**Implantación de un SOC**

Wazuh + TheHive

horizontal line

# 

**Índice**

[Resumen 4](#_arolcxe0i15c)

[Objetivo 5](#_edqhta7t76ne)

[**Alcance 6**](#_fu5dwy6o1okb)

[Inclusiones 6](#_srxvbzc2hf6r)

[Exclusiones 6](#_5t3o8leaj7t2)

[Análisis 7](#_o2whw2kn1cdo)

[Componentes Esenciales del SOC 7](#_z17kt3xccyik)

[SIEM 7](#_o7oyb7vmp2yd)

[Herramienta seleccionada: Wazuh 8](#_nllg3jvgh0wx)

[Funcionamiento 9](#_x42rqe1mj6j6)

[- EDR 10](#_24c6wibzqfwq)

[Herramienta seleccionada: WAZUH + Syslog + Misp 11](#_gwn1kpuoikk)

[- Software gestor de incidentes 11](#_99e64gybu1a8)

[Herramienta seleccionada The Hive 11](#_xqwqz6tjff5a)

[Características principales de TheHive 11](#_2mk1tr2a8p3g)

[Desarrollo del SOC 13](#_sew96gbgqs5)

[Puesta en marcha 13](#_gbaac08by9g9)

[Requisitos 13](#_izb04cmlyd7a)

[Anexos 14](#_otnajr8mtja0)

[1. Instalación de Wazuh 14](#_7nwle99mpmt2)

[Contraseña por defecto Wazuh 16](#_wovvkuhnpa7g)

[Puertos de los contenedores 16](#_8xr7pon5ybd8)

[2. Instalación del stack de TheHive 18](#_fjtreb4x68ra)

[Credenciales por defecto de TheHive 20](#_7c2thf1boxso)

[Puertos de los contenedores 20](#_hjisqoq4k8v0)

[3. Instalación del agente de Wazuh 22](#_s39cn5sfvdsr)

[Cambiar la configuración del agente 25](#_7cu691anjbsc)

[Incidencias 26](#_jd8nq6c4109n)

[El servidor deja de recibir logs de los clientes sin errores notificados. 26](#_2wjlkef9jozc)

[Solución 26](#_objd54cut7df)

[Instalación y configuración de Sysmon (XDR) 28](#_x6tov0jgtizl)

[Agente 28](#_26gm9zixwn81)

[Instalación de sysmon 28](#_k0rrzdqrckq4)

[Servidor 29](#_axnelbv8uo3i)

[Integración con Wazuh 35](#_duhdxq215533)

[Herramientas 36](#_xlvze58o6q2e)

[Archivos de configuración 36](#_gqgrhtvvsmk4)

[Comprobar conexión 44](#_5pcgeuupnbur)

[Incidentes 45](#_wzfk4wp8miht)

[Glosario 46](#_6zzcjqm098fl)

[- SOC 46](#_3ie5ycc50bc7)

[- MISP 46](#_e7fh7i6r8kul)

[**Bibliografía 47**](#_mwp4462bquj5)

# 

# Resumen

Este documento proporciona un análisis sobre qué es un SOC, abordando sus componentes esenciales. Además, se propone e implementa una variedad de herramientas de código abierto para la **detección de amenazas**, **la gestión del incidente** y el **análisis de amenazas**.

# 

*Ejemplo de SOC con tecnologías open source*

# Objetivo

El objetivo principal de este documento es proporcionar una guía práctica y sólida para aquellos que deseen establecer un SOC basado en soluciones de software de código abierto.

# Requisitos previos

Serán necesarios conocimientos básicos de **sistemas operativos y redes**. A su vez, para las guías prácticas propuestas será necesario una máquina con **docker** instalado y conocimientos básicos sobre su uso

# Plan de mejora

La seguridad de un SOC no depende de un solo software o procedimiento, sino a la agrupación de medidas afines a la seguridad. De este modo, al seguir los procesos descritos en este informe, se podrá continuar mejorando los procedimientos con:

* Medidas de seguridad perimetral
* Protección de correo electrónico.
* Conexión a api’s de servicios no contemplados (virusTotal, Misp, Slack)
* Agregación de reglas personalizadas (Recomendación: <https://github.com/socfortress/Wazuh-Rules/>)

# Componentes Esenciales del SOC

### SIEM

Un SIEM (Security Information and Event Management) es una solución de seguridad informática diseñada para recopilar, analizar y correlacionar información de eventos y datos de seguridad en tiempo real, provenientes de diversas fuentes dentro de una red o sistema de información.

Constituye una parte esencial de un centro de operaciones de seguridad (SOC).

