



Seguridad en docker

Francisco Peña Luque,
Curso 2022 / 2023 I.E.S. Medina Azahara de
Córdoba,
C.F.G.S. Administración de Sistemas Informáticos en
Red



Copyright © 2023, Francisco Peña Luque

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3



Índice

| | |
|--|-----------|
| 1. Introducción | 5 |
| 1.1 Contexto y justificación del Proyecto | 5 |
| 1.2 Objetivos del Proyecto | 5 |
| 1.3 Enfoque y método seguido | 5 |
| 1.4 Planificación del Proyecto | 6 |
| 1.4.2 Tareas | 6 |
| 1.4.3 Planificación semanal | 7 |
| 2. Entorno de pruebas | 7 |
| 2.1 Subsistema de Linux para Windows (WSL) | 7 |
| 2.1.1 La integración entre datos | 8 |
| 2.1.2 Procesos | 9 |
| 2.2 Configuración e instalación de WSL 2 | 11 |
| 2.3 Instalación de docker y extensiones para Visual Studio | 14 |
| 2.4 Wazuh como HIDS | 17 |
| 2.4.1 Mono-nodo | 17 |
| 2.4.2 Multi-nodo | 17 |
| 2.4.3 Instalación de Wazuh | 17 |
| 2.5 Instalación de dependencias | 19 |
| 3. Seguridad en contenedores docker | 20 |
| 3.1 Recomendaciones generales en docker | 20 |
| 3.2 CIS Docker Benchmark | 21 |
| 3.3 Componentes clave para la creación de contenedores | 22 |
| 3.3.1 Espacio de nombres | 23 |
| 3.3.2 Grupos de control | 25 |
| 3.3.3 Control de Acceso Obligatorio | 25 |
| 3.3.4 Union System File | 26 |
| 3.5 Alternativas a docker | 28 |
| 3.5.1 Plataformas alternativas | 28 |
| 3.5.2 Computación de alto rendimiento | 29 |
| 3.6 Sistemas de detección de intrusos (HIDS) | 31 |
| 3.6.1 Wazuh | 31 |
| 3.6.2 Sysdig Falco | 32 |
| 3.3.3 Docker Bench para seguridad | 32 |
| 3.3.4 OSSEC | 32 |
| 3.3.5 Anchore Engine | 32 |
| 4. Funcionalidades de los agentes de Wazuh | 33 |
| 4.1 Respuesta activa | 33 |
| 4.2 Monitorización sin agente | 34 |
| 4.3 Detección de malware | 35 |
| 4.4 Monitorización de comandos | 36 |



| | |
|--|-----------|
| 4.5 Supervisión de la integridad de los archivos | 36 |
| 4.6 Recopilación de datos de registro | 37 |
| 4.7 Supervisión de políticas de seguridad | 37 |
| 4.8 Monitorización de llamadas al sistema | 37 |
| 4.9 Integración con VirusTotal | 38 |
| 5. Marco de referencia | 39 |
| 6. Conclusión | 39 |
| 7. Referencias | 39 |



FICHA DEL PROYECTO FINAL

| | |
|---|--|
| Título del trabajo: | Seguridad en docker |
| Nombre del autor: | Francisco Peña Luque |
| Fecha de entrega (mm/aaaa): | 05/2023 |
| Área del Trabajo Final: | Ciberseguridad en contenedores |
| Ciclo Grado Superior: | Administración de Sistemas Informáticos en Red |
| Resumen del Trabajo (máximo 250 palabras): | |
| <p>Los contenedores se han convertido en una opción preferida en lugar de las máquinas virtuales debido a su mejor rendimiento y menor consumo de recursos, entre otras razones. Sin embargo, la seguridad plantea los mayores desafíos, especialmente en lo que respecta al aislamiento adecuado. Para mitigar estos problemas, se requiere un conjunto complejo y heterogéneo de medidas adicionales, como el control de acceso MAC, la segregación de privilegios mediante capacidades y el filtrado de llamadas al núcleo con seccomp. Estos mecanismos son complejos y requieren un conocimiento detallado de la tecnología subyacente.</p> <p>Los proveedores de plataformas como servicios comerciales tienen una ventaja en términos de economía de escala para abordar estos desafíos. Sin embargo, en entornos de instalaciones privadas de pequeña escala, la situación es diferente. Este trabajo se centra en la seguridad en este segundo contexto, con el objetivo de determinar si la ejecución de contenedores en una infraestructura de tamaño reducido puede mantener un nivel aceptable de seguridad.</p> | |



1. Introducción

1.1 Contexto y justificación del Proyecto

En la actualidad, la seguridad informática se ha convertido en una necesidad prioritaria para cualquier organización que utilice sistemas informáticos, sin embargo, la seguridad en Docker puede ser un desafío debido a la naturaleza de los contenedores y la forma en que interactúan con el sistema operativo. En este contexto, es importante explorar y evaluar soluciones de seguridad efectivas para proteger los contenedores y los datos que contienen. Para realizar pruebas y establecer vulnerabilidades, es conveniente tener un entorno de pruebas rápido y estable que nos permita ejecutar docker con varios contenedores ejecutándose simultáneamente sin perder muchos recursos del sistema, por ello emplearemos Visual Studio Code junto a WSL2 para realizar simulaciones de ataques a un servidor Linux y aplicar la configuración de un HIDS para ver su efectividad ante un registro no previsto. Los HIDS son una herramienta de seguridad de host ampliamente utilizada que puede ser integrada con Docker para mejorar la seguridad en su entorno de contenedores. La función de un HIDS, en este contexto, es monitorear continuamente la actividad de los contenedores, comparando los eventos con una base de datos de patrones de comportamiento sospechoso o generando tus propias reglas para alertar a los administradores de seguridad cuando se detecta una actividad que puede representar una amenaza para el sistema. Esto permite a los administradores de sistemas tomar medidas preventivas o actuar al momento de recibir un ataque.

1.2 Objetivos del Proyecto

- Ejecutar un entorno de pruebas aislado y eficiente
- Aprender el funcionamiento de aislamiento en contenedores.
- Ver vulnerabilidades de docker.
- Analizar las recomendaciones de docker sobre seguridad.
- Implementar un HIDS para securizar la virtualización de los contenedores.

1.3 Enfoque y método seguido

Ejecutaremos un entorno de pruebas en WSL con docker en el que realizaremos ataques a un contenedor para verificar la eficacia teórica de un HIDS a la hora de proteger un sistema virtualizado en docker, una vez realizado el test, podríamos implementar el proyecto en un entorno físico y determinar el coste en una infraestructura de red en caso necesario.



La decisión de usar WSL viene dada por la compatibilidad de docker con los sistemas linux, sumado a la facilidad de uso y comprensión a la hora de usar docker con Visual Studio Code. Esto supone una comodidad mayor para aquellos que deseen realizar pruebas en un entorno aislado a su sistema principal, en este caso Windows, que es aclamado por la mayoría por su entorno de escritorio, además que mediante este método, docker usa menos recursos de sistema. Los cuales puedes determinar en su configuración (núcleos, hilos, RAM, etc...)

1.4 Planificación del Proyecto

En cuanto a los recursos necesarios, solo nos bastará un equipo de gama media, aunque es posible que el entorno de pruebas funcione en otros con menos recursos, ya que vamos a implementar docker en WSL en Windows. En mi caso, poseo las siguientes características fundamentales para la virtualización en mi sistema:

- Ryzen 5 5625U (6 núcleos, 12 hilos)
- 16 GB RAM 3200Mhz (docker solo usara 3GB Máximo)

1.4.2 Tareas

La planificación del proyecto incluirá las siguientes fases en las que se dividirán las tareas a realizar para llevar a cabo la implementación y las pruebas a realizar.

Investigación inicial: se buscará información sobre el funcionamiento de docker y cómo este se puede implementar en un entorno cómodo para la manipulación de imágenes y contenedores.

Configuración de entorno de pruebas: se instalará WSL2 junto a docker en un sistema Windows como entorno de desarrollo, usaremos Visual Studio Code con una serie de extensiones que nos permitirán conectarnos a docker a través de WSL2 donde se virtualiza los contenedores.

Identificación de vulnerabilidades: seguiremos las practicas recomendadas establecidas por el Centro de Seguridad de Internet (CIS) para determinar que vulnerabilidades deberiamos tratar en la virtualizacion basada en contenedores.

Implementación de Wazuh en docker: estableceremos una infraestructura de servidores que simulan una página web con una base de datos en las que se implantara un HIDS para monitorear posibles ataques varios al servidor.

Evaluación de los ataques: se realizarán varias pruebas de distintas maneras en las que se puede ver afectada nuestra web para analizar la eficacia del IDS establecido.

Documentación y presentación : esta tarea es progresiva desde el inicio de la actividad, pero los datos y la información será tratada con mayor precisión en la última semana donde se realizará la documentación del proyecto final.

1.4.3 Planificación semanal

He establecido el siguiente cronograma para determinar el tiempo estimado para cada actividad que considero adecuado para cada tarea de las mencionadas anteriormente.



Tabla 1: Diagrama de Ganz

2. Entorno de pruebas

Algunas herramientas utilizadas durante el proceso de desarrollo sólo están disponibles o se ejecutan mejor en una plataforma determinada, y transferir datos de un sistema a otro para visualizarlos o compartirlos puede resultar tedioso. Windows Subsystem Linux resuelve este problema con una función llamada interoperabilidad. La interoperabilidad, en este caso, es la capacidad de ejecutar de forma transparente comandos y aplicaciones, compartir archivos, variables de red y de entorno entre ambos sistemas sin tener que configurar previamente para que puedan intercambiar información. La implementación de WSL con docker como entorno de desarrollo o testeo en una empresa, puede suponer un aumento de la productividad y eficacia a la hora de resolver problemas sin un mayor gasto de tiempo y energía dado las capacidades de esta implementación.

2.1 Subsistema de Linux para Windows (WSL)

En primer lugar, WSL es una característica de Windows 10 que permite ejecutar un entorno Linux. Esto se logra mediante la utilización de una capa de compatibilidad

entre los subsistemas constituido por una serie de procesos y la implementación de hyper-v, por lo que será más veloz que una máquina virtual. En cambio, una máquina virtual es un hipervisor de tipo 2 que ejecuta un sistema operativo completo, lo que implica una sobrecarga adicional y una mayor utilización de recursos. En cuanto a las ventajas de WSL sobre una máquina virtual en un hipervisor de tipo 2, se pueden mencionar las siguientes:

1. **Mayor eficiencia y mejor rendimiento:** como se mencionó anteriormente, WSL2 utiliza una capa de compatibilidad y un kernel compartido, lo que lo hace más eficiente y más rápido que una máquina virtual completa.
2. **Integración con el sistema operativo host:** se integra de manera nativa con el sistema operativo host de Windows, lo que permite el acceso directo a los recursos del host, como el sistema de archivos y la red.
3. **Menor uso de recursos:** utiliza menos recursos que una máquina virtual, lo que lo hace más liviano y rápido de ejecutar.

