Seguridad de Contenedores Docker mediante Procesos de Hardening

Danitza Helen Vallejos Quiñones
Postgrado en Informática
Universidad Mayor de San Andrés
La Paz - Bolivia
helen.vallejos.q@gmail.com

***Resumen*—Docker es un proyecto que permite la virtualización de aplicaciones en contenedores, los cuales se caracterizan por ser flexibles y ligeros a un coste mínimo para las organizaciones, a diferencia de una máquina virtual y un contenedor radica en que la última es más ligera, además de ejecutar las aplicaciones y servicios de manera independiente. Este artículo explica las posibles vulnerabilidades, asociadas al uso de contenedores, además de proporcionar recomendaciones de seguridad, en cuanto a planificación e implementación aplicables a la tecnología de contenedores Docker, así también proporcionar, mejores prácticas para proteger su entorno durante las fases de construcción e implementación para preservar los contenedores durante el tiempo de ejecución, además de mejorar su seguridad a partir de un proceso de hardening. El diseño de seguridad de los contenedores, tomara como base los dominios de COBIT, gestión de seguridad, de cambios, de operaciones, supervisión y riesgos.**

Palabras clave—contenedor, hardening, host, imágenes, kernel

I. INTRODUCCIÓN

En los últimos años la tecnología de contenedores, ha ganado popularidad en modelos empresariales a nivel internacional, ahorrando tiempo y equipamiento a las organizaciones.

Los contenedores son un paquete de elementos que permite crear un entorno donde corren aplicaciones independientes del sistema operativo. El código que corre en estos contenedores se encuentra aislada de otros contenedores, y todos ellos comparten el mismo kernel del sistema operativo. El objetivo de un contenedor va más enfocado a la ejecución de aplicaciones y servicios, de manera aislada e independiente.

La diferencia entre una máquina virtual y un contenedor, radica en que el ultimo es mucho más ligero, porque solo necesita un sistema operativo que corre en la maquina donde se aloja el contenedor, mientras que las máquinas virtuales requieren de un sistema operativo, asignación de disco, CPU y memoria RAM. Fig. 1.

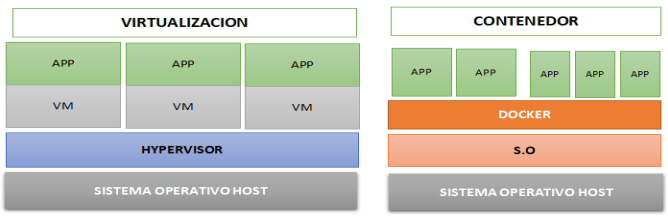

Fig. 1. Virtualización vs Contenedor
Fuente: Cuervo Víctor, Contenedores vs Máquinas virtuales

La seguridad de la infraestructura en los contenedores, se ha convertido en un paso indispensable para fortalecer esta tecnología, contra cualquier tipo de ataque. "En el año 2017, Sagie Dulce durante la conferencia de Black hat, realizo una ponencia llamada "Como abusar de la API Docker con contenedores fantasmas", donde con una prueba de concepto (PoC), demostró como en los montajes de Docker sobre Windows era posible usar la API a través de las conexiones de TCP de un contenedor para crear o probar aplicaciones en los contenedores en ejecución [1].

Esta vulnerabilidad fue subsanada por Docker, en sus últimas actualizaciones, pero a partir de la investigación realizada por Sagie Dulce, se han realizado varias pruebas de concepto.

Si bien este tipo de tecnología ha revolucionado, también presenta nuevos desafíos de seguridad, lo que ha implicado que los profesionales en seguridad incorporen prácticas de cómo realizar un *hardering* (endurecer) los contenedores y host, como primer paso para un entorno seguro.

A. *Contenedor Docker.*

Docker es un motor de contenedores que usa las características del Kernel de Linux, para crear contenedores encima del sistema operativo y automatizar el despliegue de aplicaciones. Es decir que empaqueta software en unidades estandarizadas llamadas contenedores, las cuales incluyen todo lo necesario para que el software se ejecute.