Los SIEMs ofrecen una plataforma centralizada para monitorear la seguridad de una organización, permitiendo a los equipos de seguridad identificar y responder a posibles amenazas de manera más eficiente. Algunas de las principales funciones de un SIEM incluyen:

1. **Recopilación de datos**: Los SIEMs pueden recopilar registros (logs) de eventos y datos de múltiples fuentes, como sistemas informáticos, dispositivos de red, aplicaciones, bases de datos, firewalls, y otros dispositivos y servicios de seguridad.
2. **Normalización y correlación de datos**: Una vez recopilados, los datos se normalizan y se correlacionan para identificar patrones y tendencias que puedan indicar actividades maliciosas o anómalas. Esto implica la identificación de relaciones entre eventos aparentemente no relacionados para detectar amenazas más sofisticadas.
3. **Análisis y detección de amenazas**: Los SIEMs emplean técnicas de análisis avanzado, como la detección de anomalías y la aplicación de reglas de correlación, para identificar actividades sospechosas o potencialmente maliciosas en la red.
4. **Alertas y notificaciones**: Cuando se detecta una actividad sospechosa, el SIEM puede generar alertas y notificaciones para informar al personal de seguridad, permitiendo una respuesta rápida y eficaz a las amenazas.
5. **Generación de informes y cumplimiento normativo**: Los SIEMs pueden generar informes detallados sobre la actividad de seguridad de la red, que pueden ser útiles para la gestión de riesgos, la auditoría interna y el cumplimiento de regulaciones y normativas de seguridad.

Un algunos ejemplos de tecnologías SIEM son:

* **Wazuh (open source)**
* Splunk
* IBM QRadar
* LogRhythm
* SolarWinds Security Event Manager (SEM)
* ArcSight

#### Herramienta seleccionada: [W](https://documentation.wazuh.com/current/index.html)azuh

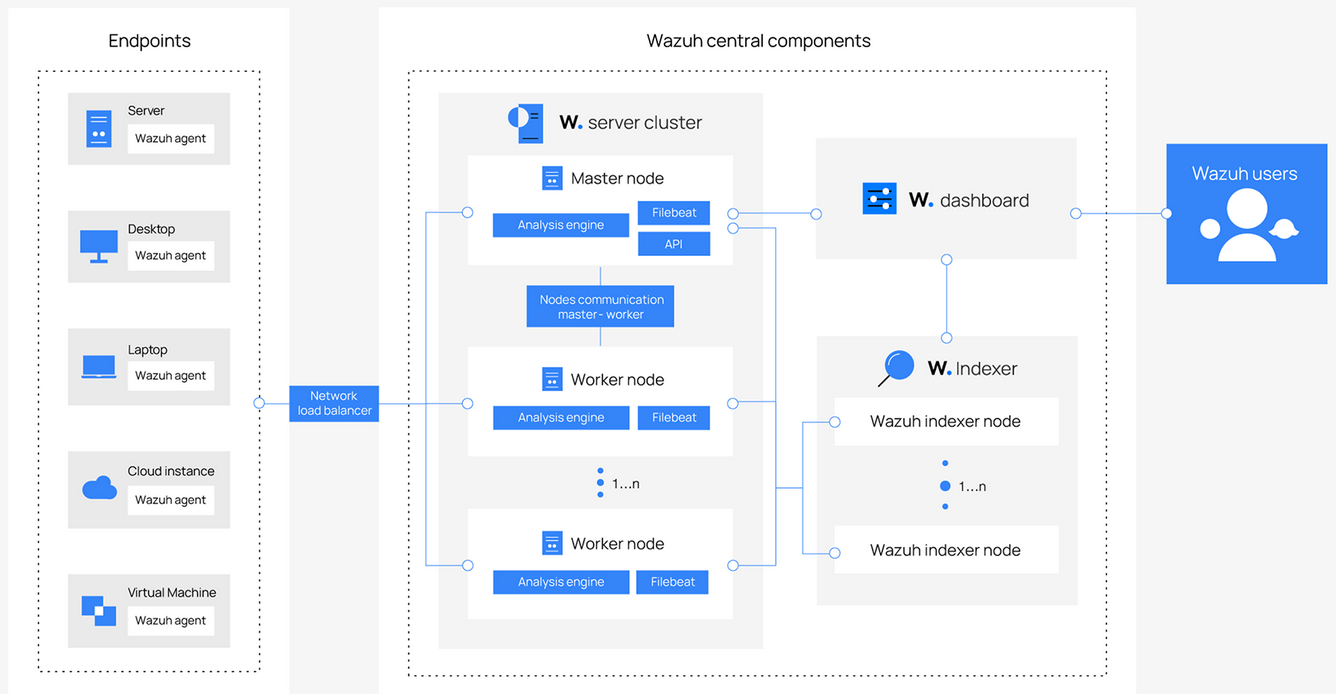
Wazuh es una plataforma de seguridad informática de **código abierto** que proporciona detección, visibilidad y respuesta a amenazas. Está compuesta por cuatro componentes principales:

