “deprecated” y que en futuras versiones ya no se usará, pero después de múltiples intentos de encontrar el fallo es la única solución que se me ocurrió para hacerlo funcionar.

Los servicios que publican el contenido hacia el exterior del cluster kubernetes también han sido modificados, estaban pensados para trabajar con un balanceador kubernetes integrado de EKS, dado que no tenemos dicho componente se podía utilizar un ingress controler como traefik o haproxy, pero finalmente he optado por un balanceador externo y usar servicios NodePort, así tenemos servicios internos (solo visibles desde dentro del cluster Kubernetes, ClusterIP y los NodePort para dar visibilidad desde fuera del cluster).

Los servicios NodePort están restringidos a un rango de puertos que no suele ser habitual para utilizar al menos en tráfico web, así que con el balanceador externo usé IPs y puertos estándar para los componentes web y para los servicios de Wazuh, como API Server y Reporting de los agentes usé sus puertos estándar.

Así pues, aplicamos nuestra “receta” para instalar los componentes:

```
kubectl apply -k wazuh
```

```
[kadmin@csirt01 wazuh_managers]$ kubectl get all -n wazuh -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP          NODE                  NOMINATED NODE   READINESS GATES
pod/wazuh-elasticsearch-0           1/1     Running   1          23d   10.36.0.2   csirt04.cyberhome.es  <none>           <none>
pod/wazuh-elasticsearch-1           1/1     Running   1          23d   10.42.0.4   csirt05.cyberhome.es  <none>           <none>
pod/wazuh-elasticsearch-master-0    1/1     Running   1          14d   10.40.0.1   csirt07.cyberhome.es  <none>           <none>
pod/wazuh-kibana-5b7f4fccdd-j9fft   1/1     Running   1          23d   10.39.0.4   csirt06.cyberhome.es  <none>           <none>
pod/wazuh-manager-master-0          1/1     Running   1          18d   10.42.0.2   csirt05.cyberhome.es  <none>           <none>
pod/wazuh-manager-worker-0          1/1     Running   1          14d   10.40.0.2   csirt07.cyberhome.es  <none>           <none>
pod/wazuh-manager-worker-1          1/1     Running   1          23d   10.36.0.3   csirt04.cyberhome.es  <none>           <none>

NAME                         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                         AGE   SELECTOR
service/elasticsearch        NodePort    10.101.62.53    <none>        9200:30029/TCP                  23d   app=wazuh-elasticsearch
service/kibana               NodePort    10.110.6.140    <none>        443:30443/TCP                   23d   app=wazuh-kibana
service/wazuh                NodePort    10.107.13.197   <none>        1515:31515/TCP,55000:30055/TCP  23d   app=wazuh-manager,node-type=master
service/wazuh-cluster        ClusterIP   None            <none>        1516/TCP                        23d   app=wazuh-manager
service/wazuh-elasticsearch  ClusterIP   None            <none>        9300/TCP                        23d   app=wazuh-elasticsearch
service/wazuh-workers        NodePort    10.108.184.27   <none>        1514:31514/TCP                  23d   app=wazuh-manager,node-type=worker

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS     IMAGES                        SELECTOR
deployment.apps/wazuh-kibana   1/1     1            1           23d   wazuh-kibana   wazuh/wazuh-kibana-odfe:4.1.1   app=wazuh-kibana

NAME                                   DESIRED   CURRENT   READY   AGE   CONTAINERS     IMAGES                        SELECTOR
replicaset.apps/wazuh-kibana-5b7f4fccdd   1         1         1       23d   wazuh-kibana   wazuh/wazuh-kibana-odfe:4.1.1   app=wazuh-kibana,pod-template-hash=5b7f4fccdd

NAME                                      READY   AGE   CONTAINERS            IMAGES
statefulset.apps/wazuh-elasticsearch      2/2     23d   wazuh-elasticsearch   amazon/opendistro-for-elasticsearch:1.12.0
statefulset.apps/wazuh-elasticsearch-master   1/1     23d   wazuh-elasticsearch   amazon/opendistro-for-elasticsearch:1.12.0
statefulset.apps/wazuh-manager-master     1/1     19d   wazuh-manager         wazuh/wazuh-odfe:4.1.1
statefulset.apps/wazuh-manager-worker     2/2     23d   wazuh-manager         wazuh/wazuh-odfe:4.1.1
```

Ilustración 47 - Elementos kubernetes Wazuh

Esta imagen es la comprobación de que todos los componentes están operativos, hay que esperar un tiempo prudencial para que se descargue las imágenes y arranque todos los componentes.

Se puede comprobar su funcionamiento tal como se indica en el apartado 4.1 accediendo a nuestro Wazuh-Kibana. Una comprobación que también podemos realizar y que me ha sido bastante útil, ya que a veces por algún motivo los nodos de wazuh no se conectaban entre ellos es usar un comando del propio wazuh.

Primero entramos en el contenedor de nuestro wazuh master (también se puede ejecutar el comando sin entrar con una Shell), pero de esta forma si algo no funciona también podemos revisar los logs de wazuh.

```
[kadmin@csirt01 manifests]$ kubectl exec -it wazuh-manager-master-0 -n wazuh bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
[root@wazuh-manager-master-0 /]# /var/ossec/bin/cluster_control -l
NAME                    TYPE    VERSION  ADDRESS
wazuh-manager-master    master  4.1.1    wazuh-manager-master-0.wazuh-cluster.wazuh
wazuh-manager-worker-0  worker  4.1.1    10.40.0.2
wazuh-manager-worker-1  worker  4.1.1    10.36.0.3
```

Ilustración 48 - Listado Nodos Wazuh

Los logs se guardan en /var/ossec/logs, aunque en este caso podemos ver que los nodos de nuestro cluster wazuh se ven disponibles.