Arquitectura de un Contenedor: La arquitectura de un contenedor Docker está compuesta por los siguientes niveles:

- Imágenes de Docker, que son plantillas de solo lectura.
- Registros Docker, estos registros guardan las imágenes, son repositorios públicos o privados donde se puede subir o bajar las imágenes.
- Contenedores Docker, El contenedor aloja todo lo necesario para ejecutar una aplicación, cada uno es creador de una imagen Docker y es una plataforma aislada [2].

Para referenciar este artículo (IEEE):
[N] D. Vallejos, «Seguridad de Contenedores Docker mediante Procesos de Hardening», *Revista PGI. Investigación, Ciencia y Tecnología en Informática*, nº 7, pp. 14-17, 2020.

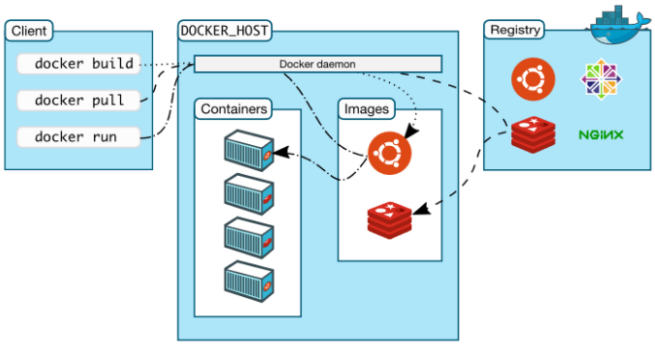

Fig. 2. Arquitectura de un contenedor
Fuente: Ginés, X. F. Seguridad en Docker

B. Hardening.

Hardening en seguridad, es el proceso de asegurar un sistema mediante la reducción de vulnerabilidades, esto se logra eliminando software, usuarios, servicios, logs, etc., que no son necesarios en el sistema, así también cerrando puertos que son innecesarios para la aplicación o sistema. Su propósito es, dificultar la labor del atacante y ganar tiempo para poder minimizar las consecuencias de un incidente de seguridad, o evitar que se concrete el ataque en su totalidad [3].

Contenedores desde la perspectiva de seguridad: Docker es un contenedor alrededor de grupos de control de Linux denominados (cgroups), los cgroups se utilizan en el Kernel de Linux para monitorear y restringir recursos entre un grupo de procesos, los espacios de nombres determinan lo que un proceso puede ver. Por ejemplo, el espacio de nombres PID restringe que procesos que se pueden ver dentro de un contenedor.

Cada contenedor que se ejecuta en un host comparte un Kernel subyacente común, los contenedores están aislados unos de otros, lo que, desde el punto de vista de seguridad, es una ventaja. Sin embargo, si el sistema operativo host está comprometido, todos los contenedores que se ejecutan en él, está en riesgo. Del mismo modo, si un contenedor utiliza una biblioteca vulnerable, podría ser una vulnerabilidad para obtener acceso al host subyacente [4].

C. Seguridad a nivel host.

El sistema operativo host, debe estar protegido para evitar que las vulnerabilidades de los contenedores afecten al mismo. Para abordar este desafió el sistema operativo Linux, proporciona varios modelos de seguridad listos para usar, entre los más populares se encuentran:

- SElinux
- AppArmor

También se pueden utilizar módulos de seguridad personalizados, utilizando (LSM) [5].

a) SElinux: Es un módulo de seguridad, de control de acceso obligatorio (MAC), se basa en un "etiquetado" de todos elementos del sistema y sobre las cuales se aplicarán las políticas de control de acceso configuradas, de tal modo que cualquier operación realizada será comprobada por las etiquetas, incluso cuando se hayan cumplido otros controles de seguridad, para determinar si la operación está permitida o no. Este tipo de configuración permite que cualquier proceso de Linux ejecute operaciones de archivo (como, abrir, leer, buscar).

b) AppArmor: También es un control de acceso obligatorio (MAC), sin embargo, esta se basa en rutas del sistema de archivos en lugar de definir tipos, en este caso se puede especificar una ruta de archivo a un binario y a los permisos que tiene. Este tipo de configuración bloquea la lectura de directorios no seguros, se puede personalizar para restringir el acceso a las aplicaciones. Sin embargo, requiere de una curva de aprendizaje para escribir buenos perfiles de aplicación.