El siguiente esquema muestra de manera general la estructura de WSL2:

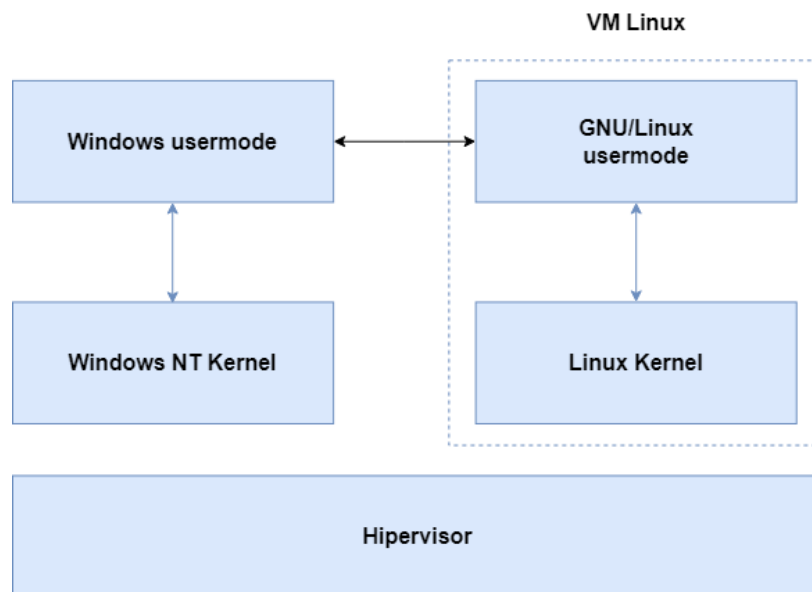


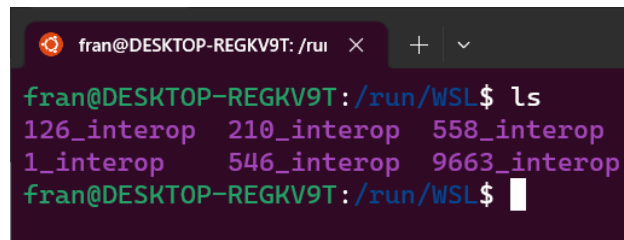
Figura 1: Vista general de WSL2

Tanto WSL2 como una máquina virtual en un hipervisor de tipo 2 se permite ejecutar un sistema operativo dentro de otro. Sin embargo, existen diferencias significativas en su funcionamiento y uso. El esquema previamente mostrado de la figura 1 se puede desglosar en las siguientes partes que componen la estructura de wsl2:

2.1.1 La integración entre datos

Mediante un sistema de archivos virtualizado (VHDX) se almacenan los datos de la distribución de Linux. Este sistema se monta como un disco duro virtual dentro del hipervisor Hyper-V. Particularmente, el sistema de archivos de Linux accede a /mnt/c para montar la partición de Windows. Por otra parte, el sistema de archivos de Windows accede a los ficheros de Linux como si fuera un almacenamiento compartido en red.

Los archivos "interop" se encargan de facilitar el acceso a los recursos de ambos entornos.



```
fran@DESKTOP-REGKV9T: /run/wsl$ ls
126_interop  210_interop  558_interop
1_interop    546_interop  9663_interop
fran@DESKTOP-REGKV9T: /run/wsl$
```

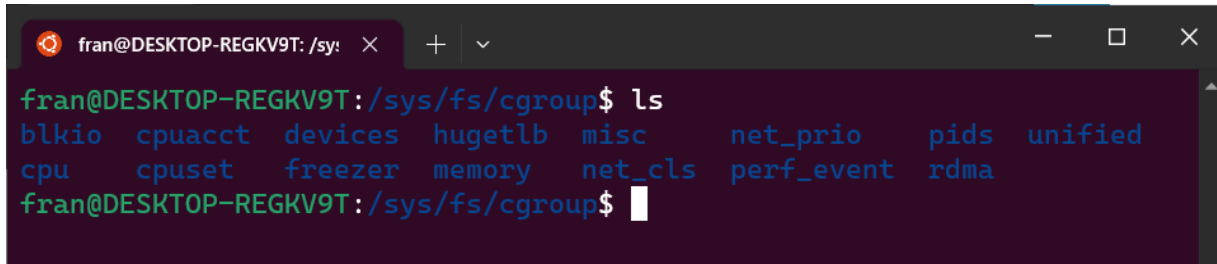
Captura 1: Archivos de interoperabilidad

Esto supone que Linux puede aceptar llamadas a las APIs de Windows, haciendo posible la integración de aplicaciones y servicios de manera bidireccional. Esto último, se usará para para realizar las pruebas con docker, se ejecutará el servicio docker en la máquinas host y el subsistema usará el motor de docker instalado para gestionar contenedores en el kernel de linux, en el que Visual Studio nos servirá de interfaz para ver las pruebas más claras.

2.1.2 Procesos

WSL: Se utiliza para interactuar con la distribución de Linux en ejecución en la máquina virtual subyacente en Windows Subsystem for Linux (WSL). Permite iniciar, detener y administrar la distribución de Linux, así como ejecutar comandos y programas en ella. Actúa como una capa de traducción entre comandos

Lxss Manager Service: Es responsable de iniciar y administrar las distribuciones de Linux que se ejecutan en WSL2, crea la instancia de la máquina virtual de Linux y establece la conexión entre la distribución de Linux y el VM Worker Process. Además, administra los recursos asignados a cada distribución, como la memoria y el número de núcleos de CPU disponibles.



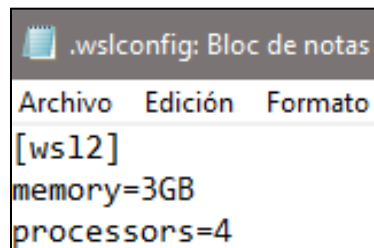
```

fran@DESKTOP-REGKV9T: /sys/fs/cgroup$ ls
blkio  cpuacct  devices  hugetlb  misc      net_prio  pids  unified
cpu    cpuset   freezer  memory   net_cls   perf_event  rdma
fran@DESKTOP-REGKV9T: /sys/fs/cgroup$

```

Captura 2: Archivos de recursos del sistema

La gestión dinámica de la memoria utilizada por el administrador de recursos se implementa mediante cgroups, un componente del kernel de Linux. Los cgroups se organizan en `/sys/fs/cgroup` y se configuran y administran a través del archivo `.wslconfig`, que es utilizado por el Lxss Manager para establecer la configuración de cada distribución de Linux. Se debe establecer en la ruta del usuario `"C:\Users\NOMBRE_USUARIO"`. Se pueden configurar distintas variables, pero las mínimas recomendables son:

Captura 3: Configuración `.wslconfig`

Estos parámetros, también se aplican a la memoria y número de procesadores permitidos para docker una vez lo configuremos con WSL2, ya que se ejecuta en la instancia de Linux, consume la RAM asociada a los grupos de control establecidos.

Host Compute Service (HCS): Es un servicio que se encarga de la gestión de contenedores y la virtualización proporcionando una interfaz (API) para que las aplicaciones y herramientas de contenedores puedan interactuar con la virtualización de Windows, incluyendo la máquina virtual de Linux subyacente que se ejecuta en el Hipervisor.

VM Worker: Se encarga de la gestión de la virtualización necesaria para ejecutar las distribuciones de Linux dentro de WSL. Controla la creación, administración y cierre de las instancias de las distribuciones de Linux, y también se encarga de la comunicación entre los procesos de Windows y los procesos de Linux que se ejecutan dentro de WSL, existen tantos procesos VM Worker como instancias de Linux tengas ejecutando simultáneamente.

La implementación de docker en este sistema se lleva a cabo mediante las extensiones y APIs que se instalan desde Visual Studio Code a WSL2. Para iniciar la

estancia, debemos ejecutar docker en Windows para que los comandos de docker y los contenedores se ejecuten en la máquina virtual ligera de linux, desde docker desktop podemos monitorizar los contenedores y el uso de recursos del sistema además de los logs de cada servicio lanzado en cada contenedor. Cuando se utiliza VS Code en conjunto con WSL2, la interfaz gráfica de se ejecuta en Windows, mientras que el código y los procesos de la aplicación se ejecutan en WSL. Para depurar aplicaciones que se ejecutan en WSL se necesita conectarse a un servidor de depuración que se ejecuta en el subsistema de Linux. Para facilitar esta comunicación, Visual utiliza un "Virtual Socket"(vsock), que proporciona un canal de comunicación de red seguro y eficiente entre Windows y WSL. Virtual Socket es una característica de Linux que permite a los procesos de Linux en diferentes máquinas virtuales comunicarse directamente entre sí utilizando un socket virtual para establecer una conexión directa entre el servidor de depuración en el subsistema y la interfaz gráfica de Visual lo que permite una experiencia de depuración fluida y sin problemas.

La siguiente imagen muestra la estructura de la arquitectura de WSL:

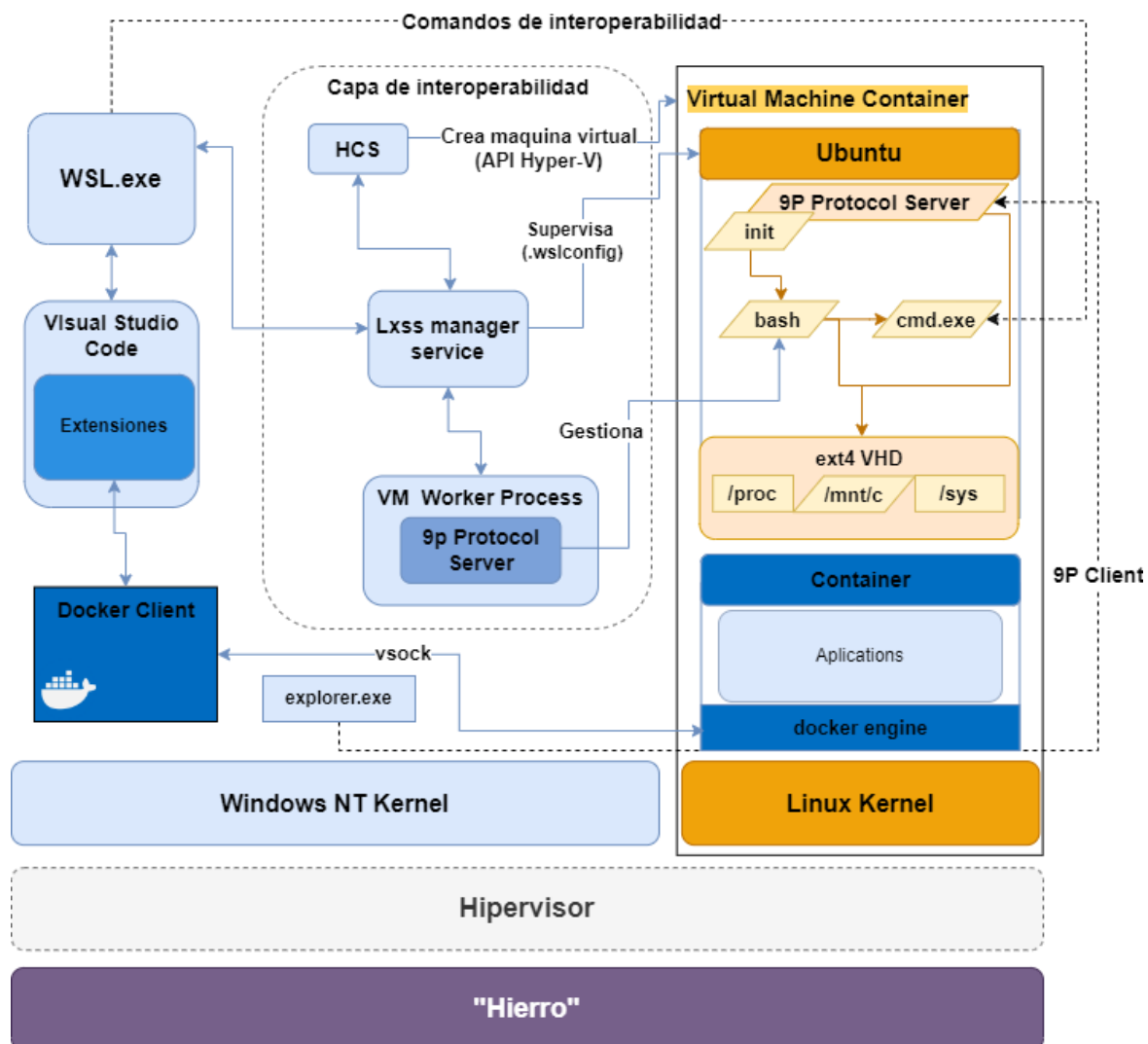
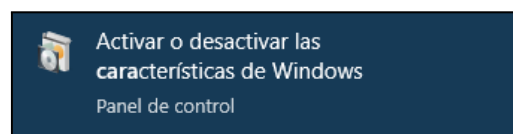