Para ver el detalle de los ficheros yaml y de configuración he creado un repositorio en GIT donde se pueden descargar e instalar en un cluster kubernetes baremetal según las condiciones ya comentadas.

https://github.com/risingonline19/Plataforma-SIRP/wazuh

Usando este repositorio no se necesita hacer el clone del repositorio original de Wazuh, eso no significa como se puede ver en los ficheros yaml que las imágenes no sean las originales creadas por el Proyecto Wazuh y que se descargan de Docker Hub.

Anexo 3 TheHive y Cortex en k8s

Para realizar esta instalación partimos de la base de que disponemos de un entorno k8s plenamente operativo y tenemos un usuario con permisos de administración en nuestro entorno de kubernetes.

Una vez instalado y operativo el entorno de Wazuh con todos sus componentes para proseguir hemos usado de base la forma en como se ha instalado Wazuh para instalar los diferentes componentes de nuestra plataforma SIRP para ello hemos usado la fórmula de usar kustomization para hacer paquetes de instalación.

Sin entrar en detalles de para que se usan, comentar que la instalación de componentes de nuestra plataforma SIRP se ha realizado usando dos namespaces diferentes, el primero y que se ha seguido la sugerencia de instalación wazuh y en este caso uno que hemos creado al instalar el primer componte namespace sirp.

El namespace es un parámetro que utilizamos cuando hacemos consultas del estado de nuestros componentes en kubernetes para que los encuentre y además para filtrar lo que queremos mostrar dado que siempre tenemos la opción de indicar --all-namespaces

Al igual que hicimos con Wazuh lo primero es descargar el contenido de los repositorios de GIT, en el caso de TheHive y Cortex en Docker hay múltiples opciones con diferentes versiones y con diferente soporte para guardar los datos, etc.

En nuestro caso seleccionamos la versión que usa TheHive con Cassandra y Cortex con Elasticsearch y para dar visibilidad desde el exterior usamos un Reverse Proxy NGINX.

```
git clone https://github.com/TheHive-Project/Docker-Templates/tree/main/docker/thehive4-cortex31-nginx-https
```

Comentar, que a diferencia de la instalación de Wazuh que, aunque se ha tenido que hacer adaptaciones venía preparada para instalar sobre k8s, esta implementación está pensada para hacer directamente sobre Docker usando la herramienta docker-compose.

Así que lo primero que se ha tenido que hacer, para no empezar desde cero ha sido usar otra herramienta kompose que permite convertir un fichero docker-compose.yml que es el que utiliza la herramienta docker-compose para saber cómo deployar y configurar las imágenes Docker de los productos.

```
[kadmin@csirt01 thehive4-costex31-nginx-https]$ kompose convert
WARN Restart policy 'unless-stopped' in service thehive is not supported, convert it to 'always'
WARN Restart policy 'unless-stopped' in service cassandra is not supported, convert it to 'always'
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/cassandra/data" isn't supported - ignoring path on the host
INFO Network backend is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO Network backend is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO Network proxy is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/elasticsearch/data" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/elasticsearch/logs" isn't supported - ignoring path on the host
INFO Network backend is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/nginx" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/ssl" isn't supported - ignoring path on the host
INFO Network proxy is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/thehive/application.conf" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/thehive/data" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/thehive4-costex31-nginx-https/vol/thehive/index" isn't supported - ignoring path on the host
INFO Network proxy is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO Network backend is detected at Source, shall be converted to equivalent NetworkPolicy at Destination
INFO Kubernetes file "nginx-service.yaml" created
INFO Kubernetes file "thehive-service.yaml" created
INFO Kubernetes file "cassandra-deployment.yaml" created
INFO Kubernetes file "cassandra-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "backend-networkpolicy.yaml" created
INFO Kubernetes file "cortex-deployment.yaml" created
INFO Kubernetes file "proxy-networkpolicy.yaml" created
INFO Kubernetes file "elasticsearch-deployment.yaml" created
INFO Kubernetes file "elasticsearch-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "elasticsearch-claim1-persistentvolumeclaim.yaml" created
INFO Kubernetes file "nginx-pod.yaml" created
INFO Kubernetes file "nginx-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "nginx-claim1-persistentvolumeclaim.yaml" created
INFO Kubernetes file "thehive-deployment.yaml" created
INFO Kubernetes file "thehive-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "thehive-claim1-persistentvolumeclaim.yaml" created
INFO Kubernetes file "thehive-claim2-persistentvolumeclaim.yaml" created
```

Ilustración 49 - Ejecución Kompose TheHive-Cortex

Cuando ejecutamos kompose en el directorio donde está ubicado el docker-compose.yml nos genera toda una serie de ficheros para hacer la instalación en kubernetes.

Se ha seguido un esquema similar al kustomization de wazuh, así que después de personalizar y corregir algunas opciones de los ficheros generados por kompose.

Hay que configurar los fqdn para la configuración de proxy revers de NGINX, crear los CSR y firmarlos con mi PKI, etc. Ya podemos deployar nuestros componentes de TheHive y Cortex con sus backends de BBDD correspondientes, como ya se ha comentado para Cortex usamos un elasticsearch standalone propio, habría que investigar si es posible utilizar el mismo que ya tenemos operativo para Wazuh, pero para evitar algún tema de compatibilidad de versiones no he entrado en eso.