D. Seguridad a nivel red.

Para abordar la seguridad de los contenedores a nivel red, se debe considerar las siguientes opciones:

a) Restricción de protocolos: Únicamente se deben abrir los protocolos que son necesarios para un servicio en cuestión, se debe habilitar el protocolo SSH en un contenedor, únicamente para la maquina host, dado que, desde esta, se controla todos los contenedores. Si se ejecuta un contenedor con el servicio SSH habilitado, la complejidad de mantener un entorno seguro en el contenedor se eleva, ya que dificultan las políticas de gestión de acceso, las calves y las contraseñas.

Si el servicio se encuentra ejecutándose dentro del contenedor desplegado, la solución es desinstalar el servicio desde el propio contenedor en ejecución, caso contrario se debería gestionar desde el fichero Dockerfile.

b) Restricción de puertos: Los puertos denominados "Well-known-ports", o puertos conocidos, son considerados como puertos privilegiados, esto significa que los usuarios y procesos que no sean root o tengan permisos root, no podrán atarse a ninguno de estos puertos. Sin embargo, Docker permite que un contenedor pueda mapear un puerto del contenedor a un puerto privilegiado, ya que el propio demonio de Docker requiere privilegios de root para ejecutarse, lo que significa que, si no se especifica un puerto a mapear entre contenedor y host, Docker automáticamente realizara un mapeo de puertos disponibles, dentro del rango 491253 - 65536 de la maquina host. Una buena práctica de seguridad, para este caso consistiría, en evitar que Docker realice un mapeo de los puertos privilegiados dentro de la maquina host [6].

c) Restricción de tráfico: Los contenedores deben configurarse para separar el tráfico, en redes virtuales discretas por nivel de sensibilidad. Si bien la segmentación por aplicación, también es posible, en la mayoría de los casos la mitigación por sensibilidad es suficiente para una organización, por ejemplo, las aplicaciones publicas pueden compartir una red virtual, mientras que las aplicaciones internas puedes usar otra, y la conexión entre ambas debe producirse a través de un número pequeño de interfaces bien definidas.

El control de tráfico de entrada y salida enviado por los contenedores, deben establecer las fronteras de red, garantizando que los contenedores no pueden enviar tráfico a través de redes de diferentes niveles.

Debido a que los contenedores implementados en varios hosts normalmente se comunican a través de una red virtual cifrada, los dispositivos de red tradicionales a menudo son "ciegos" a ese tipo de tráfico, además que a los contenedores se les asignan direcciones IP dinámicas automáticamente cuando los contenedores se implementan, y estas direcciones cambian continuamente.

Por lo tanto, una combinación de dispositivos de nivel de red existentes, y un filtrado de red compatibles con la aplicación deberían poder, no solo ver el tráfico de los contenedores, sino también poder generar dinámicamente las reglas utilizadas, para filtrar ese tráfico en función a las características de las aplicaciones que se ejecutan en los contenedores, como en [5].

E. Vulnerabilidades en contenedores.

Para un mejor estudio de las vulnerabilidades en los contenedores se clasificará de acuerdo a su arquitectura.

a) Vulnerabilidades en las imágenes: Debido a que las imágenes incluyen todos los componentes utilizados para ejecutar una aplicación determinada, es posible que sus componentes carezcan de actualizaciones de seguridad, o que se encuentren desactualizadas. Una imagen creada con los componentes actualizados, tiene menos posibilidades de tener vulnerabilidades conocidas, durante las primeras semanas.

A diferencia de los sistemas operativos tradicionales, donde el software se actualiza de manera automática, en los hosts donde se ejecutan los contenedores, estas actualizaciones deben realizarse en las propias imágenes, que posteriormente se distribuyen. Por lo tanto, un riesgo común en los contenedores, se debe a la versión de la imagen que se usa, ya que esta puede tener vulnerabilidades que pueden afectar a todo el contenedor.

Además de los defectos de software, las imágenes pueden tener defectos en la configuración, por ejemplo, una imagen puede incluir un *daemon SSH*, que puede exponer al contenedor a un riesgo de red innecesario.