Figura 2: Partes de la arquitectura de WSL2 con docker

2.2 Configuración e instalación de WSL 2

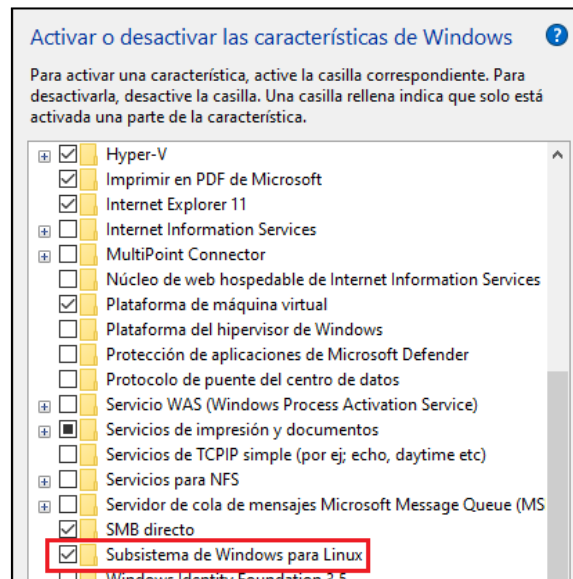
Como hemos mencionado anteriormente, para el entorno de pruebas, se utilizará WSL2 (Windows Subsystem for Linux), se instalará la distribución Ubuntu 22.04 LTS y se configurará para su uso con Docker. Además, usaremos Visual Studio Code para acceder de manera gráfica a los archivos de configuración, imágenes y datos de los contenedores para su mayor comprensión.

Debemos activar ciertas características de Windows que nos permita ejecutar este subsistema, podemos encontrar esta función en el panel de control o bien escribirla en el menú de búsqueda de la barra de tareas.



Captura 4: Instalación de WSL2

En características de Windows deberemos marcar la opción “Subsistema de Windows para Linux” y “Virtual Machine Platform”, en caso de que no aparezca esta última la activaremos a continuación mediante un comando.



Captura 5: Instalación de WSL2

Esperaremos a que finalice el proceso de instalación y negaremos el reinicio del sistema, lo dejaremos para el final. Mediante el siguiente comando podremos habilitar la característica previamente mencionada en caso de que no nos aparezca:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

```
PS [redacted] :dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Herramienta Administración y mantenimiento de imágenes de implementación
Versión: 10.0.19041.844

Versión de imagen: 10.0.19045.2965

Habilitando características
[=====100.0%=====]
La operación se completó correctamente.
```

Captura 6: Instalación de WSL2

Como recomendación, haré uso del “Windows terminal” para acceder a la consola de Linux, aun así el propio Visual Studio Code trae un terminal incorporado.



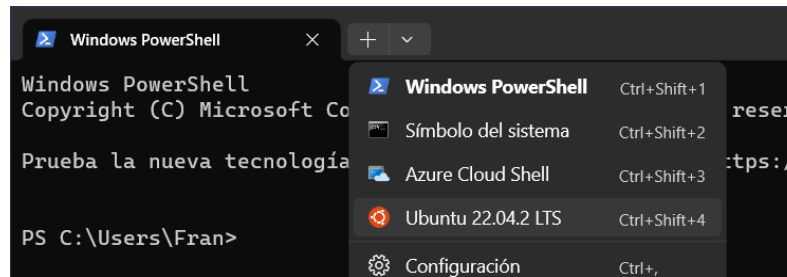
Captura 7: Instalación de WSL2

Por último, Ubuntu 22.04 LTS como es en mi caso, hay varias distribuciones de Linux para implementar.



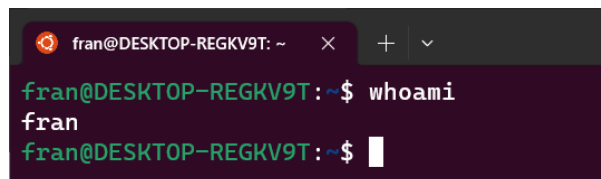
Captura 8: Instalación de WSL2

Una vez instalado todo, reiniciamos el sistema. Abrimos “Windows Terminal” desde el panel de inicio y nos debería aparecer en la ventana desplegable varias opciones entre ellas Ubuntu 22.04



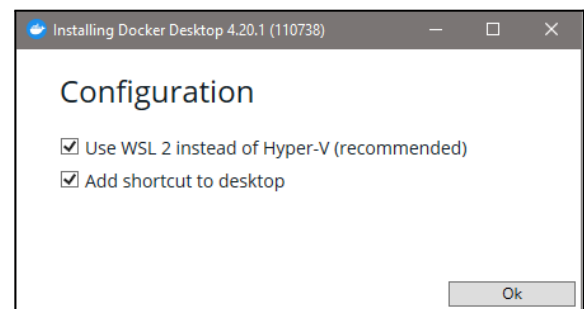
Captura 9: Instalación de WSL2

Entraremos en la distribución y comprobaremos que carga correctamente, podemos realizar algún que otro comando para verificar que lo interpreta correctamente.



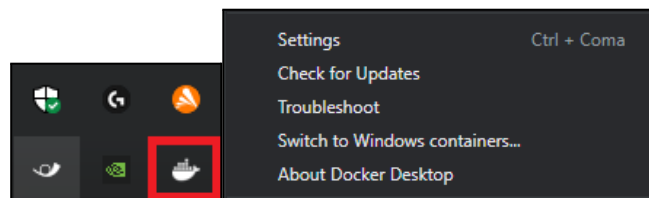
2.3 Instalación de docker y Captura 10: Instalación de WSL2I Studio

Desde docker.com, la página oficial de docker, descargamos y ejecutamos el archivo .exe. Mantenemos habilitadas ambas casillas y procederá la instalación



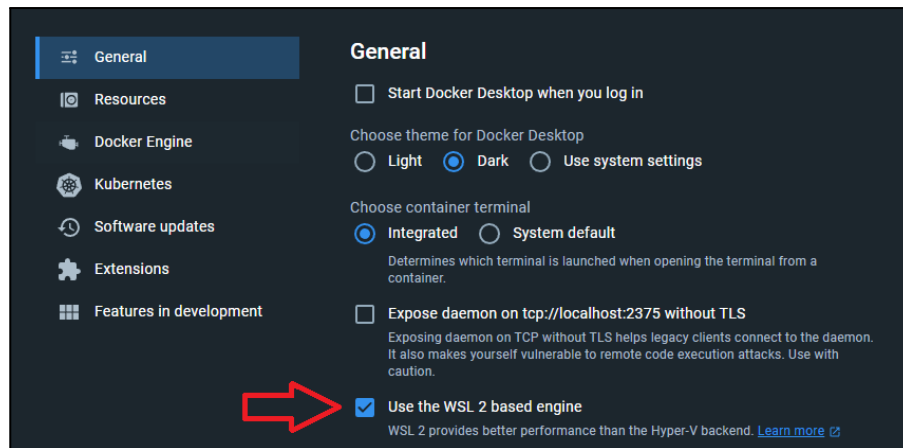
Captura 11-12: Instalación de docker

Abre Docker Desktop desde el menú Inicio , haz clic derecho en el ícono de Docker en la bandeja del sistema de Windows y selecciona "Settings".



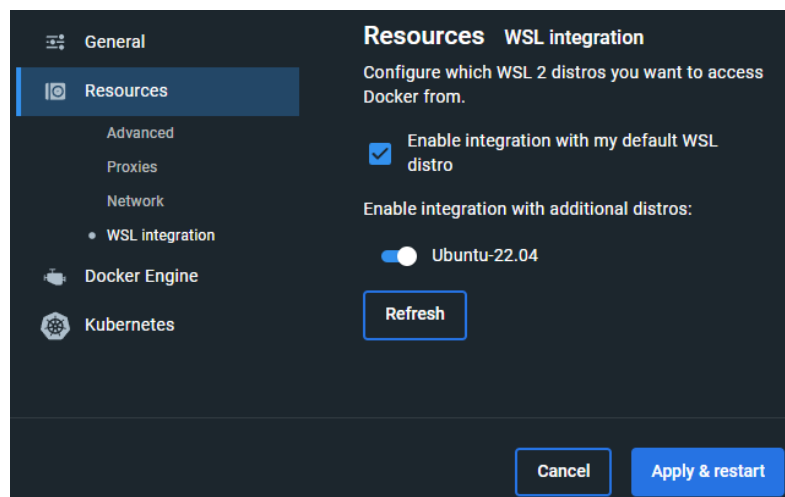
Captura 13: Ajustes de docker

En la pestaña "General", asegúrate de que la opción "Use the WSL 2 based engine" esté seleccionada.



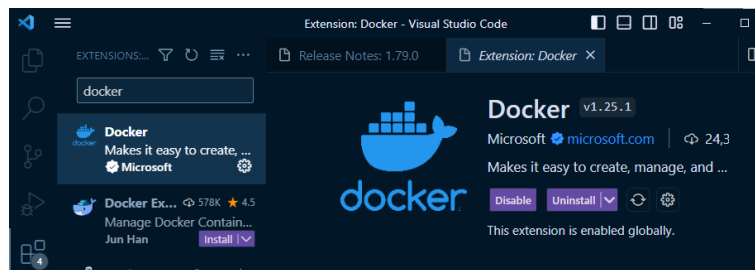
Captura 14: Configuración de docker

Luego, en la pestaña "Resources > WSL Integration", marca la casilla correspondiente a tu distribución de Linux, en este caso Ubuntu 22.04. Finalmente, haz clic en "Apply & Restart" para guardar los cambios y reiniciar Docker Desktop.