* **Agente**: se instala en los sistemas a proteger, envía logs al servidor.
* **Servidor** 
  + **Manager** → Analiza los datos recibidos de los agentes. Lo procesa a través de decodificadores y reglas, usando inteligencia de amenazas para buscar indicadores bien conocidos de compromiso (COI).
  + **Dashboard** → Interfaz de usuario web para visualización y análisis de datos. Incluye paneles fuera de la caja para eventos de seguridad, cumplimiento regulatorio (por ejemplo, PCI DSS, GDPR, CIS, HIPAA, NIST 800-53), aplicaciones vulnerables detectadas, datos de monitoreo de integridad de archivos, resultados de evaluación de la configuración, eventos de monitoreo de infraestructura en la nube, y otros. También se utiliza para gestionar la configuración de Wazuh y para monitorear su estado.
  + **Indexer →** Motor de análisis y análisis altamente escalable, de texto completo. Este componente central indexa y almacena alertas generadas por el servidor Wazuh.

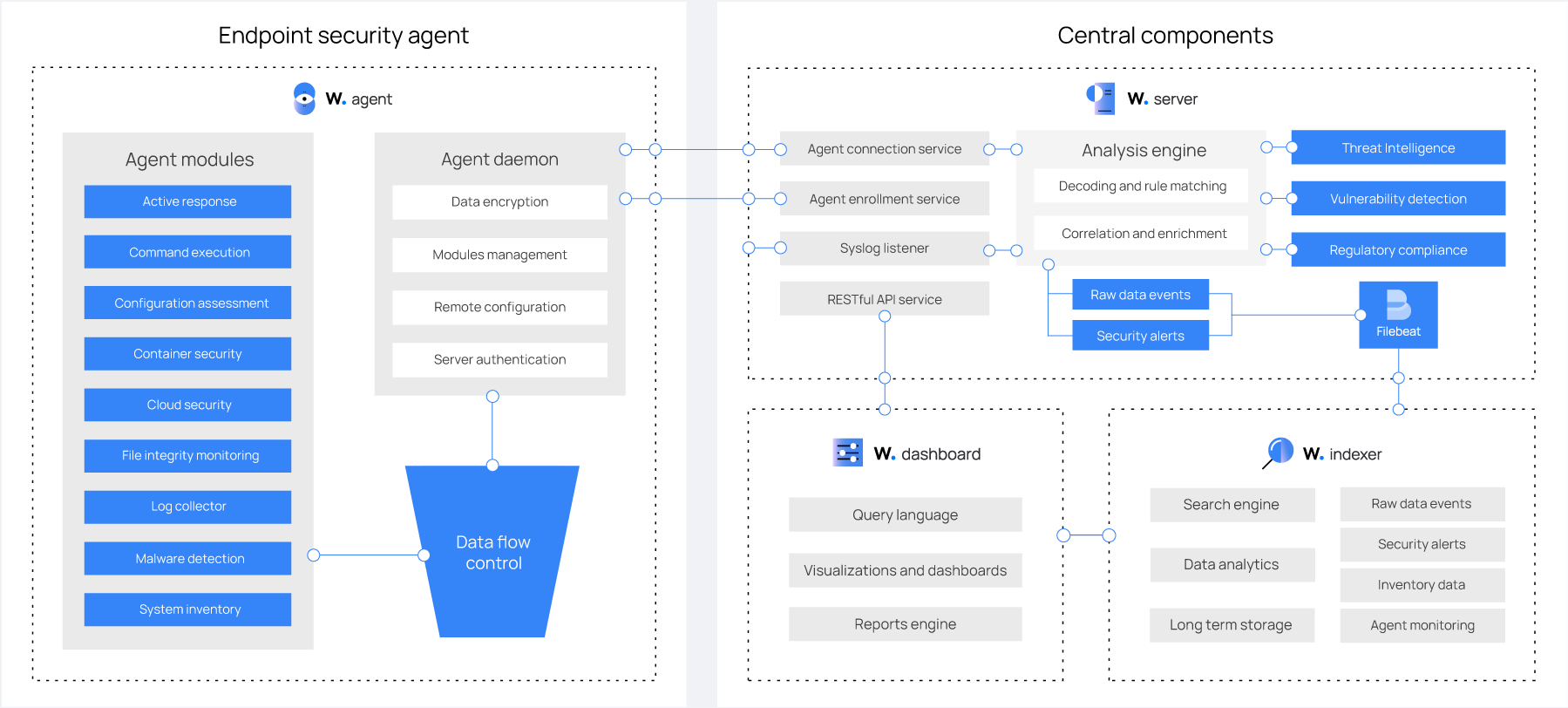
Wazuh ofrece capacidades unificadas de **XDR** (Extended Detection and Response) y **SIEM** (Security Information and Event Management). Facilita la seguridad en todas las cargas de trabajo tanto en entornos locales como en la nube, proporcionando una vista centralizada para supervisar, detectar y alertar sobre eventos e incidentes de seguridad en endpoints.