Aplicamos nuestra configuración en nuestro k8s para estos componentes:

```
kubectl apply -k thehive
```

```
[kadmin@csirt01 manifests]$ kubectl get all -n sirp -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP          NODE                  NOMINATED NODE   READINESS GATES
pod/cassandra-0               1/1     Running   0          11d   10.39.0.1   csirt06.cyberhome.es  <none>           <none>
pod/cortex-6bb4d47cff-plbwr   1/1     Running   0          11d   10.40.0.4   csirt07.cyberhome.es  <none>           <none>
pod/elasticsearch-0           1/1     Running   1          19d   10.39.0.3   csirt06.cyberhome.es  <none>           <none>
pod/misp-db-0                 1/1     Running   1          22d   10.39.0.2   csirt06.cyberhome.es  <none>           <none>
pod/misp-web-7dd986bf5-hc6kx  1/1     Running   1          14d   10.36.0.4   csirt04.cyberhome.es  <none>           <none>
pod/nginx                     1/1     Running   0          11d   10.40.0.5   csirt07.cyberhome.es  <none>           <none>
pod/thehive-6dcbbf4bc9-22xnq  1/1     Running   0          11d   10.40.0.6   csirt07.cyberhome.es  <none>           <none>

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                  AGE   SELECTOR
service/cassandra     ClusterIP   None            <none>        7000/TCP,7001/TCP,7199/TCP,9042/TCP,9160/TCP  19d   app=cassandra
service/cortex        ClusterIP   10.109.182.177  <none>        9001/TCP                                 20d   io.kompose.service=cortex
service/elasticsearch ClusterIP   None            <none>        9200/TCP,9300/TCP                        19d   app=elasticsearch
service/misp-db       ClusterIP   None            <none>        3306/TCP                                 22d   app=misp
service/misp-web      NodePort    10.105.173.19   <none>        443:31443/TCP                            20d   app=misp-web
service/nginx         NodePort    10.99.79.244    <none>        443:32443/TCP                            20d   io.kompose.service=nginx
service/thehive       ClusterIP   10.97.179.72    <none>        9000/TCP                                 20d   io.kompose.service=thehive

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                        SELECTOR
deployment.apps/cortex   1/1     1            1           19d   cortex       thehiveproject/cortex:latest  io.kompose.service=cortex
deployment.apps/misp-web 1/1     1            1           15d   misp-web     risingonline/misp:latest      app=misp-web
deployment.apps/thehive  1/1     1            1           19d   thehive      thehiveproject/thehive4:latest io.kompose.service=thehive

NAME                                DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                        SELECTOR
replicaset.apps/cortex-6bb4d47cff   1         1         1       19d   cortex       thehiveproject/cortex:latest  io.kompose.service=cortex,pod-template-hash=6bb4d47cff
replicaset.apps/misp-web-7dd986bf5  1         1         1       15d   misp-web     risingonline/misp:latest      app=misp-web,pod-template-hash=7dd986bf5
replicaset.apps/thehive-6dcbbf4bc9  1         1         1       19d   thehive      thehiveproject/thehive4:latest io.kompose.service=thehive,pod-template-hash=6dcbbf4bc9

NAME                        READY   AGE   CONTAINERS     IMAGES
statefulset.apps/cassandra  1/1     19d   cassandra      cassandra:3.11
statefulset.apps/elasticsearch 1/1  19d   elasticsearch  elasticsearch:7.11.1
statefulset.apps/misp-db    1/1     22d   misp-db        mysql/mysql-server:5.7
```

Ilustración 50 - Elementos kubernetes SIRP

Esperamos a que los componentes se hayan deployado, descargado imágenes, creado contenedores, inicializado BBDD, etc. y deberíamos tener una foto muy similar a la captura anterior.

Al igual que en wazuh he subido mis yaml manifests a un repositorio GIT:

https://github.com/risingonline19/Plataforma-SIRP/thehive

Las imágenes son las de los repositorios de Docker Hub y que se informaban en el repositorio GIT de donde hemos obtenido la instalación original, pero no es necesario hacer los pasos previos si usamos esta “receta” sobre un cluster k8s baremetal.

Anexo 4 MISP en k8s

Para realizar esta instalación partimos de la base de que disponemos de un entorno k8s plenamente operativo y tenemos un usuario con permisos de administración en nuestro entorno de kubernetes como en los casos anteriores.

En este caso la instalación de este producto sigue las pautas utilizadas para Wazuh y TheHive, descargar del repositorio GIT una instalación para Docker, pasar el kompose y adaptar los ficheros resultantes.

Así que descargamos el contenido del repositorio:

```
git clone https://github.com/misp/misp-docker
```

Ejecutamos el kompose para obtener los ficheros preparados para instalación en entorno kubernetes.

```
[kadmin@csirt01 misp-docker]$ kompose convert
WARN Restart policy 'unless-stopped' in service web is not supported, convert it to 'always'
WARN Restart policy 'unless-stopped' in service db is not supported, convert it to 'always'
WARN Volume mount on the host "/db" isn't supported - ignoring path on the host
WARN Volume mount on the host "/dev/urandom" isn't supported - ignoring path on the host
WARN Volume mount on the host "/web" isn't supported - ignoring path on the host
INFO Kubernetes file "web-service.yaml" created
INFO Kubernetes file "db-deployment.yaml" created
INFO Kubernetes file "db-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "web-deployment.yaml" created
INFO Kubernetes file "web-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "web-claim1-persistentvolumeclaim.yaml" created
```

Ilustración 51 - Ejecución Kompose MISP

Pero en este caso y debido a utilizar el repositorio oficial de MISP Project no tenemos una imagen docker para descargar de Docker HUB como en los casos anteriores.

Existen opciones de instalar imágenes ya creadas por terceros como por ejemplo Harvard-itsecurity.

```
https://github.com/harvard-itsecurity/docker-misp
```

Pero en este caso no he optado por utilizar las imágenes de terceros, concretamente por dos motivos, la primera es una imagen que no es parte del proyecto oficial (aunque parece que es bastante utilizada por lo que su reputación es positiva) y la otra es porque a mi modo de entender transgrede un poco la filosofía de kubernetes (tenemos en una única imagen todo el MISP, la BBDD y la aplicación web, es decir, el front-end y el backend todo junto).

Así que me creo mi propia imagen de Docker con los ficheros del repositorio oficial de GIT y para ello primero tengo que generarla en local y luego subirla a Docker Hub, aunque la he generado en local, cada vez que se deploye sobre un nodo de k8s necesitará disponer de la imagen y no la voy a subir manualmente a cada repositorio local de Docker en cada nodo.