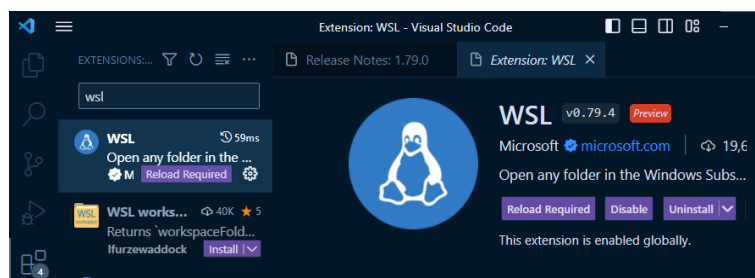
Captura 15: Habilitar distribución en docker

Por último, instala las extensiones de Docker y WSL para Visual Studio. Abre Visual Studio, haz clic en "Extensions" en la barra de menú superior y selecciona "Manage Extensions". Busca la extensión llamada "Docker" y haz clic en "Download" para iniciar la instalación.



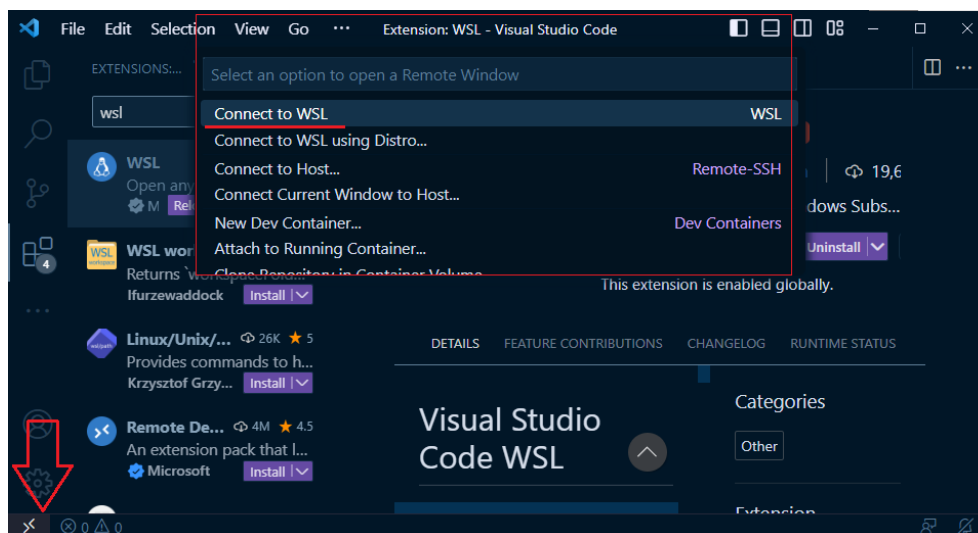
Captura 16: Extensiones de visual

También instalaremos la extensión de WSL2, que nos generará una conexión directa con la distribución de linux que solicitamos.



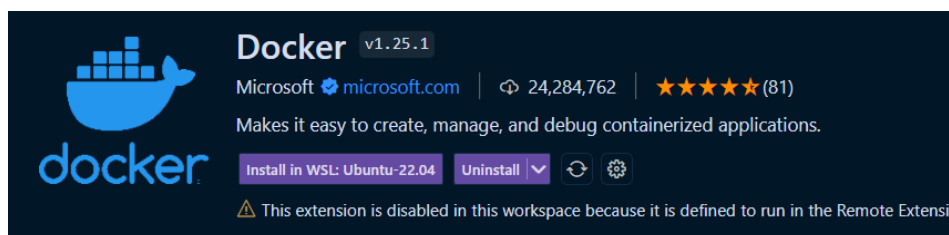
Captura 17: Extensiones de visual

Una vez instaladas, reinicie Visual Studio. En la esquina inferior izquierda, se encuentra la consola de Visual. La abriremos y se desplegará un menú con diversas opciones, le daremos a “Connect to WSL” y automáticamente se reabrirá la venta de Visual Studio con la versión de Ubuntu 22.04.



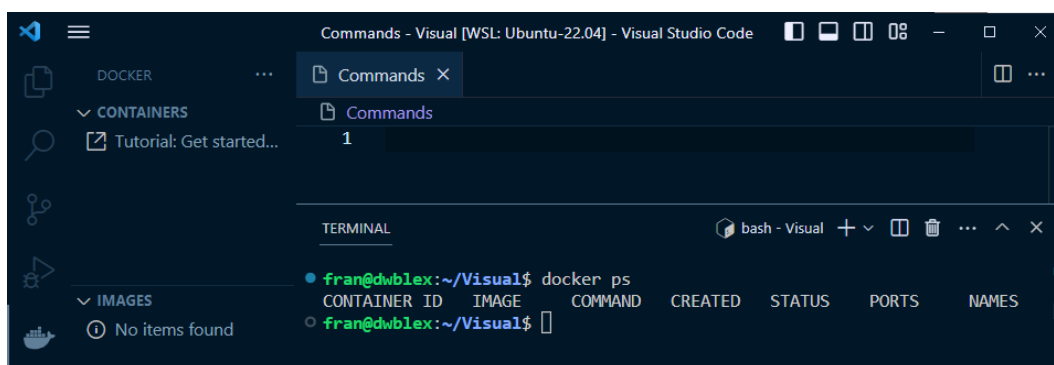
Captura 18: Elegir distribución en Visual

Una vez conectados a la de Ubuntu en WSL2, debemos instalar la extensión de docker para esta distribución.



Captura 19 : Extensiones de Visual en WSL

Visual se reiniciará con la distribución de WSL y nos aparecerá un nuevo icono en el menú izquierdo de la ventana, también podemos acceder de manera simultánea al terminal de Ubuntu 22.04



Captura 20 : GUI de Visual

2.4 Wazuh como HIDS

La instalación de Wazuh implementado en docker se puede realizar en un despliegue de un “solo nodo” o en un “despliegue de varios nodos”. A continuación, explicaré cada uno de estos modos y sus ventajas. Escogeremos la que más se adapte a nuestras necesidades y requisitos. La principal diferencia entre Wazuh en modo single-node y multi-node radica en la escalabilidad y la distribución de la carga de trabajo. A continuación, se presentara las características y el ámbito de cada uno:

2.4.1 Mono-nodo

El modo single-node está diseñado para entornos más pequeños o casos de uso donde no es necesario un alto nivel de escalabilidad. Es adecuado para implementaciones en una única máquina o servidor. En este modo, todos los componentes de Wazuh, como el motor de reglas, el motor de correlación y la base de datos, se ejecutan en un solo nodo. Esto puede limitar la capacidad para manejar grandes volúmenes de datos o una carga de trabajo intensiva. A nivel de hardware, un single-node de Wazuh requeriría al menos una máquina o servidor con suficiente capacidad de procesamiento y memoria para ejecutar todos los componentes necesarios. Los requisitos exactos dependen de la cantidad de eventos y logs que se esperen procesar, pero en términos generales está enfocado a un público que no tiene previsiones de escalabilidad en

2.4.2 Multi-nodo

El modo multi-node está enfocado para entornos más grandes o casos de uso donde se requiere escalabilidad y distribución de la carga de trabajo para implementaciones en entornos en los que se generan grandes volúmenes de datos de seguridad. Los componentes de Wazuh se pueden distribuir en varios nodos, lo que permite aumentar la capacidad de procesamiento y el rendimiento general, por lo que, requeriría varios servidores o máquinas distribuidas en un clúster.

2.4.3 Instalación de Wazuh

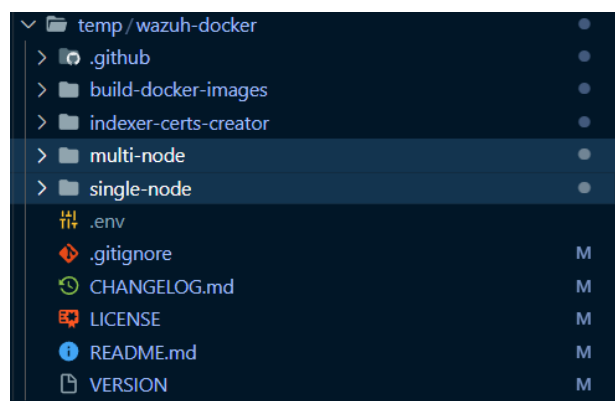
Primero, Clona el repositorio de Wazuh en tu sistema ejecutando el siguiente comando en la terminal:

```
git clone https://github.com/wazuh/wazuh-docker.git -b v4.4.4
```

```
● fran@DESKTOP-REGKV9T:~/Visual/Docker/temp$ git clone https://github.com/wazuh/wazuh-docker.git
-b v4.4.4
Cloning into 'wazuh-docker'...
remote: Enumerating objects: 10512, done.
remote: Counting objects: 100% (552/552), done.
remote: Compressing objects: 100% (289/289), done.
remote: Total 10512 (delta 280), reused 507 (delta 242), pack-reused 9960
Receiving objects: 100% (10512/10512), 313.84 MiB | 15.46 MiB/s, done.
Resolving deltas: 100% (5353/5353), done.
Note: switching to '4e7c2cf72aa29f8717cccf681bb8d64237a05460'.
```

Captura 21: Pull de Wazuh

Una vez descargado, nos posicionamos dentro de la carpeta descargada. Encontraremos dos carpeta relacionadas directamente con el tipo de implementación de Wazuh, multi-node y single-node. Seleccionaremos la que más se adapte a nuestras necesidades:



Captura 22: Directorio de wazuh

Ahora pasaremos a ejecutar los docker-compose, primero debemos configurar certificados para asegurar las comunicaciones entre nodos. Desde dentro de la carpeta del paquete seleccionado ejecutamos el siguiente comando:

```
docker-compose -f generate-indexer-certs.yml run --rm generator
```

```

fran@dwblex:~/Visual/temp/wazuh-docker/multi-node$ docker-compose -f generate-indexer-certs.yml run --rm generator
[+] Building 0.0s (0/0)

[+] Creating 1/0
✓ Network multi-node_default Created

[+] Building 0.0s (0/0)

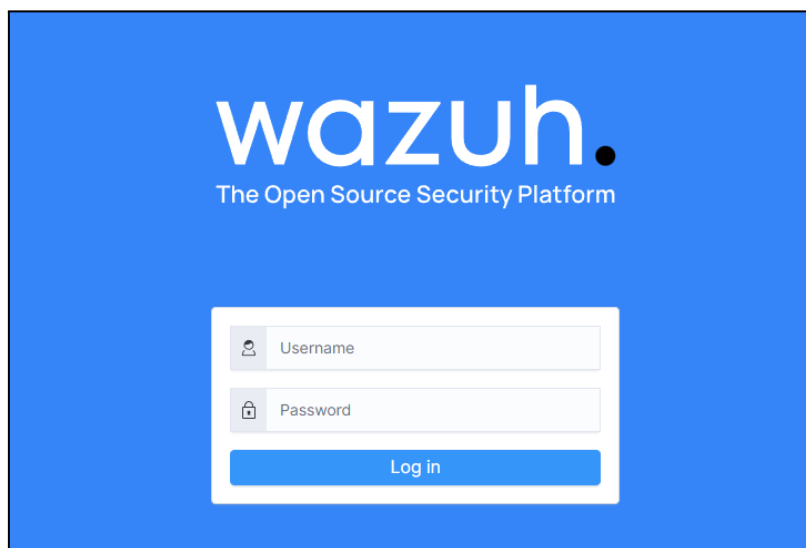
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
             Dload  Upload   Total   Spent    Left   Speed
100 26847  100 26847    0     0  468k      0 --:--:-- --:--:-- --:--:--  468k
Cert tool exists in Packages bucket
14/06/2023 20:40:05 INFO: Admin certificates created.
14/06/2023 20:40:05 INFO: Wazuh indexer certificates created.
14/06/2023 20:40:06 INFO: Wazuh server certificates created.
14/06/2023 20:40:06 INFO: Wazuh dashboard certificates created.