##### Funcionamiento

El agente de Wazuh se encarga de recopilar información crucial de los sistemas a proteger, incluyendo registros de eventos, datos de procesos, estado de servicios y archivos modificados. Esta información es enviada al servidor de Wazuh, donde se procesa y compara con las políticas de seguridad establecidas.

*Componentes software de Wazuh*

Ante la detección de una posible amenaza, Wazuh genera una alerta dirigida al administrador del sistema, quien puede tomar acciones para mitigar el riesgo. Además, Wazuh incorpora un sistema de correlación de eventos que permite identificar patrones de comportamiento anómalos, facilitando la detección de amenazas que podrían no ser detectadas mediante una inspección manual.

*Funcionamiento de los componentes del sistema Wazuh*

### EDR

Una herramienta **Endpoint Detection and Response** proporciona monitorización y análisis continuo del endpoint. Mientras que un antivirus se centra en la detección y prevención mediante el análisis de los hashes, un EDR monitoriza el comportamiento del software en búsqueda de actividades maliciosas y realiza acciones de contención y respuesta. Se puede resumir en un cambio del enfoque reactivo a uno proactivo, una **protección ante amenazas avanzada.**

Funcionalidades claves:

* Detección
* Contención
* Investigación
* Respuesta

Un XDR (**Extended Detection and Response)** mantiene las funciones del XDR y amplía el alcance de la detección con el fin de proporcionar detección, análisis y respuesta no solo en los endpoints, sino también en las redes, servidores, cargas de trabajo en la nube, SIEM.

#### Herramienta seleccionada: [WAZUH](https://documentation.wazuh.com/current/index.html) + Syslog + Misp

Wazuh permite realizar acciones de respuesta y contención de amenazas de manera nativa. Además, al integrar **Syslog** en el agente dota al servidor de funciones de detección de amenazas y respuesta avanzada (**XDR)**. Esto lo logra principalmente correlacionando los procesos obtenidos y consultando la correlación con la integración del MISP.

### Software gestor de incidentes

Una vez implementadas las herramientas mencionadas en una o varias organizaciones, se recibirá una gran cantidad de información que deberá ser gestionada por el equipo del Centro de Operaciones de Seguridad (SOC).

Para facilitar esta tarea, se utilizan herramientas para el seguimiento de incidentes de ciberseguridad.

Estas herramientas permiten conectar las distintas partes de SOC para lograr **el seguimiento, la trazabilidad, y el registro** de cada incidente mediante clasificación por proyectos, casos, incidentes y la definición los roles para los analistas.

#### Herramienta seleccionada The Hive

##### Características principales de TheHive

1. **Gestión de Casos**: Permite crear, asignar y seguir el progreso de los casos de seguridad, facilitando la coordinación entre los miembros del equipo.
2. **Enriquecimiento de Datos**: Se integra con diversas herramientas y servicios (como VirusTotal, MISP, Cortex) para enriquecer los datos de los incidentes con información adicional, lo que ayuda en el análisis y la toma de decisiones.
3. **Automatización**: Soporta la automatización de tareas repetitivas mediante playbooks y el uso de APIs, mejorando la eficiencia operativa.
4. **Colaboración**: Facilita la colaboración entre los miembros del equipo mediante comentarios, la asignación de tareas y el intercambio de información relevante sobre los incidentes.
5. **Alertas y Notificaciones**: Permite configurar alertas y notificaciones para asegurar que los incidentes críticos se gestionen a tiempo.
6. **Reportes y Dashboards**: Proporciona herramientas de visualización y reportes que ayudan a los equipos a tener una visión clara del estado de los incidentes y las tendencias de seguridad.

**Componentes propuestos por los desarrolladores de TheHive**

* **TheHive**: La plataforma central de gestión de incidentes.
* **Cortex**: Un motor de análisis que permite ejecutar analizadores y responder con acciones específicas a los datos recibidos. Similar a nessus, Qualys.
* **Cortex-Analyzers**: Conjunto de analizadores que pueden integrarse con Cortex para realizar diversas tareas de análisis de seguridad.

# 

# Desarrollo del SOC

### 

### Puesta en marcha

Dado que la implementación de un SOC es una tarea extensa y lograda a través de la acumulación de hitos, se propone la siguiente lista a la hora de implementarse.