Así que, dentro del repositorio clonado en local de GIT, entramos en el directorio web (donde se encuentra el dockerfile) y ejecutamos el siguiente comando:

```
docker build -t risingonline/misp:latest .
```

Una vez se crea la imagen, tarda un poco porque tiene que construir con varias capas, si revisamos el dockerfile podemos ver todo lo que hace, cogiendo una imagen Ubuntu de Docker Hub instala todo lo demás y la verdad es que genera una imagen que bien podría ser una OVA por el tamaño (4,36 GB), pero es lo que hay, no he entrado a comprobar si todo lo que se instala en esta imagen es realmente necesario.

```
[root@csirt02 ~]# docker images
REPOSITORY                          TAG        IMAGE ID       CREATED         SIZE
risingonline/misp                   latest     59d68fefeb59   3 weeks ago     4.36GB
risingonline/misp                   <none>     fad94356a05d   4 weeks ago     4.36GB
ubuntu                              latest     7e0aa2d69a15   5 weeks ago     72.7MB
k8s.gcr.io/kube-apiserver           v1.21.0    4d217480042e   7 weeks ago     126MB
k8s.gcr.io/kube-proxy               v1.21.0    38ddd85fe90e   7 weeks ago     122MB
k8s.gcr.io/kube-controller-manager  v1.21.0    09708983cc37   7 weeks ago     120MB
k8s.gcr.io/kube-scheduler           v1.21.0    62ad3129eca8   7 weeks ago     50.6MB
wazuh/wazuh-kibana-odfe             4.1.1      03a37501c46d   3 months ago    1.15GB
weaveworks/weave-npc                2.8.1      7f92d556d4ff   4 months ago    39.3MB
weaveworks/weave-kube               2.8.1      df29c0a4002c   4 months ago    89MB
k8s.gcr.io/pause                    3.4.1      0f8457a4c2ec   4 months ago    683kB
k8s.gcr.io/coredns/coredns          v1.8.0     296a6d5035e2   7 months ago    42.5MB
k8s.gcr.io/etcd                     3.4.13-0   0369cf4303ff   9 months ago    253MB
haproxy                             2.1.4      6600fae04efd   12 months ago   92.4MB
kubernetesui/metrics-scraper        v1.0.4     86262685d9ab   14 months ago   36.9MB
metallb/speaker                     v0.9.3     f241be9dae66   14 months ago   44.3MB
osixia/keepalived                   1.3.5-1    966bf52f7b56   3 years ago     53.4MB
```

Ilustración 52 - Listado imágenes docker local

Después me autentico en mi repositorio de Docker Hub y ejecuto un push de docker para subir la imagen.

Otra cosa importante, al igual que en las instalación de Wazuh aquí los certificados se integran en el contenedor de MISP web y por tanto hay que generar los CSR para firmarlos con mi PKI.

Ahora ya podemos ejecutar nuestra “receta”, nuestra configuración Kustomization para hacer el deploy en k8s de MISP.

```
kubectl apply -k misp
```

```
[kadmin@csirt01 manifests]$ kubectl get all -n sirp -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP          NODE                   NOMINATED NODE   READINESS GATES
pod/cassandra-0                     1/1     Running   0          11d   10.39.0.1   csirt06.cyberhome.es   <none>           <none>
pod/cortex-6bb4d47cff-plbwr         1/1     Running   0          11d   10.40.0.4   csirt07.cyberhome.es   <none>           <none>
pod/elasticsearch-0                 1/1     Running   1          19d   10.39.0.3   csirt06.cyberhome.es   <none>           <none>
pod/misp-db-0                       1/1     Running   1          22d   10.39.0.2   csirt06.cyberhome.es   <none>           <none>
pod/misp-web-7dd986bf5-hc6kx        1/1     Running   1          14d   10.36.0.4   csirt04.cyberhome.es   <none>           <none>
pod/nginx                           1/1     Running   0          11d   10.40.0.5   csirt07.cyberhome.es   <none>           <none>
pod/thehive-6dcbbf4bc9-22xnq        1/1     Running   0          11d   10.40.0.6   csirt07.cyberhome.es   <none>           <none>

NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                                    AGE   SELECTOR
service/cassandra     ClusterIP   None            <none>        7000/TCP,7001/TCP,7199/TCP,9042/TCP,9160/TCP   19d   app=cassandra
service/cortex        ClusterIP   10.109.182.177  <none>        9001/TCP                                   20d   io.kompose.service=cortex
service/elasticsearch ClusterIP   None            <none>        9200/TCP,9300/TCP                          19d   app=elasticsearch
service/misp-db       ClusterIP   None            <none>        3306/TCP                                   22d   app=misp
service/misp-web      NodePort    10.105.173.19   <none>        443:31443/TCP                              20d   app=misp-web
service/nginx         NodePort    10.99.79.244    <none>        443:32443/TCP                              20d   io.kompose.service=nginx
service/thehive       ClusterIP   10.97.179.72    <none>        9000/TCP                                   20d   io.kompose.service=thehive

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES                        SELECTOR
deployment.apps/cortex   1/1     1            1           19d   cortex       thehiveproject/cortex:latest  io.kompose.service=cortex
deployment.apps/misp-web 1/1     1            1           15d   misp-web     risingonline/misp:latest      app=misp-web
deployment.apps/thehive  1/1     1            1           19d   thehive      thehiveproject/thehive4:latest io.kompose.service=thehive

NAME                                DESIRED   CURRENT   READY   AGE   CONTAINERS   IMAGES                        SELECTOR
replicaset.apps/cortex-6bb4d47cff   1         1         1       19d   cortex       thehiveproject/cortex:latest  io.kompose.service=cortex,pod-template-hash=6bb4d47cff
replicaset.apps/misp-web-7dd986bf5  1         1         1       15d   misp-web     risingonline/misp:latest      app=misp-web,pod-template-hash=7dd986bf5
replicaset.apps/thehive-6dcbbf4bc9  1         1         1       19d   thehive      thehiveproject/thehive4:latest io.kompose.service=thehive,pod-template-hash=6dcbbf4bc9

NAME                      READY   AGE   CONTAINERS     IMAGES
statefulset.apps/cassandra 1/1    19d   cassandra      cassandra:3.11
statefulset.apps/elasticsearch 1/1 19d  elasticsearch  elasticsearch:7.11.1
statefulset.apps/misp-db  1/1     22d   misp-db        mysql/mysql-server:5.7
```

Ilustración 53 - Elementos kubernetes SIRP (incluyendo MISP)

En la imagen anterior aparecen todos los componentes instalados hasta el momento dentro del namespace sirp, en este caso los que hacen referencia a misp en su nombre son los que se han deployado tras la ejecución del comando anterior.