Uno de los escenarios de más alto riesgo, es la ejecución de software que no es de confianza, la facilidad y reutilización de los contenedores aumenta la probabilidad de que se ejecuten imágenes de fuentes externas, que no se encuentre validadas o no sean confiables, como en [2].

b) Conexiones inseguras: Si las conexiones a los registros se realizan a través de canales inseguros, el contenido de las imágenes están sujetas a los mismos riesgos de confidencialidad que cualquier otro dato transmitido, además de estar sujeto a riesgos como los ataques conocidos como el *man-in-the-middle* (ataque del hombre en el medio), que intercepta el tráfico de red, destinado a los registros para robar credenciales de desarrollador o administrador dentro de ese tráfico, además de proporcionar imágenes fraudulentas u obsoletas a los contenedores.

En la mayoría de los entornos en contenedores, el tráfico entre nodos individuales se enruta a través de una red de superposición virtual, esta red normalmente se administra por el orquestador, y obstaculiza de alguna manera las herramientas de seguridad y administración de red. Aunque una red de superposición cifrada ayudaría, a proporcionar mayor seguridad, esta también puede crear un escenario de "ceguera", donde las organizaciones no podrían supervisar eficazmente el tráfico de sus propias redes.

Un riesgo aun mayor, es el tráfico que diferentes aplicaciones comparten en las mismas redes virtuales, si las aplicaciones de diferentes niveles de sensibilidad, como un sitio web público y una aplicación interna, pueden estar expuestas a un mayor riesgo de ataque de red. Ya que, si se ve comprometida la aplicación web, puede ser un conducto para ingresar a la aplicación interna de la organización [7].

II. MÉTODOS

Para el presente estudio se han aplicado los siguientes métodos y técnicas de investigación:

TABLA I. METODOLOGÍA

Método	Hardening	Docker
Deducción	Investigación a partir de un conjunto dado de premisas, un modo de pensamiento que va de lo general a lo específico.	
	En el presente artículo parte de la planificación, antes de la instalación, configuración e implementación de los contenedores, para dar cumplimiento a las políticas organizacionales relevantes para que el contenedor sea lo más seguro posible.	
Analítico	Consiste en la desmembración de un todo, descomponiéndolo en sus partes o elementos.	
	Dado que los contenedores se basan principalmente en componentes ya presentes en los sistemas operativos, las herramientas y técnicas para realizar análisis forenses, se basan en un análisis analítico que parte de la separación de un todo, en resultados específicos.	
Sintético	Es un proceso que tiende a reconstruir un todo, a partir de los elementos.	
		Estudiar de las políticas internas de las organizaciones, para considerar los requerimientos de seguridad de los contenedores.
Técnicas de investigación		
Investigación documental	Para el presente estudio, se usó la investigación documental, la cual fundamenta la recopilación de información, a partir de documentos relacionados con seguridad, hardening y contenedores (Docker), provenientes de diferentes autores. El estudio se realizó a partir de conocimientos y teorías obtenidas.	

Fuente: Elaboración propia

Para el caso de hardening se usó los dominios de COBIT lo que ayudara al diseño, los mismos se detallan a continuación.

- Gestión de seguridad
- Gestión de cambios
- Gestión de operaciones
- Gestión de riesgos
- Supervisión, evaluación.

La gestión de seguridad, tiene por objetivo definir, operar y supervisar, la información sobre los contenedores Docker, la gestión de cambios permitirá gestionar todos los cambios estándar, o de mantenimiento en relación, a los procesos y aplicaciones que se realizaran en los contenedores.

El propósito del dominio de supervisión y evaluación, permite ofrecer confianza en las operaciones que se ejecutan en los contenedores. La intención de la gestión de operaciones se ejecutan las actividades y procedimientos operativos.

III. RESULTADOS

A continuación, se detalla el esquema que se propone para el proceso de hardening para contenedores Docker:

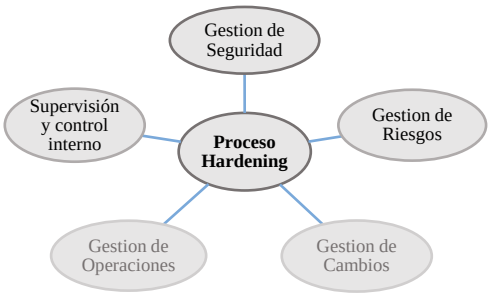

Fig. 3. Modelo de hardening
Fuente: Elaboración propia, adaptado de COBIT

En la figura 4, se muestra el proceso de hardening, a nivel host y red, para un contenedor Docker de manera general.