* [Instalación del servicio Wazuh](#_7nwle99mpmt2)
* [Instalación de los agentes](#_s39cn5sfvdsr)
* [Instalación y configuración de Sysmon](#_k0rrzdqrckq4)
* [Instalación del stack de tecnologías TheHive](#_fjtreb4x68ra)
* [Integración Wazuh - TheHive](#_duhdxq215533)

# Anexos

| Instalación de Wazuh |
| --- |
| Clonar repositorio de Wazuh |
| git clone https://github.com/wazuh/wazuh-docker.git -b v4.7.4 |
| Entrar en la carpeta **single-node** |
| *Estructura de directorios* |
| 2. Crear llaves para las comunicaciones del servidor. El documento **generate-indexer-certs.yml** se hará cargo de esto. La documentación oficial propone la imagen 0.0.1, en este caso, se recomienda la versión 0.0.2. |
| # Wazuh App Copyright (C) 2021 Wazuh Inc. (License GPLv2)  version: '3'  services:  generator:  image: wazuh/wazuh-certs-generator:0.0.**2**  hostname: wazuh-certs-generator  volumes:  - ./config/wazuh\_indexer\_ssl\_certs/:/certificates/  - ./config/certs.yml:/config/certs.yml  environment:  - HTTP\_PROXY=YOUR\_PROXY\_ADDRESS\_OR\_DNS |
| Ejecución de las claves: |
| docker-compose -f generate-indexer-certs.yml run --rm generator |
| En caso de ya contar con claves, editar el archivo: config/wazuh\_indexer\_ssl\_certs |
| *Estructura de directorios de las claves* |
| Lanzamiento del container: |
| docker-compose up |
| Contraseña por defecto Wazuh |
| admin:SecretPassword |
| Puertos de los contenedores |
|  |
|  |
|  |
|  |

| Instalación del stack de TheHive |
| --- |
| **docker-compose.yaml** |
| version: "3"  services:  thehive:  image: strangebee/thehive:5.2  depends\_on:  - cassandra  - elasticsearch  - minio  - cortex  mem\_limit: 1500m  ports:  - "9000:9000"  environment:  - JVM\_OPTS="-Xms1024M -Xmx1024M"  command:  - --secret  - "mySecretForTheHive"  - "--cql-hostnames"  - "cassandra"  - "--index-backend"  - "elasticsearch"  - "--es-hostnames"  - "elasticsearch"  - "--s3-endpoint"  - "http://minio:9000"  - "--s3-access-key"  - "minioadmin"  - "--s3-secret-key"  - "minioadmin"  - "--s3-bucket"  - "thehive"  - "--s3-use-path-access-style"  - "--cortex-hostnames"  - "cortex"  - "--cortex-keys"  # put cortex api key once cortex is bootstraped  - "<cortex\_api\_key>"  cassandra:  image: 'cassandra:4'  mem\_limit: 1600m  ports:  - "9042:9042"  environment:  - MAX\_HEAP\_SIZE=1024M  - HEAP\_NEWSIZE=1024M  - CASSANDRA\_CLUSTER\_NAME=TheHive  volumes:  - cassandradata:/var/lib/cassandra  restart: on-failure  elasticsearch:  image: docker.elastic.co/elasticsearch/elasticsearch:7.17.12  mem\_limit: 1500m  ports:  - "59200:59200"  environment:  - discovery.type=single-node  - xpack.security.enabled=false  volumes:  - elasticsearchdata:/usr/share/elasticsearch/data  minio:  image: quay.io/minio/minio  mem\_limit: 512m  command: ["minio", "server", "/data", "--console-address", ":9090"]  environment:  - MINIO\_ROOT\_USER=minioadmin  - MINIO\_ROOT\_PASSWORD=minioadmin  ports:  - "9090:9090"  volumes:  - "miniodata:/data"  cortex:  image: thehiveproject/cortex:3.1.7  depends\_on:  - elasticsearch  environment:  - job\_directory=/tmp/cortex-jobs  volumes:  - /var/run/docker.sock:/var/run/docker.sock  - /tmp/cortex-jobs:/tmp/cortex-jobs  ports:  - "9001:9001"  volumes:  miniodata:  cassandradata:  elasticsearchdata: |
| Ejecutar: |
| docker-compose up |
| Credenciales por defecto de TheHive |
| admin@thehive.local:secret |
| Ampliar sobre la configuración inicial → [link](https://docs.strangebee.com/thehive/setup/installation/docker/#using-your-own-configuration-file) |
| Puertos de los contenedores |
|  |
|  |

| Instalación del agente de Wazuh |
| --- |
| Acceder a la sección de agentes del dashboard |
| https://<ip>/:443 |
|  |
| En este caso, la ip del servidor Wazuh corresponde a la del container de docker. |
|  |
| Seguir los pasos recomendados: |
|  |
|  |
| La instalación ha de correr en el powershell del cliente. |
|  |
| Lanzar el servicio con **NET START WazuhSvc** |
|  |
| Si la instalación ha sido exitosa el agente ya aparecerá en los agentes del dashboard. |
|  |
| Cambiar la configuración del agente |
| Buscar el agente en la barra de inicio de sesión. |
|  |
|  |
| Incidencias |
| El servidor deja de recibir logs de los clientes sin errores notificados. |
| En entornos de prueba, es más común que la **dirección ip** del servidor sea cambiada, esto supone un problema pues el cliente de Wazuh no se sincroniza. Esta situación provocará que los envíos del cliente se pierdan. |
| Solución |
| Comprobar la ip’s del servidor en la configuración del agente. |
|  |
| Solución: Configurar una ip estática para el servidor y actualizar la configuración de los agentes. |