Como siempre hay que dar un tiempo prudencial para que todo este operativo, se puede ir revisando logs, etc. para ver que todo va funcionando correctamente, se crean los usuarios y la BBDD en mysql y se crea la configuración php de nuestro MISP.

Ahora ya podríamos validar el funcionamiento siguiendo el apartado 4.4 de este documento.

Al igual que en las configuraciones anteriores he subido a mi repositorio GIT el código yaml y scripts correspondientes.

https://github.com/risingonline19/Plataforma-SIRP/misp

Las imágenes en este caso como ya se han comentado, una es de creación propia, pero que es pública para su uso en Docker Hub y la otra se basa en una imagen estándar de mysql (una versión concreta, porque según tengo entendido a partir de cierta versión hay cambios considerables en mysql y hay software que no está preparado para funcionar que la nueva forma de conectar a la BBDD).

Anexo 5 Praeco-Elastalert en k8s

Aunque a estas alturas parezca repetitivo, pero creo que este “disclaimer” es necesario. Para poder instalar, partimos de la base de que disponemos de un entorno k8s plenamente operativo y tenemos un usuario con permisos de administración en nuestro entorno de kubernetes.

Como ya se ha tratado en apartados anteriores, la instalación se ha realizado en partes, con respecto a la plataforma SIRP, todas ellas se pueden instalar de forma independiente ya que son autocontenidas, es decir, no se relacionan con nadie, solo con los componentes que se deployan de forma conjunta usando el Kustomization creado, por ello hay algunos apartados que se repiten como puede ser la creación del namespace (que si hay existe al tener los mismos valores k8s no hace nada simplemente indica “unchanged”).

El comentario anterior viene a colación de que podemos instalar en el orden que queramos, seguir el orden que aquí hemos establecido, es el que yo he seguido en mi proyecto, pero podríamos deployar primero MISP, después Wazuh y finalmente TheHive/Cortex o en otro orden, pero el Praeco-Elastalert debería ser el último porque es un componente que enlaza con los demás ya desde la propia instalación.

Esto que comento en el párrafo anterior es como la integración que yo he elegido para TheHive y Cortex, en la propia instalación se gestiona la conexión entre ambos, lo que en el caso de TheHive significó redeployarlo debido a que hasta que Cortex no está operativo no se puede crear la API KEY y por tanto pasarla como parámetro de deploy.

Aquí podríamos hacer lo mismo, pero conviene tener disponible elasticsearch y aquí no nos vale uno propio, dado que los metadatos que sí podrían estar almacenados en uno propio pero las consultas para generar alertas no, estas requieren de nuestro SIEM de verdad y además tampoco queremos crear más pods elasticsearch sin necesidad.

Así comenzamos como siempre descargando del repositorio de Github de Praeco.

```
git clone https://github.com/johnsusek/praeco
```

Como ya nos ha pasado en casos anteriores la instalación en contenedores está preparada para ejecutarse directamente en un entorno Docker y por ello ponen a nuestra disposición un docker-compose.yml, pero como nosotros tenemos un cluster k8s necesitamos convertir esto en yaml manifests compatibles con kubernetes (si no queremos empezar desde cero, que también es una opción, no necesariamente hay que usar lo que nos dan, sino que podemos crear los manifests entendiendo como funciona la aplicación).

```
[kadmin@csirt01 praeco]$ kompose convert
WARN Volume mount on the host "/tmp/praeco/config/elastalert.yaml" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/config/api.config.json" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/rules" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/rule_templates" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/public/praeco.config.json" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/nginx_config/nginx.conf" isn't supported - ignoring path on the host
WARN Volume mount on the host "/tmp/praeco/nginx_config/default.conf" isn't supported - ignoring path on the host
INFO Kubernetes file "elastalert-service.yaml" created
INFO Kubernetes file "webapp-service.yaml" created
INFO Kubernetes file "elastalert-deployment.yaml" created
INFO Kubernetes file "elastalert-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "elastalert-claim1-persistentvolumeclaim.yaml" created
INFO Kubernetes file "elastalert-claim2-persistentvolumeclaim.yaml" created
INFO Kubernetes file "elastalert-claim3-persistentvolumeclaim.yaml" created
INFO Kubernetes file "webapp-deployment.yaml" created
INFO Kubernetes file "webapp-claim0-persistentvolumeclaim.yaml" created
INFO Kubernetes file "webapp-claim1-persistentvolumeclaim.yaml" created
INFO Kubernetes file "webapp-claim2-persistentvolumeclaim.yaml" created
```

Ilustración 54 - Ejecución Kompose Praeco-Elastalert

Vale decir, que en este caso también hay una opción de instalación vía paquetes Helm, que es un sistema de paquetes preparado para deployar aplicaciones en kubernetes, pero dado que no tenía instalado este sistema de paquetería he seguido utilizando las opciones que ya sé que me funcionan.