```

Captura 23: Certificados Wazuh

Una vez generado los certificados, podremos levantar el docker-compose con **docker-compose up** (si quieres ver los registros de las acciones que se realizan) o **docker-compose up -d** si solo te interesa saber si el compose funciona y levanta todos los dockers correctamente.

El nombre de usuario y contraseña por defecto para el login de Wazuh son admin y SecretPassword, podremos cambiar la contraseña por defecto para el usuario administrador más adelante.



Captura 24: Login de Wazuh

Una vez dentro de Wazuh, solo quedaría implementar las medidas que nos interesen en nuestro entorno y verificar que funcione correctamente.



3. Seguridad en contenedores docker

La seguridad en Docker es un tema importante a considerar, especialmente porque Docker es un demonio que se ejecuta dentro del sistema operativo y los contenedores dependen del demonio base. Si un contenedor es comprometido, podría tener acceso a los recursos del sistema operativo anfitrión y a otros contenedores en la misma máquina.

Para mitigar estos riesgos, Docker ofrece varias características de seguridad, como la separación de recursos, la limitación de recursos, el aislamiento de red y la gestión de permisos. Estas características permiten que los contenedores se ejecuten de manera segura sin comprometer la seguridad del sistema operativo anfitrión. Sin embargo, como Docker es un demonio que se ejecuta en el sistema operativo, es importante asegurarse de que se estén aplicando las últimas actualizaciones de seguridad y parches de software. Además, se deben aplicar las mejores prácticas de seguridad de Docker, cómo utilizar imágenes de contenedor confiables, limitar los permisos del contenedor, y asegurarse de que los contenedores se estén ejecutando con el menor número de privilegios necesarios.

Otra práctica recomendada es utilizar herramientas de seguridad de terceros que permitan escanear las imágenes de contenedor en busca de vulnerabilidades conocidas y verificar que los contenedores se estén ejecutando en un entorno seguro. Además, se recomienda implementar una política de control de acceso y monitoreo de actividad para detectar cualquier actividad sospechosa en los contenedores.

3.1 Recomendaciones generales en docker

La seguridad en la virtualización en contenedores es un tema muy importante y en constante evolución, ya que los contenedores se utilizan cada vez más para desplegar aplicaciones en entornos de producción.

Las imágenes son la base de los contenedores y es importante asegurarse de que se están utilizando imágenes seguras y actualizadas. Para ello, se recomienda utilizar repositorios de imágenes oficiales y mantenerlas actualizadas con parches de seguridad. También tener en cuenta que los contenedores pueden ejecutar procesos con privilegios de root en el sistema anfitrión. Por lo tanto, es recomendable ejecutar los procesos con el menor nivel de permisos posible y evitar la ejecución de procesos como root. Además de controlar el acceso a los contenedores. Para ello, se pueden utilizar herramientas como los sistemas de autenticación y autorización para controlar quién tiene acceso a los contenedores. Es importante monitorear y registrar los eventos en los contenedores para detectar posibles brechas de seguridad. Para ello, se pueden utilizar herramientas como los



registros de auditoría y los sistemas de detección de intrusos, como veremos posteriormente.

Otro aspecto crítico en la seguridad de los contenedores, es la configuración de red de los contenedores de manera segura, aislarlos del resto de la red y asegurarse de que sólo se están exponiendo los puertos necesarios.

3.2 CIS Docker Benchmark

El CIS Docker Benchmark es un conjunto de directrices de seguridad establecidas por el Center for Internet Security (CIS) para asegurar la implementación y configuración segura de Docker. Estas directrices se basan en las mejores prácticas y recomendaciones de seguridad para ayudar a proteger los entornos Docker y minimizar las vulnerabilidades y riesgos de seguridad.

El CIS Docker Benchmark proporciona una serie de controles de seguridad que abarcan diferentes áreas clave, como la configuración del daemon de Docker, la configuración del host, las imágenes de contenedor y los contenedores en sí. Estas directrices establecen pautas claras sobre cómo configurar adecuadamente Docker para mitigar riesgos de seguridad comunes.

Algunos ejemplos de los controles de seguridad recomendados en el CIS Docker Benchmark incluyen:

1. Configuración del demonio de Docker:

- Uso de TLS para la comunicación segura con el daemon.
- Limitación de los recursos del daemon, como CPU y memoria.
- Configuración de registros de auditoría para el demonio.

2. Configuración del host:

- Implementación de actualizaciones de seguridad regulares en el host.
- Configuración de políticas de firewall para limitar el tráfico de red.
- Restricción de los privilegios del usuario del daemon de Docker.

3. Imágenes de contenedor:

- Utilización de imágenes de contenedor oficialmente firmadas y verificadas.
- Evitación del uso de imágenes con vulnerabilidades conocidas.
- Configuración de políticas de seguridad para escanear imágenes de contenedor en busca de problemas de seguridad.



4. Contenedores:

- Uso de namespaces y cgroups para aislar y limitar los recursos de los contenedores.
- Implementación de políticas de control de acceso y restricciones de capacidad.
- Desactivación de privilegios innecesarios dentro de los contenedores.

El CIS Docker Benchmark proporciona recomendaciones detalladas y puntuales sobre cómo implementar cada control de seguridad. Siguiendo estas directrices, los administradores de Docker pueden fortalecer la seguridad de sus entornos y reducir la superficie de ataque.

3.3 Componentes clave para la creación de contenedores

Vamos a explicar el funcionamiento de los contenedores y las diferencias fundamentales respecto a virtualizar máquinas de manera independiente al sistema operativo operativo.

En primer lugar, la virtualización utiliza un hipervisor que es una capa de software que permite ejecutar varias máquinas virtuales (VM) en una única máquina física. Crea un entorno virtual que emula el hardware subyacente, incluida la CPU, la memoria y el almacenamiento, y aísla cada máquina virtual de las demás. Cada VM puede ejecutar su propio sistema operativo, aplicaciones y datos, y puede gestionarse de forma independiente. Por otro lado, el motor de Docker permite a los desarrolladores crear, desplegar y ejecutar aplicaciones en contenedores ligeros y portátiles que depende del kernel nativo en el que esté instalado docker.

Los contenedores Docker comparten el mismo núcleo que el sistema operativo anfitrión, pero están aislados entre sí mediante espacios de nombres y grupos de control. Cada contenedor incluye todas las dependencias y bibliotecas necesarias para ejecutar la aplicación, lo que facilita su traslado de un entorno a otro. En la siguiente imagen se muestra la comparativa entre la virtualización por hipervisores de tipo 2 con el motor docker

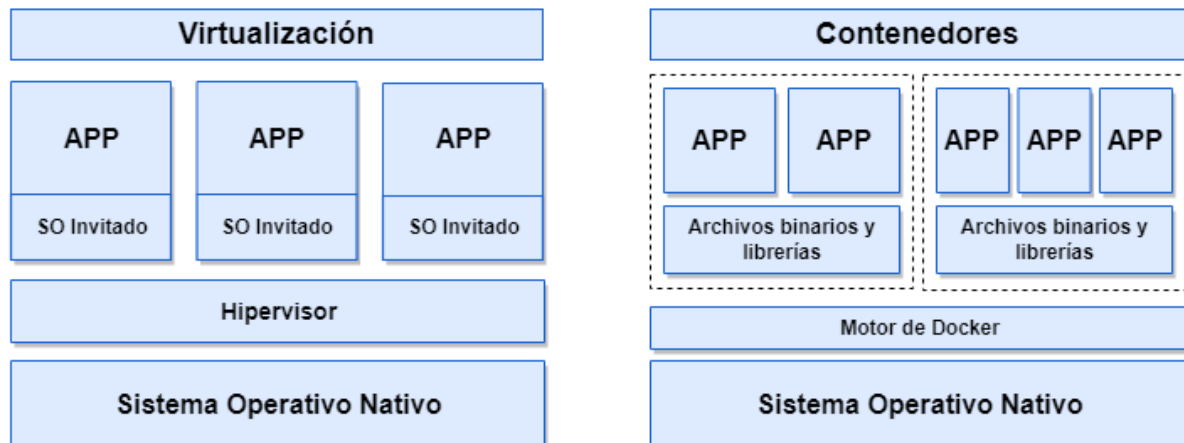


Figura 3: Comparativa virtualización

La principal diferencia entre un hipervisor y un motor Docker es el nivel de abstracción que proporcionan. Un hipervisor crea una máquina virtual que emula el hardware subyacente, mientras que el motor Docker crea un contenedor que comparte el núcleo del host. Esto hace que el motor Docker sea mucho más ligero y rápido que un hipervisor, ya que no hay necesidad de virtualizar el hardware. Otra diferencia es que los hipervisores suelen utilizarse para ejecutar varios sistemas operativos en una única máquina física, mientras que el motor Docker se utiliza principalmente para ejecutar varias instancias de la misma aplicación o servicio, pero usando el kernel existente en la máquina.

3.3.1 Espacio de nombres

Los espacios de nombres permiten que un proceso vea una visión restringida y aislada del sistema, incluyendo la vista del sistema de archivos, los procesos en ejecución, las interfaces de red y los identificadores de proceso. De esta manera, se pueden crear entornos aislados para ejecutar aplicaciones sin interferir con el sistema operativo y otras aplicaciones. Por ejemplo, Docker utiliza espacios de nombres para crear contenedores aislados como mencionamos anteriormente.

Existen varios tipos de espacios de nombres en Linux, incluyendo los espacios de nombres de proceso, red, sistema de archivos, IPC (Inter Process Communication) y usuarios. Cada tipo de espacio de nombres crea un entorno aislado para el proceso en un aspecto específico del sistema, hay 7 espacios de nombre en linux:

| Espacios de nombres | |
|---------------------|--|
| Mount | Puntos de montaje |
| PID | Identificación de procesos |
| Network | Dispositivos de red, pilas, puertos, etc |
| User | Identificación de usuarios y grupos |
| UTS | Nombre de host y nombre de dominio NIS |
| IPC | IPC de System V, colas de mensajes POSIX |
| cgroup | Directorio raíz del grupo de control |

Figura 4: Comparativa virtualización

mnt: El espacio de nombres de montaje en Docker permite particionar virtualmente el sistema de archivos. Cada contenedor de Docker tiene su propio espacio de nombres de montaje, lo que significa que los procesos dentro del contenedor sólo pueden acceder a los archivos dentro de su punto de montaje, proporcionando un nivel adicional de seguridad al limitar el acceso a archivos fuera del contenedor.