# 

# 

| Instalación y configuración de Sysmon (XDR) |
| --- |
| Agente |
| Instalación de sysmon |
| Descargar el ejecutable → [link](https://learn.microsoft.com/es-es/sysinternals/downloads/sysmon) |
|  |
| Descargar reglas de configuración del Sysmon y colocar el archivo en la misma carpeta en la que se encuentra el anterior ejecutable. |
| Link → https://github.com/SwiftOnSecurity/sysmon-config |
|  |
| Lanzar la powershell como administrador y ejecutar la siguiente linea: |
| sysmon64.exe -i sysmonconfig-export.xml |
|  |
| Servidor |
| Desde el dashboard accedemos a la configuración: |
|  |
|  |
| Añadir la siguiente etiqueta. |
| Etiquetamos el nuevo tipo de log que nos llegará |
| <remote>  <connection>syslog</connection>  <port>514</port>  <protocol>udp</protocol>  <allowed-ips>172.17.0.0/24</allowed-ips>  <local\_ip>172.17.0.0</local\_ip>  </remote> |
| Guardar y restaurar el servidor para recibir los logs de Syslog. |

# 

# 

| **Configuración de TheHive** |
| --- |
| Actualizar base de datos de cortex  **http://localhost:9001/** |
| Acceder al contenedor de TheHive con las credenciales por defecto. |
|  |
| Crear una organización |
|  |
| Crear usuario |
|  |
|  |
| Generar un usuario. Cada usuario podrá generar un código con el que se Wazuh realizará la integración |
|  |
|  |
| Tipos de usuario: |
| **Administrador:** puede gestionar todos los objetos y usuarios globales. No puede crear casos.  **Analista:** puede gestionar casos y otros objetos relacionados (observables, tareas, etc.), incluyendo compartirlos.  **Administrador de organización:** tiene todos los permisos excepto aquellos relacionados con objetos globales.  **Solo lectura**: sin permisos. |
| Una vez añadirle contraseña y generar código API |
|  |
| Integración con Wazuh |
| **docker ps** # buscar el ID de wazuh-manager |
|  |
| Abrir una terminal interactiva con el contenedor |
| **docker exec -it 8a62321afffb bash** |
| Instalar módulo de compatibilidad de the hive |
| **/var/ossec/framework/python/bin/pip3 install thehive4py==1.8.1** |
| El contenedor carece de editor de texto y el comando ping. |
| Herramientas |
| **apt- get install vim iputils-ping -y** |
| Crear archivos de integración. Wazuh lo implementa con dos archivos, uno en python con la lógica y otro con la llamada en bash/powershell.  # se recomienda usar el código de la guía oficial y añadir las redes externas  # como las del siguiente ejemplo. |
| Archivos de configuración |
| **vim /var/ossec/integrations/custom-w2thive.py** |
| # La variable **lvl\_threshold** indica el nivel mínimo de criticidad que se enviará, se recomienda 3 o más. |
| #!/var/ossec/framework/python/bin/python3  import json  import sys  import os  import re  import logging  import uuid  from thehive4py.api import TheHiveApi  from thehive4py.models import Alert, AlertArtifact  #start user config  # Global vars  #threshold for wazuh rules level  lvl\_threshold=0  #threshold for suricata rules level  suricata\_lvl\_threshold=3  debug\_enabled = False  #info about created alert  info\_enabled = True  #end user config  # Set paths  pwd = os.path.dirname(os.path.dirname(os.path.realpath(\_\_file\_\_)))  log\_file = '{0}/logs/integrations.log'.format(pwd)  logger = logging.getLogger(\_\_name\_\_)  #set logging level  logger.setLevel(logging.WARNING)  if info\_enabled:  logger.setLevel(logging.INFO)  if debug\_enabled:  logger.setLevel(logging.DEBUG)  # create the logging file handler  fh = logging.FileHandler(log\_file)  formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')  fh.setFormatter(formatter)  logger.addHandler(fh)  *def* main(*args*):  logger.debug('#start main')  logger.debug('#get alert file location')  alert\_file\_location = args[1]  logger.debug('#get TheHive url')  thive = args[3]  logger.debug('#get TheHive api key')  thive\_api\_key = args[2]  thive\_api = TheHiveApi(thive, thive\_api\_key )  logger.debug('#open alert file')  w\_alert = json.load(open(alert\_file\_location))  logger.debug('#alert data')  logger.debug(*str*(w\_alert))  logger.debug('#gen json to dot-key-text')  alt = pr(w\_alert,'',[])  logger.debug('#formatting description')  format\_alt = md\_format(alt)  logger.debug('#search artifacts')  artifacts\_dict = artifact\_detect(format\_alt)  alert = generate\_alert(format\_alt, artifacts\_dict, w\_alert)  logger.debug('#threshold filtering')  if w\_alert['rule']['groups']==['ids','suricata']:  #checking the existence of the data.alert.severity field  if 'data' in w\_alert.keys():  if 'alert' in w\_alert['data']:  #checking the level of the source event  if *int*(w\_alert['data']['alert']['severity'])<=suricata\_lvl\_threshold:  send\_alert(alert, thive\_api)  elif *int*(w\_alert['rule']['level'])>=lvl\_threshold:  #if the event is different from suricata AND suricata-event-type: alert check lvl\_threshold  send\_alert(alert, thive\_api)  *def* pr(*data*,*prefix*, *alt*):  for key,value in data.items():  if hasattr(value,'keys'):  pr(value,prefix+'.'+*str*(key),*alt*=alt)  else:  alt.append((prefix+'.'