Una vez personalizada la instalación, ficheros de configuración apuntando a nuestro elasticsearch y algún detalle más, procedemos a aplicar la configuración sobre nuestro cluster kubernetes.

```
kubectl apply -k praeco
```

He añadido estos componentes, lo que podríamos decir backend elastalert y el frontend la webapp de praeco en el namespace sirp, así pues, ahora tenemos los siguientes elementos.

```
[kadmin@csirt02 misp_http]$ kubectl get all -n sirp -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP          NODE                  NOMINATED NODE   READINESS GATES
pod/cassandra-0                     1/1     Running   0          17d     10.39.0.1   csirt06.cyberhome.es  <none>           <none>
pod/cortex-6bb4d47cff-plbwr         1/1     Running   0          17d     10.40.0.4   csirt07.cyberhome.es  <none>           <none>
pod/elastalert-755fb89894-zpkvc     1/1     Running   0          19h     10.42.0.6   csirt05.cyberhome.es  <none>           <none>
pod/elasticsearch-0                 1/1     Running   1          25d     10.39.0.3   csirt06.cyberhome.es  <none>           <none>
pod/misp-db-0                       1/1     Running   1          28d     10.39.0.2   csirt06.cyberhome.es  <none>           <none>
pod/misp-web-7dd986bf5-hc6kx        1/1     Running   1          20d     10.36.0.4   csirt04.cyberhome.es  <none>           <none>
pod/nginx                           1/1     Running   5          16h     10.42.0.5   csirt05.cyberhome.es  <none>           <none>
pod/praeco-6897b7f4f7-dh6ml         1/1     Running   0          3d17h   10.36.0.5   csirt04.cyberhome.es  <none>           <none>
pod/thehive-7d9b459f7-2w7vx         1/1     Running   0          5d12h   10.40.0.3   csirt07.cyberhome.es  <none>           <none>

NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                                   AGE     SELECTOR
service/cassandra       ClusterIP   None             <none>        7000/TCP,7001/TCP,7199/TCP,9042/TCP,9160/TCP   25d     app=cassandra
service/cortex          ClusterIP   10.109.182.177   <none>        9001/TCP                                  26d     io.kompose.service=cortex
service/elastalert      ClusterIP   10.110.234.188   <none>        3030/TCP,3333/TCP                         3d18h   app=elastalert
service/elasticsearch   ClusterIP   None             <none>        9200/TCP,9300/TCP                         25d     app=elasticsearch
service/misp-db         ClusterIP   None             <none>        3306/TCP                                  28d     app=misp
service/misp-web        NodePort    10.105.173.19    <none>        443:31443/TCP                             26d     app=misp-web
service/nginx           NodePort    10.99.79.244     <none>        443:32443/TCP                             26d     app=nginx
service/praeco          ClusterIP   10.99.108.168    <none>        8080/TCP                                  3d17h   app=praeco
service/thehive         ClusterIP   10.99.202.37     <none>        9000/TCP                                  5d12h   app=thehive

NAME                      READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES                            SELECTOR
deployment.apps/cortex    1/1     1            1           25d     cortex       thehiveproject/cortex:latest      io.kompose.service=cortex
deployment.apps/elastalert 1/1    1            1           3d18h   elastalert   praecoapp/elastalert-server:latest app=elastalert
deployment.apps/misp-web  1/1     1            1           21d     misp-web     risingonline/misp:latest          app=misp-web
deployment.apps/praeco    1/1     1            1           3d17h   praeco       praecoapp/praeco                  app=praeco
deployment.apps/thehive   1/1     1            1           5d12h   thehive      thehiveproject/thehive4:latest    app=thehive

NAME                                  DESIRED   CURRENT   READY   AGE     CONTAINERS   IMAGES                            SELECTOR
replicaset.apps/cortex-6bb4d47cff     1         1         1       25d     cortex       thehiveproject/cortex:latest      io.kompose.service=cortex,pod-template-hash=6bb4d47cff
replicaset.apps/elastalert-755fb89894 1         1         1       19h     elastalert   praecoapp/elastalert-server:latest app=elastalert,pod-template-hash=755fb89894
replicaset.apps/misp-web-7dd986bf5    1         1         1       21d     misp-web     risingonline/misp:latest          app=misp-web,pod-template-hash=7dd986bf5
replicaset.apps/praeco-6897b7f4f7     1         1         1       3d17h   praeco       praecoapp/praeco                  app=praeco,pod-template-hash=6897b7f4f7
replicaset.apps/thehive-59f4877f66    0         0         0       5d12h   thehive      thehiveproject/thehive4:latest    app=thehive,pod-template-hash=59f4877f66
replicaset.apps/thehive-7d9b459f7     1         1         1       5d12h   thehive      thehiveproject/thehive4:latest    app=thehive,pod-template-hash=7d9b459f7
replicaset.apps/thehive-d995bf96f     0         0         0       5d12h   thehive      thehiveproject/thehive4:latest    app=thehive,pod-template-hash=d995bf96f

NAME                          READY   AGE   CONTAINERS      IMAGES
statefulset.apps/cassandra    1/1     25d   cassandra       cassandra:3.11
statefulset.apps/elasticsearch 1/1    25d   elasticsearch   elasticsearch:7.11.1
statefulset.apps/misp-db      1/1     28d   misp-db         mysql/mysql-server:5.7
```

Ilustración 55 - Elementos kubernetes SIRP (Praeco y Elastalert)

Aparte de lo que podemos ver en la imagen que son los pods, servicios, deployments y replicasets donde tenemos nuestros nuevos componentes elastalert y praeco, hay que decir que he añadido una configuración a nuestro NGINX proxy revers (el que utilizamos en TheHive/Cortex) para dar acceso seguro al Praeco, dado que por defecto utiliza un acceso HTTP por el puerto 8080, es la opción por defecto, dado que también es un NGINX y se podría configurar TLS.

Con el cambio de certificado, ahora tenemos bien por subject o alternative name thehive.cyberhome.es, cortex.cyberhome.es y praeco.cyberhome.es. En la configuración de praeco además se ha añadido un Basic Auth dado que este componente como ya se ha tratado a nivel aplicativo no tiene control de usuarios.

Finalmente, como en los casos anteriores el código yaml utilizado se ha subido a mi repositorio GIT para ser compartido, revisado y usado si procede.

https://github.com/risingonline19/Plataforma-SIRP/praeco

Anexo 6 Instalación de los agentes Wazuh

Tal como se ha indicado anteriormente se ha preparado el entorno para hacer la instalación de los agentes con Ansible, en los nodos que he utilizado para administración de kubernetes, he instalado también los paquetes del ansible core para poder ejecutar playbooks.