Procesos o pid: El espacio de nombres de procesos en Docker garantiza que cada contenedor tenga su propio árbol de procesos aislado. Los contenedores de Docker tienen su propio PID namespace, lo que significa que los procesos dentro del contenedor tienen su propio conjunto de PID, evitando conflictos y permitiendo que los contenedores se ejecuten de manera independiente, con sus propios procesos y PID.

Comunicación entre procesos o ipc: Docker permite controlar la comunicación entre procesos dentro de los contenedores utilizando espacios de nombres IPC (Inter-Process Communication). Esto significa que los procesos dentro de un contenedor solo pueden comunicarse directamente con otros procesos dentro del mismo contenedor, lo que proporciona un aislamiento adicional y evita interferencias no deseadas entre contenedores.

net: El espacio de nombres de red en Docker administra los dispositivos de red visibles para cada contenedor. Docker crea interfaces de red virtuales para cada contenedor y gestiona la conexión entre las interfaces de red del contenedor y las interfaces de red del host. Esto permite que los contenedores tengan su propia configuración de red aislada, lo que facilita la conexión y comunicación con otros contenedores y el host.



Usuario: El espacio de nombres de usuario en Docker proporciona un entorno aislado para los usuarios dentro de los contenedores. Cada contenedor tiene su propio espacio de nombres de usuario, lo que significa que los usuarios dentro del contenedor tienen su propia "raíz virtual" sin acceso de usuario real al sistema principal. Esto mejora la seguridad y permite que los contenedores tengan sus propias configuraciones de usuario sin afectar al host.

UTS: Docker utiliza el espacio de nombres de UTS para cambiar el nombre de host y la información del dominio dentro de un contenedor. Esto permite que los contenedores se ejecuten con nombres de host diferentes, lo que puede ser útil para simular diferentes entornos de host y evitar conflictos en la red.

3.3.2 Grupos de control

Los grupos de control permiten asignar y controlar el uso de recursos del sistema para un conjunto de procesos, lo que permite una gestión eficiente de los recursos en un sistema compartido. Con cgroups, se puede limitar la cantidad de CPU, memoria, E/S y otros recursos que un conjunto de procesos puede usar.

Cada cgroup se puede configurar con una cantidad específica de recursos del sistema, lo que permite una gestión detallada del uso de recursos del sistema. Los cgroups también se pueden anidar para crear una jerarquía de control de recursos.

3.3.3 Control de Acceso Obligatorio

El objetivo principal del control de acceso obligatorio es restringir y controlar de manera precisa el acceso a los recursos del sistema, incluso para los propietarios de los recursos. Esto es particularmente útil en entornos donde la confidencialidad, integridad y disponibilidad de los datos son críticos y de vital importancia, porque deben protegerse de todo tipo de vulnerabilidades. Un ejemplo conocido de implementación de MAC es SELinux (Security-Enhanced Linux), utilizado en sistemas basados en Linux. SELinux proporciona una capa adicional de seguridad en la que se asignan políticas de acceso a procesos y archivos en función de sus contextos de seguridad. Otros sistemas operativos también tienen implementaciones de MAC, como AppArmor en Linux y Integrity Levels en Windows.

Docker en sí mismo no proporciona una funcionalidad nativa de control de acceso obligatorio (MAC). Docker se basa en las capacidades de control de acceso del sistema operativo subyacente en el que se ejecuta. Sin embargo, es posible implementar medidas adicionales de control de acceso obligatorio en Docker utilizando diferentes enfoques y herramientas. Algunas opciones a considerar según lo mencionado previamente son:



SELinux: Si estás utilizando un sistema operativo basado en Linux con soporte para SELinux (como Red Hat Enterprise Linux, CentOS o Fedora), puedes habilitar y configurar SELinux para proporcionar políticas de control de acceso obligatorio para los contenedores Docker. SELinux permite definir políticas granulares para restringir los accesos de los contenedores a otros recursos del sistema.

AppArmor: AppArmor es otro mecanismo de control de acceso obligatorio que se puede utilizar en sistemas Linux, incluidas las distribuciones Ubuntu y SUSE. Permite definir perfiles de seguridad para los contenedores Docker, limitando sus capacidades y restringiendo su acceso a recursos del sistema no autorizados.

Seccomp: Seccomp (Secure Computing Mode) es una característica del kernel de Linux que permite restringir las llamadas al sistema que pueden realizar los contenedores Docker. Al definir perfiles de seccomp, puedes limitar aún más las capacidades de los contenedores, lo que reduce el riesgo de ataques.

3.3.4 Union System File

Los sistemas de archivos de unión son una tecnología utilizada en Docker y otros sistemas de contenedores para proporcionar una capa de abstracción entre el sistema operativo del host y los contenedores. Permiten que los contenedores compartan y reutilicen eficientemente el sistema de archivos del host, al tiempo que les brindan su propia capa de escritura aislada. Comprender las capas de Docker es fundamental para optimizar el tamaño de la imagen, su reutilización y el rendimiento de la construcción. A continuación, explico las capas de un dockerfile:

Imagen Base: La capa de imagen base es la capa inicial de la imagen del contenedor. Contiene el sistema operativo mínimo o una imagen sobre la cual se agregarán las dependencias y el código de tu aplicación. La imagen base generalmente se obtiene de un registro de Docker, como Docker.

Capas de configuración : Cada instrucción en un archivo Dockerfile contribuye a crear una capa intermedia. Ejemplos de instrucciones son RUN, COPY y ADD. Cada instrucción se ejecuta en una capa separada, y el resultado de la instrucción se almacena en esa capa. Esto permite la reutilización de capas si las instrucciones y sus contextos no han cambiado. Las capas intermedias son de solo lectura y no se pueden modificar una vez creadas.

Capa de Escritura: es la capa superior y es donde se guardan los cambios específicos del contenedor, como la escritura de archivos o la modificación del sistema. Cada contenedor basado en una imagen tiene su propia capa de escritura aislada que se agrega encima de las capas anteriores. Esta capa permite que los

contenedores sean inmutables y evita que se modifiquen las capas subyacentes compartidas.

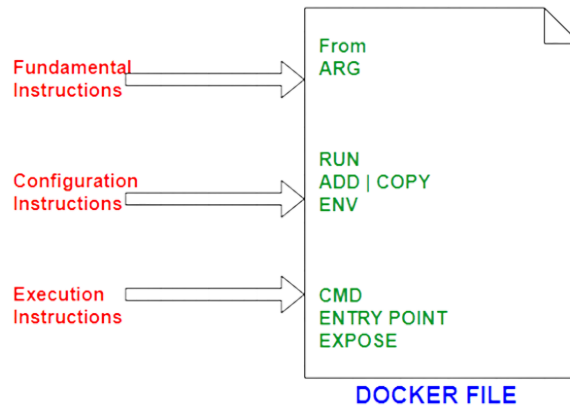


Figura 5: Comparativa virtualización

En Docker, se utiliza un sistema de archivos de unión llamado AUFS (Another Unión File System) como predeterminado en sistemas operativos basados en Linux anteriores a la versión 1.13. A partir de la versión 1.13, Docker utiliza OverlayFS como el sistema de archivos de unión predeterminado en sistemas Linux que admiten OverlayFS.

```

1 FROM php:8.1-fpm
2
3 # Copy composer.lock and composer.json
4 COPY composer.lock composer.json /var/www/
5
6 # Set working directory
7 WORKDIR /var/www
8
9 # Install dependencies
10
11 RUN apt-get update && apt-get install -y \
12     build-essential \
13     libpng-dev \
14     libjpeg62-turbo-dev \
15     libfreetype6-dev \
16     locales \
17     libwebp-dev \
18     zip \
19     jpegoptim optipng pngquant gifsicle \
20     vim \
21     libonig-dev \
22     unzip \
23     git \
24     curl \
25     libzip-dev
26 # Clear cache
27 RUN apt-get clean && rm -rf /var/lib/apt/lists/*
  
```

Captura 25: Ejemplo de dockerfile

Cuando se crea un contenedor en Docker, se crea una capa base de solo lectura que contiene la imagen del contenedor. Luego, Docker agrega una o más capas adicionales en la parte superior de la capa base utilizando un sistema de archivos de unión. Estas capas adicionales contienen todos los cambios realizados dentro del contenedor, como la instalación de paquetes, la modificación de archivos y la creación de nuevos archivos.



El uso de un sistema de archivos de unión proporciona varias ventajas en Docker:

1. **Eficiencia de almacenamiento:** debido a que las capas son de solo lectura y se reutilizan entre contenedores, se evita la duplicación de datos y se ahorra espacio en el disco, lo que supone un ahorro en costes de almacenamiento para empresas con bajos recursos.
2. **Rendimiento:** los sistemas de archivos de unión permiten un acceso rápido a los archivos, ya que solo se leen las capas superiores que contienen los cambios realizados.
3. **Aislamiento:** cada contenedor tiene su propia capa de escritura aislada, lo que significa que los cambios realizados dentro de un contenedor no afectan a otros contenedores ni al sistema operativo del host.
4. **Capacidad de revertir cambios:** dado que las capas son independientes, es posible eliminar o revertir rápidamente los cambios realizados en un contenedor, simplemente eliminando las capas adicionales.

3.5 Alternativas a docker

Existen una variedad de entornos de contenedores que compiten directamente con docker o se aplican en ámbitos distintos dependiendo del sector mercado ya que ofrecen funcionalidades adicionales o se centran en aspectos específicos.

3.5.1 Plataformas alternativas

Kubernetes: Kubernetes es una plataforma de orquestación de contenedores de código abierto que permite gestionar y desplegar aplicaciones en contenedores a gran escala. A diferencia de Docker, que se centra en el empaquetado de aplicaciones en contenedores, Kubernetes proporciona capacidades de escalabilidad, administración de recursos, alta disponibilidad y tolerancia a fallos.

rocket(rktr): Es otro motor de contenedores de código abierto que se enfoca en la seguridad y la interoperabilidad. Ofrece un enfoque modular y minimalista, con énfasis en la separación de procesos y la capacidad de ejecutar contenedores en entornos aislados.

LXC (Linux Containers): LXC es una tecnología de virtualización a nivel de sistema operativo similar a Docker. Proporciona entornos ligeros y aislados llamados contenedores, pero a diferencia de Docker, LXC utiliza máquinas virtuales basadas en Linux para crear estos contenedores.



OpenShift: OpenShift es una plataforma de desarrollo y despliegue de aplicaciones de nivel empresarial basada en Kubernetes. Ofrece una solución completa para el ciclo de vida de las aplicaciones en contenedores, incluyendo la construcción, implementación y gestión de contenedores, así como herramientas de monitoreo y escalabilidad.

