Maestría en

CIBERSEGURIDAD

Trabajo previo a la obtención de título de
Magister en Ciberseguridad

AUTOR/ES:

Cristhian Gabriel Campos González
Cristhian Jeampier Espín Villafuerte
Diego Armando Hurtado Recalde
Johan Steven Navarrete Quirola

TUTOR/ES:

Alejandro Cortés L.

Iván Reyes Chacón

TEMA

ANÁLISIS DE SEGURIDAD EN ENTORNO DOCKER: AMENAZAS Y DISEÑO DE CONTROLES PARA MITIGAR ATAQUES EN ENTORNOS CORPORATIVOS

Quito - Ecuador

Diciembre – 2025

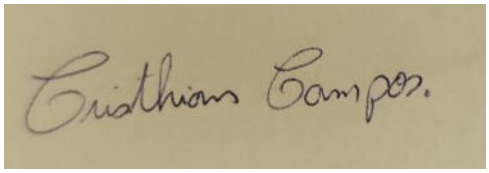
REINVENTEMOS EL FUTURO

Certificación de autoría

Nosotros, **(Cristhian Jeampier Espín Villafuerte, Cristhian Gabriel Campos González, Diego Armando Hurtado Recalde, Johan Steven Navarrete Quirola),** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

Firma
Cristhian Jeampier Espín Villafuerte



Firma
Cristhian Gabriel Campos González

Firma
Diego Armando Hurtado Recalde

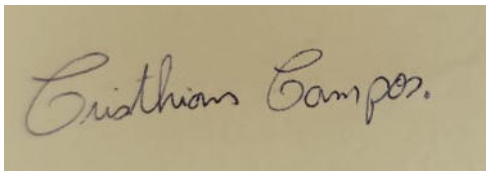
Firma
Johan Steven Navarrete Quirola

Autorización de Derechos de Propiedad Intelectual

Nosotros, **(Cristhian Jeampier Espín Villafuerte , Cristhian Gabriel Campos González, Diego Armando Hurtado Recalde, Johan Steven Navarrete Quirola)**, en calidad de autores del trabajo de investigación titulado ***ANÁLISIS DE SEGURIDAD EN ENTORNO DOCKER: AMENAZAS Y DISEÑO DE CONTROLES PARA MITIGAR ATAQUES EN ENTORNOS CORPORATIVOS***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, (diciembre 2025)

Firma
Cristhian Jeampier Espín Villafuerte



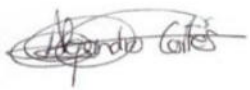
Firma
Cristhian Gabriel Campos González

Firma
Diego Armando Hurtado Recalde

Firma
Johan Steven Navarrete Quirola

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés e Iván Reyes,** declaramos que: **(Cristhian Jeampier Espín Villafuerte, Cristhian Gabriel Campos González, Diego Armando Hurtado Recalde, Johan Steven Navarrete Quirola)** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés L.

Maestría en Ciberseguridad

Iván Reyes Ch.

Maestría en Ciberseguridad

DEDICATORIA

A mi familia, por su apoyo permanente, su paciencia y la confianza que depositaron en mí a lo largo de este proceso. Los valores que me inculcaron y el respaldo que siempre me han brindado han sido pilares fundamentales para la realización de este trabajo.

A mi esposa e hija, por su acompañamiento constante, por su comprensión ante los momentos de dedicación exclusiva que este proyecto requirió y por su firme convicción en mis capacidades. Su aliento, orientación y fortaleza fueron determinantes para avanzar y culminar este desafío académico.

A todas las personas que, de alguna manera, contribuyeron al desarrollo de este proyecto, ya sea mediante consejos, colaboración o palabras de motivación. Cada apoyo recibido ha sido parte esencial en la consecución de este logro.

Cristhian Gabriel Campos González

Dedico este trabajo con profundo cariño y gratitud a mis padres, que han sido un pilar fundamental en mi vida, gracias por su ejemplo, por su apoyo y por enseñarme a luchar con constancia y humildad y por acompañarme con palabras de aliento durante todo este proceso aun cuando parecía imposible avanzar.

A mis hermanos, quienes han sido una fuente constante de apoyo e inspiración en mi vida, gracias por acompañarme siempre y darme una mano sobre todo en los momentos difíciles.