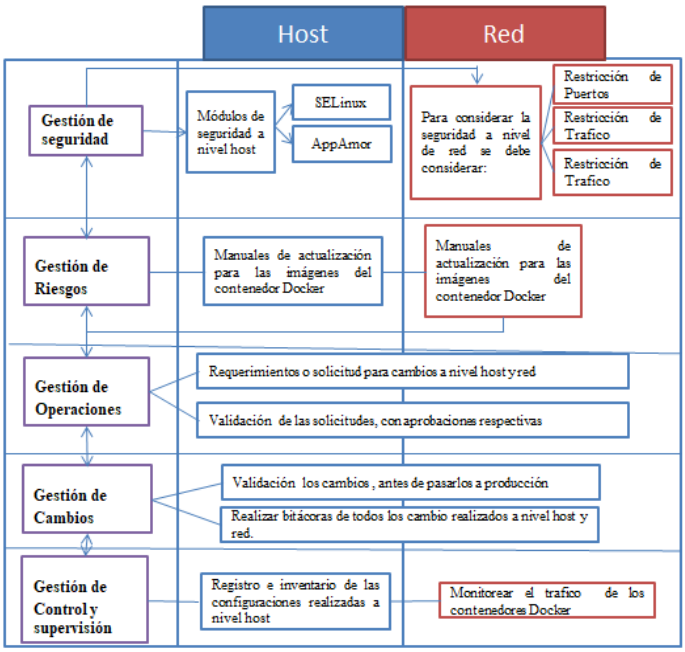

Fig. 4. Modelo hardening para contenedores
Fuente: Elaboración propia

Si bien las nuevas tecnológicas, representan un nuevo reto para la seguridad de la información, las organizaciones optan por implementarlo, por los beneficios que aportan, a su entorno, por lo tanto el primer paso para establecer una postura de seguridad en los contenedores es reducir, la superficie de ataque realizando un procedimiento de hardening, algunas de las opciones presentadas anteriormente son efectivas para que los contenedores se encuentren, menos expuestas a vulnerabilidades a nivel host, cuando estas aún se están ejecutando, la cual es considerado un eslabón débil en lo que se refiere a contenedores.

Uno de los mayores desafíos para la seguridad en un contenedor se encuentra dentro de la fase, tiempo de ejecución. Esta fase del ciclo de vida del contenedor exige una plataforma de seguridad aplicada específicamente para esta etapa.

IV. CONCLUSIONES

Hace tiempo, se dejaba de lado la seguridad, priorizando otros temas como el rendimiento, reducción de costes, etc., hoy en día, la seguridad se ha convertido en un pilar fundamental para las empresas, debido a la prioridad y confidencialidad de su información.

En conclusión, la seguridad de un contenedor depende de la configuración y medidas de seguridad, que se configuran al momento de realizar el proceso de hardening, fundamentalmente al anfitrión kernel del contenedor.

REFERENCIAS

[1] D.S. Espitita “Telefónica. S.A.”, 2018.
[2] J. Morello, and M. Souppaya, And K. Scarfone, “Aplication container security guide,” National Institute Of Standars and Technology, 2017.
[3] A. G. Caiza “Diseño de un proceso de hardening de servidores para una institución financiera del sector público,” Ecuador, 2019, pp34-38.
[4] B. Hernández “Seguridad en Docker”, España, 2018.
[5] Y. Naik. “CIS, Center for internet security,” Agosto 2017.
[6] X. Gines. “Seguridad en Docker, “inédito.
[7] C. “CIS Center for internet security”, julio 2019.

Breve CV de la autora

Danitza Helen Vallejos Quiñones es Ingeniera de Sistemas por la Escuela Militar de Ingeniería (La Paz, 2015), Diplomada en Gestión de incidentes de tecnología de la información por la Escuela de Gestión Pública (La Paz, 2017). Ejerce actualmente como Operadora de Infraestructura Tecnológica, en el Banco Nacional de Bolivia. Anteriormente como técnica de proyectos especiales en DATACOM S.A. Email: helen.vallejos.q@gmail.com.