Podman: Podman es una herramienta de administración de contenedores compatible con la interfaz de línea de comandos de Docker, pero sin necesidad de un demonio en ejecución. Se centra en la seguridad y el aislamiento, permitiendo ejecutar y gestionar contenedores sin privilegios de root.

3.5.2 Computación de alto rendimiento, Singularity

La computación de alto rendimiento (HPC) se refiere a la utilización de sistemas informáticos avanzados y técnicas especializadas para resolver problemas computacionales complejos y realizar cálculos intensivos en términos de recursos computacionales. Singularity es una herramienta diseñada específicamente para entornos de computación de alto rendimiento.

A diferencia de Docker, Singularity permite ejecutar aplicaciones dentro de contenedores que están optimizados para entornos HPC y que aprovechan las características específicas del sistema operativo del host sin los problemas de compatibilidad y rendimiento que a veces se encuentran al usar Docker. Es usado en muchas disciplinas científicas, como la astronomía, la genómica, la física de partículas y la bioinformática, se realizan cálculos intensivos y se requiere acceso a grandes clústeres de computadoras para analizar datos y ejecutar modelos computacionales complejos. Ambas tecnologías están relacionadas con la virtualización y la gestión de entornos de ejecución de aplicaciones. Aunque tienen objetivos similares, existen diferencias clave entre ellos, lo que diferencia Singularity de docker a primera vista es la carencia de un demonio (Container daemon, CD en la siguiente figura) encargado de mantener el servicio, por lo que es una mejora en términos de seguridad respecto a docker.

- El tiempo de arranque del contenedor es un poco más rápido que en Docker, ya que elimina gran parte de la configuración del espacio de nombres.
- En el sistema respaldado por Lustre (un sistema de archivos), donde la búsqueda de metadatos abarca un servidor diferente de la búsqueda de bloques de datos, la imagen de contenedor de archivo único mejora significativamente el rendimiento al reducir la multitud de búsquedas de metadatos con el MDS.

- Singularity tiene soporte integrado para MPI (OpenMPI, MPICH, IntelMPI).

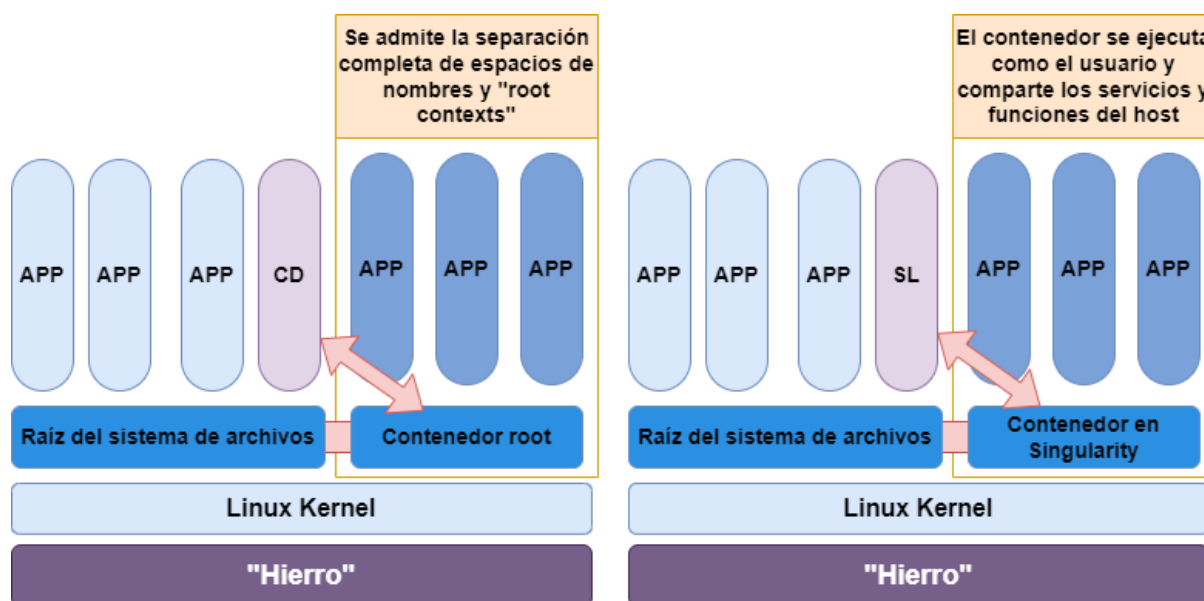


Figura 6: Comparativa Docker con Singularity

Una vez que la aplicación se construye con las bibliotecas MPI, cuando ejecutas un comando `mpirun` (equivalente al comando `docker`) de forma normal, sustituyendo el binario habitual con la aplicación singularity de un solo archivo. Mientras la aplicación se ejecuta dentro del contenedor singularity, las llamadas a la interfaz de gestión de procesos (PMI/PMIx) pasarán a través del lanzador singularity (SL en la figura anterior) a ORTED.

3.6 Sistemas de detección de intrusos (HIDS)

Los sistemas de detección de intrusos basados en host (HIDS, por sus siglas en inglés) se utilizan para monitorear y detectar actividades sospechosas en un sistema operativo y aplicaciones específicas. Aunque Docker ofrece aislamiento y seguridad a nivel de contenedor, es posible utilizar HIDS dentro de contenedores Docker para mejorar aún más la detección de intrusos.

3.6.1 Wazuh

Wazuh es una plataforma de seguridad de código abierto que incluye HIDS y monitoreo de logs. Cuenta con una amplia gama de reglas predefinidas y escenarios de detección para Docker. Proporciona alertas en tiempo real sobre actividades sospechosas y análisis de seguridad. También ofrece capacidades de correlación de eventos y un panel de control centralizado. Es altamente escalable y compatible con una variedad de sistemas operativos. Ofrece una interfaz web intuitiva para la visualización de eventos y una sólida comunidad de soporte.



Algunos usuarios pueden encontrar que la configuración y personalización avanzada pueden ser un poco complejas. La documentación puede ser extensa y requerir tiempo para comprender completamente.

3.6.2 Sysdig Falco

Falco es un HIDS diseñado específicamente para contenedores y orquestadores, como Docker y Kubernetes. Utiliza reglas y perfiles de comportamiento para detectar actividades sospechosas en tiempo real. Proporciona alertas detalladas y también permite la integración con otras herramientas de seguridad. Es muy liviano y está diseñado para funcionar bien en entornos de contenedores. Tiene una amplia comunidad de usuarios y desarrolladores que brindan soporte y contribuyen con nuevas reglas. Además, es fácil de configurar y ofrece una gran cantidad de eventos predefinidos para la detección de amenazas. También puede generar una gran cantidad de alertas, lo que puede requerir un análisis cuidadoso para distinguir las amenazas reales de los falsos positivos, pero no tiene capacidades de respuesta automática.

3.3.3 Docker Bench para seguridad

Docker Bench para seguridad es una herramienta de código abierto que escanea imágenes y contenedores de Docker en busca de configuraciones de seguridad comunes que puede identificar vulnerabilidades conocidas y proporcionar recomendaciones de seguridad. Es fácil de usar y proporciona una evaluación rápida de la seguridad de los contenedores y las imágenes de Docker. Proporciona pautas y recomendaciones para mejorar la seguridad. Enfocado principalmente en la configuración de seguridad de Docker y no ofrece detección en tiempo real de actividades anómalas.

3.3.4 OSSEC

OSSEC es otro HIDS de código abierto que proporciona monitoreo de host y detección de intrusos. Es compatible con Docker y puede monitorear los contenedores en tiempo real. Proporciona alertas y notificaciones en caso de actividad sospechosa, y también ofrece análisis de integridad de archivos y registro. Es altamente configurable y ofrece una amplia gama de reglas predefinidas para la detección de amenazas. Es compatible con múltiples plataformas y es escalable para entornos grandes, también tiene una comunidad activa que brinda soporte y actualizaciones regulares.

Al igual que Wazuh, la configuración y personalización avanzada pueden ser complicadas para algunos usuarios. La interfaz de usuario puede parecer un poco anticuada y requiere conocimientos técnicos para aprovechar todas las características.



3.3.5 Anchore Engine

Permite escanear las imágenes de Docker en busca de vulnerabilidades conocidas y configuraciones incorrectas, además de proporcionar políticas de seguridad personalizables y generar informes detallados sobre los riesgos encontrados. Es altamente automatizado y proporciona una evaluación exhaustiva de las imágenes de contenedores antes de su implementación. Ofrece una amplia cobertura de vulnerabilidades conocidas y configuraciones inseguras. Además, es compatible con integraciones de CI/CD y herramientas de orquestación de contenedores. Principalmente se enfoca en la evaluación de imágenes y no en la detección en tiempo real de actividades sospechosas en los contenedores en ejecución. Su enfoque está más orientado hacia la prevención y el análisis estático en lugar de la detección dinámica de intrusiones.

4. Funcionalidades de los agentes de Wazuh

En Wazuh, un agente es un componente de software que se instala en los sistemas que se desean monitorizar. Los agentes de Wazuh recopilan información sobre eventos, logs y actividad del sistema, y la envían al servidor de Wazuh para su análisis y procesamiento.

Cada agente se ejecuta en un sistema operativo específico y está diseñado para recopilar información relevante de ese sistema, como logs del sistema, logs de aplicaciones, eventos de seguridad, registros de cambios de archivo, entre otros. Los agentes son responsables de recolectar, normalizar y enviar esta información al servidor central de Wazuh.

4.1 Respuesta activa

Wazuh tiene un módulo de respuesta activa que ayuda a los equipos de seguridad a automatizar las acciones de respuesta basadas en desencadenantes específicos, lo que les permite gestionar eficazmente los incidentes de seguridad.

Los equipos de seguridad se encuentran a menudo con problemas en la respuesta a incidentes, como abordar a tiempo los sucesos de alta gravedad o proporcionar acciones de mitigación completas. Pueden tener dificultades para recopilar información relevante en tiempo real, lo que dificulta la comprensión del alcance total de un incidente. Estos problemas aumentan la dificultad para contener y mitigar el impacto de un ciberataque.

La plataforma SIEM y XDR de Wazuh mejora la respuesta a incidentes al:

- Proporcionando visibilidad en tiempo real de los eventos de seguridad.



- Reduciendo la fatiga por alertas.
- Automatizando las acciones de respuesta a las amenazas.
- Proporcionando scripts de respuesta out-of-the-box.

La automatización de las acciones de respuesta garantiza que los incidentes de alta prioridad se abordan y solucionan de forma oportuna y coherente. Esto es especialmente valioso en entornos en los que los equipos de seguridad tienen recursos limitados y necesitan priorizar sus esfuerzos de respuesta.

Además, el módulo incluye una serie de secuencias de comandos de respuesta listas para usar que ayudan a responder a las amenazas y mitigarlas. Por ejemplo, algunas secuencias de comandos bloquean el acceso malintencionado a la red y eliminan los archivos maliciosos en los puntos finales supervisados. Estas acciones reducen la carga de trabajo de los equipos de seguridad y les permiten gestionar eficazmente los incidentes.