+*str*(key)+'|||'+*str*(value)))  return alt  *def* md\_format(*alt*,*format\_alt*=''):  md\_title\_dict = {}  #sorted with first key  for now in alt:  now = now[1:]  #fix first key last symbol  dot = now.split('|||')[0].find('.')  if dot==-1:  md\_title\_dict[now.split('|||')[0]] =[now]  else:  if now[0:dot] in md\_title\_dict.keys():  (md\_title\_dict[now[0:dot]]).append(now)  else:  md\_title\_dict[now[0:dot]]=[now]  for now in md\_title\_dict.keys():  format\_alt+='### '+now.capitalize()+'\n'+'| key | val |\n| ------ | ------ |\n'  for let in md\_title\_dict[now]:  key,val = let.split('|||')[0],let.split('|||')[1]  format\_alt+='| \*\*' + key + '\*\* | ' + val + ' |\n'  return format\_alt  *def* artifact\_detect(*format\_alt*):  artifacts\_dict = {}  artifacts\_dict['ip'] = re.findall(*r*'\d+\.\d+\.\d+\.\d+',format\_alt)  artifacts\_dict['url'] = re.findall(*r*'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+',format\_alt)  artifacts\_dict['domain'] = []  for now in artifacts\_dict['url']: artifacts\_dict['domain'].append(now.split('//')[1].split('/')[0])  return artifacts\_dict  *def* generate\_alert(*format\_alt*, *artifacts\_dict*,*w\_alert*):  #generate alert sourceRef  sourceRef = *str*(uuid.uuid4())[0:6]  artifacts = []  if 'agent' in w\_alert.keys():  if 'ip' not in w\_alert['agent'].keys():  w\_alert['agent']['ip']='no agent ip'  else:  w\_alert['agent'] = {'id':'no agent id', 'name':'no agent name'}  for key,value in artifacts\_dict.items():  for val in value:  artifacts.append(AlertArtifact(*dataType*=key, *data*=val))  alert = Alert(*title*=w\_alert['rule']['description'],  *tlp*=2,  *tags*=['wazuh',  'rule='+w\_alert['rule']['id'],  'agent\_name='+w\_alert['agent']['name'],  'agent\_id='+w\_alert['agent']['id'],  'agent\_ip='+w\_alert['agent']['ip'],],  *description*=format\_alt ,  *type*='wazuh\_alert',  *source*='wazuh',  *sourceRef*=sourceRef,  *artifacts*=artifacts,)  return alert  *def* send\_alert(*alert*, *thive\_api*):  response = thive\_api.create\_alert(alert)  if response.status\_code == 201:  logger.info('Create TheHive alert: '+ *str*(response.json()['id']))  else:  logger.error('Error create TheHive alert: {}/{}'.format(response.status\_code, response.text))  if \_\_name\_\_ == "\_\_main\_\_":  try:  logger.debug('debug mode') # if debug enabled  # Main function  main(sys.argv)  except *Exception*:  logger.exception('EGOR') |
| **vim /var/ossec/integrations/custom-w2thive** |
| #!/bin/sh  # Copyright (C) 2015-2020, Wazuh Inc.  # Created by Wazuh, Inc. <info@wazuh.com>.  # This program is free software; you can redistribute it and/or modify it under the terms of GP>  WPYTHON\_BIN="framework/python/bin/python3"  SCRIPT\_PATH\_NAME="$0"  DIR\_NAME="$(cd $(dirname ${SCRIPT\_PATH\_NAME}); pwd -P)"  SCRIPT\_NAME="$(basename ${SCRIPT\_PATH\_NAME})"  case ${DIR\_NAME} in  \*/active-response/bin | \*/wodles\*)  if [ -z "${WAZUH\_PATH}" ]; then  WAZUH\_PATH="$(cd ${DIR\_NAME}/../..; pwd)"  fi  PYTHON\_SCRIPT="${DIR\_NAME}/${SCRIPT\_NAME}.py"  ;;  \*/bin)  if [ -z "${WAZUH\_PATH}" ]; then  WAZUH\_PATH="$(cd ${DIR\_NAME}/..; pwd)"  fi  PYTHON\_SCRIPT="${WAZUH\_PATH}/framework/scripts/${SCRIPT\_NAME}.py"  ;;  \*/integrations)  if [ -z "${WAZUH\_PATH}" ]; then  WAZUH\_PATH="$(cd ${DIR\_NAME}/..; pwd)"  fi  PYTHON\_SCRIPT="${DIR\_NAME}/${SCRIPT\_NAME}.py"  ;;  esac  ${WAZUH\_PATH}/${WPYTHON\_BIN} ${PYTHON\_SCRIPT} $@ |
| Cambiar los permisos y propiedad de los archivos |
| chmod 755 /var/ossec/integrations/custom-w2thive.py  chmod 755 /var/ossec/integrations/custom-w2thive  chown root:wazuh /var/ossec/integrations/custom-w2thive.py  chown root:wazuh /var/ossec/integrations/custom-w2thive |
| Editar la configuración general, se recomienda usar el dashboard para aprovechar el validador de xml. |
|  |
| <integration>  <name>custom-w2thive</name>  <hook\_url>http://**thehive-thehive-1**:9000</hook\_url>  <api\_key>***codigo del usuario de TheHive***</api\_key>  <alert\_format>json</alert\_format>  </integration> |
| **thehive-thehive-1** → nombre del contenedor de thehive. Dentro de la red de docker, se resuelve estos nombre. |
| Obtención del nombre: |
| **docker ps**; #Seleccionar el ID del contenedor de thehive |
| **docker inspect <ID> | grep Name** |
|  |
| Comprobar conexión |
| Para asegurar que las redes tienen conexión, lanzar un ping **dentro** de la máquina Wazuh a el anterior nombre. |
|  |
| Guardar la configuración y reiniciar el manager. |
|  |
|  |
| Incidentes |
| * No se reconocen las redes.   **Solución**: realizar la comprobación y editar el docker-compose si procede.   * Se han corregido errores, todo ha sido solventado pero no funciona como se espera ni se muestran logs de error.   Aparentemente, se pueden guardar configuraciones erróneas en los volúmenes de de docker.  **Solución**: en la máquina huésped, desarmar ambos contenedores y borrar los volúmenes:  **docker-compose down -v**  Repetir la instalación.   * Integración de Wazuh es borrada en ossec.conf   **Solución**: aunque se encuentre contemplado en los volúmenes, al volver a levantar el contenedor borra. Esperando respuesta de servicio técnico de Wazuh. |
|  |
|  |