Adicionalmente en las máquinas Linux se ha creado el usuario ansible que tiene permisos para escalar privilegios vía sudo, se ha intercambiado una clave SSH en los servidores destino que además se limita el rango de IPs desde donde se puede invocar por temas de seguridad.


```
from="192.168.80.40/30" ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQCm4zJh8kV3FD96FfjLt6PNl1RgwLu3BkGbYDYxaZbFtlyzW2mqba/eAj5uac
x384E1R0WNFph2Nr77F1cR9VSlKPCBunA+swLlthYWQGc23JoOD53JEYe640yRCRc4zzjwfdsuqKxsBK4AkfOhgehHc
fsKHCzI4yQPybs8eU+niZzydjowYGJy3qvu9okNzTGQi6IOZmNQvglKzll4dOj1JLuFkBUaUZDuecHpXXnXcxtZyrx6qn/Ev
Y4waEl5O8XB4I8GBNIcjmkxJlUgyVhBTu1WxEesQEgKCZSfUx7QX51Ai4Euk3m4D+JaA392KFsb8FawiewupeyRSDP
APR4Y3zw7feKLigRvhlt3yohR0lp4ffEoi+/4xp1qAy2EiOzYvD0ulnKoJiscirdRjpXWBmExjYCa47tNWISLAf8BogbbtB4Anu
3OTNc0PIhhxw6UsWuOv5Xw4p6cVSpWMTGCBOhc/a9FRrQF1uRUHWW8HZPleeMpmXoIBCE7RJ3Lg9bn9TLUXnp
HNHDHgUg5Xpu658jae1fBWhOXOQx9IczTEtSFS9CcxM2p902eH8zcM4SY0Jj+DWrbEviHQdp61YK0GDCFufB81PnO
b/6qcKBh5mPMxW32RO6VbN9v/Y22P7Tf4/PH2vQSthF0WZmca0ewHWVXntonhD3EPPW+SYydfQ==
root@csirt01.cyberhome.es
```


Una vez distribuida la clave pública en los authorized_keys del usuario ansible de cada servidor Linux comprobamos que tenemos conectividad.

```
[root@csirt01 ~]# ansible linux -u ansible -m ping
ips001.myhome-inc.es | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
prx001.myhome-inc.es | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

En el caso de Windows hay que crear un usuario y el acceso no se hace por ssh, sino por winrm, pero no he llegado a configurarlo porque para hacerlo de forma segura habría que hacerlo con certificados, dado que es un usuario con privilegios de administrador al que se le permite el acceso remoto.

Finalmente, después de hacer toda la instalación y descargar los roles del repositorio de Wazuh, crear los hosts en el fichero /etc/ansible/hosts, etc. no se ha utilizado esta opción de instalación, aunque es la ideal para una plataforma/organización donde existan muchos equipos, dado que además es la tendencia actual para ahorrar costes de administración y mantenimiento de los sistemas.

El motivo de no realizar la instalación usando el sistema ansible después de implementarlo como mínimo para Linux que era la intención, es que me daba problemas, se conectaba a los equipos pero daba errores y aunque dedique un tiempo a revisar el motivo, habiendo alternativas tampoco es cuestión de centrarse en este problema por un factor de tiempo y dedicación.

Así que pasaré a describir la instalación en un equipo Linux y en un equipo Windows con una instalación manual, tal como se ha realizado en mi instalación

Windows

Empezaremos por la instalación en Windows, para ello descargaremos el paquete de instalación:

https://packages.wazuh.com/4.x/windows/wazuh-agent-4.1.5-1.msi

Una vez descargado hay varias opciones desde entorno gráfico o desde línea de comandos, he optado por la línea de comandos, así que desde una ventana cmd ejecutamos lo siguiente:

```
wazuh-agent-4.1.5-1.msi /q WAZUH_MANAGER="wazuh-manager.cyberhome.es"
WAZUH_REGISTRATION_SERVER="wazuh-manager.cyberhome.es"
WAZUH_REGISTRATION_PASSWORD="********"
```

La password por defecto es demasiado fácil es una de las que habría que cambiar (es poco sería para un producto dedicado a la seguridad).

Linux

Primero hay que habilitar los repositorios para poder descargar los paquetes, ya que por suerte tenemos disponible para la distribución que se ha instalado.

```
rpm --import https://packages.wazuh.com/key/GPG-KEY-WAZUH
```

```
cat > /etc/yum.repos.d/wazuh.repo << EOF
[wazuh]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=EL-$releasever - Wazuh
baseurl=https://packages.wazuh.com/4.x/yum/
protect=1
EOF
```

Después de habilitar el repositorio ya podemos instalar el paquete, haremos como en Windows usando variables de entorno para que el agente se instale y quede registrado correctamente en nuestro Wazuh Server.

```
WAZUH_MANAGER="wazuh-manager.cyberhome.es" WAZUH_REGISTRATION_SERVER="wazuh-
manager.cyberhome.es" WAZUH_REGISTRATION_PASSWORD="********" yum install wazuh-agent
```

Los agentes se han instalados en el servidor Windows (WINSRV001) y en el cliente (WINCLT001), así como en los Linux (IPS001 y PRX001), las máquinas que hacer de IPS y DNS, Revers Proxy, etc. en la organización ficticia MyHome Inc., no se ha instalado en el servidor OWASP-BWA porque este servidor es una OVA con muchas vulnerabilidades para explotar del Top Ten OWASP, pero está basado en un SO bastante antiguo y por tanto lo más probable es que no funcionara nuestro agente Wazuh.

El resultado de la instalación de los agentes se puede ver en las capturas de pantalla mostradas en el apartado 4.1 de este documento.

Anexo 7 Otras configuraciones

En este apartado trataremos otras configuraciones que si bien no se trata de los productos propiamente que conforman la plataforma SIRP si que dan soporte como por ej. la configuración de los balanceadores, que complementa la información de los servicios NodePort de nuestro cluster kubernetes.

Así la configuración de HAProxy es la siguiente (haproxy.cfg), es una configuración bastante simple y mejorable con “probes” más especializados en los sistemas que tiene que dar acceso y balancear.