El módulo de respuesta activa de Wazuh ejecuta estas secuencias de comandos en los terminales supervisados cuando se activa una alerta de una regla, nivel o grupo de reglas específico. Puede establecer cualquier número de scripts para iniciar en respuesta a un disparador; sin embargo, debe considerar estas respuestas cuidadosamente. Una mala implementación de reglas y respuestas podría aumentar la vulnerabilidad de un punto final.

Una respuesta activa puede ser:

- Respuestas activas sin estado: son acciones únicas sin una definición de evento para revertirlas o detenerlas.
- Respuestas con estado: revierten o detienen sus acciones tras un periodo de tiempo.

4.2 Monitorización sin agente

El servidor Wazuh analiza los datos que recibe de los agentes Wazuh para monitorear, detectar y activar alertas de eventos e incidentes de seguridad en los endpoints. Sin embargo, algunos puntos finales pueden tener limitaciones que impiden la instalación del agente Wazuh. Wazuh resuelve este problema utilizando la capacidad de monitorización sin agente.

La monitorización sin agente se refiere a un tipo de monitorización de puntos finales que no requiere la instalación de un agente o software. Este enfoque utiliza



los protocolos existentes para acceder y recopilar información del punto final supervisado.

La capacidad de monitoreo sin agente de Wazuh usa el protocolo SSH (Secure Shell) para recolectar y transferir eventos desde los puntos finales al servidor Wazuh. Las plataformas soportadas incluyen routers, firewalls, switches y sistemas Linux/BSD. Permite que los puntos finales con restricciones de instalación de software cumplan los requisitos de seguridad y conformidad.

4.3 Detección de malware

La detección de malware se refiere al proceso de analizar un sistema informático o una red para detectar la existencia de software y archivos maliciosos. Los productos de seguridad pueden identificar programas maliciosos buscando firmas de programas maliciosos conocidos. Las herramientas de seguridad también pueden detectar actividades maliciosas detectando comportamientos sospechosos en la actividad del software. Cuando el malware infecta un sistema, puede modificarlo utilizando diversas técnicas para eludir la detección. Wazuh utiliza un enfoque de amplio espectro para contrarrestar esas técnicas con el fin de detectar archivos maliciosos y patrones anormales que indican la presencia de malware.

El módulo de monitorización de la integridad de los archivos (FIM) de Wazuh ayuda a detectar archivos maliciosos en los puntos finales monitorizados. Por sí solo, el módulo FIM no puede detectar archivos maliciosos. Sin embargo, puede detectar malware combinando el módulo FIM con reglas de detección de amenazas y fuentes de inteligencia de amenazas. Puede configurar Wazuh para usar eventos FIM con fuentes de inteligencia de amenazas como VirusTotal y listas CDB que contengan hashes de archivos, y escaneos YARA para detectar malware.

Wazuh detecta el comportamiento rootkit en los puntos finales monitorizados usando el módulo Rootcheck. Rootcheck supervisa continuamente los puntos finales y genera alertas cuando detecta alguna anomalía. La supervisión de anomalías garantiza que Wazuh detecte malware que las técnicas basadas en firmas podrían haber pasado por alto. Rootcheck también utiliza firmas conocidas de rootkits y troyanos para detectar su presencia en los puntos finales supervisados. La flexibilidad de Wazuh garantiza que los usuarios puedan actualizar ellos mismos estas firmas de rootkits.

La capacidad de recopilación de registros de Wazuh le permite recopilar registros de software de detección de malware de terceros. Usando esta capacidad, Wazuh recoge y analiza los registros de varios software de detección de malware como Windows Defender y ClamAV.



4.4 Monitorización de comandos

Hay ocasiones en las que puede querer monitorizar cosas que no están en los registros. Para solucionar esto, Wazuh incorpora la capacidad de monitorizar la salida de comandos específicos y tratar la salida como si fuera contenido del archivo de registro.

4.5 Supervisión de la integridad de los archivos

La supervisión de la integridad de los archivos (FIM) es un proceso de seguridad utilizado para supervisar la integridad de los archivos del sistema y de las aplicaciones. FIM es una capa de defensa de seguridad importante para cualquier organización que supervise activos sensibles. Proporciona protección para datos sensibles, aplicaciones y archivos de dispositivos mediante la supervisión, el escaneo rutinario y la verificación de su integridad. Ayuda a las organizaciones a detectar cambios en los archivos críticos de sus sistemas, lo que reduce el riesgo de que los datos sean robados o se vean comprometidos. Este proceso puede ahorrar tiempo y dinero en pérdida de productividad, pérdida de ingresos, daños a la reputación y sanciones legales y de cumplimiento normativo.

Wazuh tiene una capacidad integrada para la supervisión de la integridad de los archivos. El módulo Wazuh FIM supervisa archivos y directorios y activa una alerta cuando un usuario o proceso crea, modifica y elimina archivos supervisados. Ejecuta un escaneo de línea base, almacenando la suma de control criptográfica y otros atributos de los archivos monitoreados. Cuando un usuario o proceso modifica un archivo, el módulo compara su suma de comprobación y atributos con la línea de base. Si detecta una discrepancia, activa una alerta. El módulo FIM realiza escaneos en tiempo real y programados dependiendo de la configuración FIM para agentes y manager.

Algunas de las ventajas de la capacidad FIM de Wazuh son:

- Gestión de cambios
- Detección de amenazas y respuesta
- Cumplimiento normativo

La capacidad FIM ayuda a las organizaciones a cumplir los requisitos normativos de seguridad, privacidad y retención de datos. La supervisión de los archivos críticos para detectar cambios es un requisito importante para normativas como PCI DSS, HIPAA y GDPR.



4.6 Recopilación de datos de registro

La recopilación de datos de registro es el proceso en tiempo real de dar sentido a los registros generados por servidores o dispositivos. Este componente puede recibir registros a través de archivos de texto o registros de eventos de Windows. También puede recibir directamente registros a través de syslog remoto, lo que resulta útil para cortafuegos y otros dispositivos de este tipo.

El propósito de este proceso es la identificación de errores de aplicación o del sistema, malas configuraciones, intentos de intrusión, violaciones de políticas o problemas de seguridad.

Los requisitos de memoria y CPU del agente Wazuh son insignificantes, ya que su función principal es enviar eventos al gestor. Sin embargo, en el gestor Wazuh, el consumo de CPU y memoria puede aumentar rápidamente dependiendo de los eventos por segundo (EPS) que el gestor tenga que analizar.

4.7 Supervisión de políticas de seguridad

La monitorización de políticas es el proceso de verificar que todos los sistemas se ajustan a un conjunto de reglas predefinidas relativas a los ajustes de configuración y al uso aprobado de las aplicaciones.

Wazuh utiliza tres componentes para realizar esta tarea: Rootcheck, OpenSCAP y CIS-CAT.

4.8 Monitorización de llamadas al sistema

La supervisión de las llamadas al sistema en los puntos finales de Linux proporciona información con fines de auditoría de seguridad. Recopilar y analizar datos de llamadas al sistema ayuda a los equipos de seguridad a identificar patrones de comportamiento sospechoso e investigar a tiempo posibles incidentes de seguridad.

El sistema de auditoría de Linux es una potente herramienta para recopilar eventos de seguridad y no relacionados con la seguridad en los puntos finales de Linux. Sin embargo, el gran volumen de datos generados por los registros de auditoría puede dificultar a los administradores de sistemas la identificación de posibles amenazas y violaciones de la seguridad.

Wazuh utiliza el sistema de auditoría de Linux para monitorizar las llamadas al sistema en los terminales Linux. El agente Wazuh instala y configura reglas de auditoría en los puntos finales monitorizados para recoger eventos de llamadas al sistema y los envía al servidor Wazuh para su análisis. Estas reglas de auditoría capturan eventos relevantes para la supervisión de la seguridad. Wazuh proporciona reglas de detección listas para usar que utilizan los eventos de llamada al sistema para detectar múltiples actividades, incluyendo acceso a



archivos, ejecución de comandos, escalada de privilegios, malware y más. Los equipos de seguridad pueden personalizar estas reglas para cumplir requisitos de seguridad específicos o normas de conformidad, obteniendo así información en tiempo real sobre posibles incidentes de seguridad.

Al proporcionar una visión centralizada de los eventos de auditoría, Wazuh simplifica la tarea de supervisar las actividades del sistema y ayuda a las organizaciones a cumplir con los requisitos reglamentarios. En general, la capacidad de auditoría de Wazuh proporciona una solución de monitorización de seguridad robusta y completa para los sistemas Linux, ayudando a las organizaciones a mejorar su postura de seguridad y protegerse contra las amenazas cibernéticas.

4.9 Integración con VirusTotal

Wazuh detecta archivos maliciosos a través de una integración con VirusTotal, una potente plataforma que agrega múltiples productos antivirus y un motor de escaneo en línea. La combinación de esta herramienta con nuestro módulo FIM proporciona una forma eficaz de inspeccionar los archivos supervisados en busca de contenido malicioso. Hay dos formas de implementar eso según la API que se use de VirusTotal:

- **API pública**

Este método utiliza una API gratuita con muchas de las funcionalidades de VirusTotal. Sin embargo, tiene algunas limitaciones importantes, tales como:

- Limitaciones en la tasa de peticiones, que puedes consultar en la web de VirusTotal.
- Acceso de baja prioridad para las peticiones realizadas por esta API al motor de VirusTotal.

- **API privada**

VirusTotal también proporciona una API Privada premium en la que las tasas de petición permitidas sólo están limitadas por los Términos de Servicio del usuario. Aparte de eso, proporciona acceso de alta prioridad para las peticiones, junto con ventajas adicionales.

5. Marco de referencia

Seguridad y alta disponibilidad: recomendaciones y tecnologías más usadas para la securización de docker y la monitorización de los contenedores.



6. Conclusión

La implementación de un entorno de pruebas basado en Visual Studio, WSL y docker para el desarrollo y virtualización de software puede representar una mejora en el entendimiento y manejo de las tecnologías referentes a contenedores, ahora bien, el uso a nivel profesional en un entorno laboral depende de la comodidad del usuario con este IDE. Un punto importante a tener en cuenta es la posibilidad de errores al ser un entorno muy limitado en algunos aspectos.

Ahora bien, securizar docker no es una tarea fácil e implementar Wazuh para el monitoreo y análisis de log en los contenedores o del propio servidor de docker es una tarea que puede llevar algo de tiempo implementar dada la complejidad de este.

7. Referencias

<https://forums.docker.com/t/is-there-a-pictorial-diagram-of-how-wsl-2-docker-docker-desktop-are-related/100071>

<https://ubuntu.com/tutorials/windows-and-ubuntu-interopability-on-wsl2#1-overview>

<https://learn.microsoft.com/es-es/windows/wsl/wsl-config>

<https://www.ionos.es/digitalguide/servidores/know-how/podman-vs-docker/>

<https://docs.sylabs.io/guides/3.5/user-guide/introduction.html>

<https://researchcomputing.princeton.edu/support/knowledge-base/singularity>

<https://documentation.wazuh.com/current/deployment-options/docker/wazuh-container.html>

<https://documentation.wazuh.com/current/container-security/docker-monitor/index.html>