# 

# Glosario

### SOC

Un Centro de operaciones de seguridad (SOC, por sus siglas en inglés), a veces denominado Centro de operaciones de seguridad de la información (ISOC). Es un sistema dentro de una organización que emplea personas, procesos y tecnología para monitorear y mejorar continuamente la postura de seguridad de una organización a la vez que previene, detecta, analiza y responde a incidentes de ciberseguridad.

### MISP

Malware Information Sharing Platform ([MISP](https://www.misp-project.org/)) es una herramienta de código abierto que facilita el análisis, gestión y compartición de información relacionada con malware. Con MISP, equipos de seguridad y compañías asociadas pueden intercambiar descubrimientos sobre diferentes muestras de malware, permitiendo así mejorar sus motores antivirus y aumentar la capacidad de detección. Es una herramienta esencial para cualquier SOC ya que amplía y actualiza los datos de nuevas amenazas de forma automática.

La plataforma MISP permite agregar y compartir eventos de seguridad de manera sencilla. Ofrece funciones que permiten controlar con quién y hasta qué nivel se comparte la información. Además, incluye más de 60 bases de datos predefinidas, evitando la necesidad de comenzar desde cero en el proceso de inteligencia de amenazas.

# Bibliografía

* <https://socfortress.medium.com/worlds-best-free-siem-stack-series-compilation-75f12abff21e>
* <https://github.com/TheHive-Project/TheHive?tab=readme-ov-file>
* <https://www.maquinasvirtuales.eu/crear-soc-mediante-herramientas-opensource/>
* [Container docker de The Hive.](https://docs.strangebee.com/thehive/setup/installation/docker/#quick-start)
* [Reglas Wazuh open source](https://github.com/socfortress/Wazuh-Rules/tree/main)
* [Artículo de contexto](https://medium.com/@ehztech/wazuh-75731fcff4ab)
* [Documentación de Wazuh](https://documentation.wazuh.com/current/index.html)
* <https://www.maquinasvirtuales.eu/implementar-soc-instalar-thehive-cortex-y-misp/>
* [Uso de Wazuh y TheHive para análisis de amenazas](https://wazuh.com/blog/using-wazuh-and-thehive-for-threat-protection-and-incident-response/)