```
#---------------------------------------------------------------------
# Example configuration for a possible web application.  See the
# full configuration options online.
#
#   https://www.haproxy.org/download/1.8/doc/configuration.txt
#
#---------------------------------------------------------------------

#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
   # to have these messages end up in /var/log/haproxy.log you will
   # need to:
   #
   # 1) configure syslog to accept network log events.  This is done
   #    by adding the '-r' option to the SYSLOGD_OPTIONS in
   #    /etc/sysconfig/syslog
   #
   # 2) configure local2 events to go to the /var/log/haproxy.log
   #   file. A line like the following can be added to
   #   /etc/sysconfig/syslog
   #
   #    local2.*                    /var/log/haproxy.log
   #
   log        127.0.0.1 local2

   chroot      /var/lib/haproxy
   pidfile     /var/run/haproxy.pid
   maxconn     4000
   user        haproxy
   group       haproxy
   daemon

   # turn on stats unix socket
   stats socket /var/lib/haproxy/stats

   # utilize system-wide crypto-policies
   ssl-default-bind-ciphers PROFILE=SYSTEM
   ssl-default-server-ciphers PROFILE=SYSTEM

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
   mode                    http
   log                     global
   option                  httplog
   option                  dontlognull
   option http-server-close
   option forwardfor       except 127.0.0.0/8
   option                  redispatch
   retries                 3
   timeout http-request    10s
```

```
   timeout queue           1m
   timeout connect         10s
   timeout client          1m
   timeout server          1m
   timeout http-keep-alive 10s
   timeout check           10s
   maxconn                 3000

#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
frontend kibana
   bind 192.168.80.201:443
   mode tcp
   default_backend wazuh-kibana

frontend wazuh
   bind 192.168.80.201:1515
   mode tcp
   default_backend wazuh-master

frontend wazuh-reporter
   bind 192.168.80.201:1514
   mode tcp
   default_backend wazuh-manager

frontend elasticsearch
   bind 192.168.80.202:9200
   mode tcp
   default_backend es-ingest

frontend nginx
   bind 192.168.80.203:443
   mode tcp
   default_backend thehive-cortex

frontend misp-web
   bind 192.168.80.204:443
   mode tcp
   default_backend misp

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend wazuh-kibana
   option      forwardfor
   mode tcp
   balance     roundrobin
      server  worker1 192.168.80.44:30443 check
      server  worker2 192.168.80.45:30443 check
      server  worker3 192.168.80.46:30443 check
      server  worker4 192.168.80.47:30443 check

backend wazuh-master
   mode tcp
   balance     roundrobin
      server  worker1 192.168.80.44:31515 check
      server  worker2 192.168.80.45:31515 check
      server  worker3 192.168.80.46:31515 check
      server  worker4 192.168.80.47:31515 check

backend wazuh-manager
   mode tcp
   balance     roundrobin
      server  worker1 192.168.80.44:31514 check
      server  worker2 192.168.80.45:31514 check
      server  worker3 192.168.80.46:31514 check
      server  worker4 192.168.80.47:31514 check

backend es-ingest
   mode tcp
```

```
   balance     roundrobin
      server  worker1 192.168.80.44:30029 check
      server  worker2 192.168.80.45:30029 check
      server  worker3 192.168.80.46:30029 check
      server  worker4 192.168.80.47:30029 check

backend thehive-cortex
   mode tcp
   balance     roundrobin
      server  worker1 192.168.80.44:32443 check
      server  worker2 192.168.80.45:32443 check
      server  worker3 192.168.80.46:32443 check
      server  worker4 192.168.80.47:32443 check

backend misp
   mode tcp
   balance     roundrobin
      server  worker1 192.168.80.44:31443 check
      server  worker2 192.168.80.45:31443 check
      server  worker3 192.168.80.46:31443 check
      server  worker4 192.168.80.47:31443 check

#---------------------------------------------------------------------
#---------------------------------------------------------------------
#---------------------------------------------------------------------
#---------------------------------------------------------------------
# info haproxy
#---------------------------------------------------------------------
listen stats
   bind *:8404
   stats enable
   stats uri /monitor
   stats refresh 5s
```

Complementando esta configuración tenemos asignados en el DNS del dominio cyberhome.es las siguientes asignaciones de registros.

wazuh-manager.cyberhome.es	A	192.168.80.201
wazuh-master.cyberhome.es	A	192.168.80.201
kibana.cyberhome.es	A	192.168.80.201
wazuh-kibana.cyberhome.es	CNAME	kibana.cyberhome.es
mail.cyberhome.es	A	192.168.10.18
thehive.cyberhome.es	A	192.168.80.203
cortex.cyberhome.es	A	192.168.80.203
misp.cyberhome.es	A	192.168.80.204
praeco.cyberhome.es	A	192.168.80.203
elasticsearch.cyberhome.es	A	192.168.80.202

Estos nombres son públicos (ya que se trata de un DNS externo a nuestro simulado entorno de Cloud), aunque en el Firewall solo se permite el acceso a las IPs conectadas por VPN como es lógico, no son servicios que tengan que estar expuestos en Internet.

Si se hubiera implementado un DNS interno también se podría haber delegado la resolución con el DNS interno implementado en el servidor AD de la empresa fictica MyHome Inc. dado que el DNS accesible público de la empresa solo publicaría los nombre que realmente tengan que ser visibles en Internet, así como el propio servicio de DNS ubicados en la máquina PRX001.myhome-inc.es

Como no es el caso comentado en el punto anterior, se han dado de alta en el /etc/hosts de los nodos de nuestro cluster Kubernetes los nombres e IPs de las máquinas de la empresa MyHome Inc. Esto principalmente era necesario para utilizar ansible con nombres de máquina y no con las IPs directamente que es una solución menos elegante, aunque finalmente no se haya utilizado el servicio de Ansible en la instalación de agentes.