Finalmente, me lo dedico a mi mismo, por la perseverancia, el esfuerzo y la determinación puesta en cada paso y por recordarme a mi mismo que todo se puede lograr si se trabaja con dedicación

Cristhian Jeampier Espín Villafuerte

A mi hijo Diego y a mi hija Alma, quienes cada día han sido mi pilar de motivación para continuar esforzándome durante todo este proceso de formación como Máster en Ciberseguridad. A mi esposa Josselyn, cuyo apoyo en momentos clave contribuyó a que pudiera continuar este camino académico.

A mi madre, que desde pequeño me enseñó que el esfuerzo y el trabajo duro nos permiten alcanzar grandes metas y ha sido siempre mi mayor inspiración para seguir preparándome; a mis hermanas, por su amor, cariño y apoyo incondicional en los momentos más difíciles; a mis sobrinos, a quienes amo como hijos, por recordarme que con un objetivo claro y compromiso siempre podemos sacar lo mejor de nosotros; a mi suegro y a mi hermano, quienes desde el cielo me brindan fuerza y fe para continuar adelante y ser un mejor ser humano.

A mis compañeros de trabajo, por tenderme la mano y apoyarme en temas técnicos de ciberseguridad. A mis compañeros de grupos de trabajo en las diferentes clases de esta maestría, a mis compañeros de grupo de este trabajo de fin de máster, profesores y tutores.

A todos ustedes, con profundo cariño y gratitud, dedico este logro.

Diego Armando Hurtado Recalde

A toda mi familia, por su apoyo incondicional, por creer en mí y ser un pilar fundamental en cada etapa de mi vida. A mis padres, por ser mi ejemplo, por todo su apoyo, sus ilusiones de verme salir adelante y por enseñarme que la constancia y el esfuerzo pueden traspasar cualquier frontera. A mis hermanos, por sus consejos, alientos y momentos incomparables, quiero que sepan que siempre les tengo presentes. A mis sobrinos, por haberles visto crecer y ahora sentir que poco a poco están forjando su futuro con cada paso que dan.

De manera muy especial, dedico a mi esposa, por ser ejemplo de madurez y

superación, por su gran amor, por ser mi compañera de vida y corazón de nuestro hogar, por siempre tener su apoyo y dar lo mejor de ella en cada momento juntos.

A todos quienes han sido parte de este proceso, amigos, compañeros de trabajo, compañeros de grupo, docentes y tutor de la maestría, que han hecho posible que pueda alcanzar este logro.

Johan Steven Navarrete Quirola

AGRADECIMIENTO

Agradezco profundamente a mis padres, hermanos, esposa e hija, quienes con su esfuerzo y cariño me brindaron un apoyo incondicional e hicieron posible este logro.

A mis compañeros de proyecto, cuyo compromiso y dedicación nos permitieron culminar juntos esta importante fase; les deseo siempre lo mejor.

A nuestros tutores, por su guía permanente a lo largo de cada uno de los módulos cursados, así como por el tiempo, la dedicación y el apoyo brindados durante todo el desarrollo de este proyecto.

Y, finalmente, a la UIDE, por ofrecerme la oportunidad de crecer, así como los recursos y conocimientos que fortalecieron mi formación profesional.

Mi gratitud para todos quienes, de una u otra forma, contribuyeron y estuvieron presentes en mi vida para alcanzar esta significativa meta.

Cristhian Gabriel Campos González

Agradezco profundamente a Dios por toda la fortaleza, paciencia y claridad que me brindo para poder culminar con este trabajo de tesis que representa un gran logro en el ámbito académico y personal.

A mis padres y hermanos, les doy las gracias por todo su apoyo, acompañamiento incondicional y por toda la confianza depositada en mis capacidades, a ellos toda mi gratitud ya que, sin su impulso constante en cada etapa de este proceso, este logro no sería posible

A mis docentes, quienes con sus enseñanzas y experiencia contribuyeron de manera directa a mi formación como profesional.

Finalmente, a mis compañeros de tesis por su colaboración, compromiso y apoyo en la elaboración de este trabajo de tesis, cuyo trabajo en conjunto enriqueció esta investigación.

Cristhian Jeampier Espín Villafuerte

Agradezco profundamente a mi hijo Diego y a mi hija Alma, cuya motivación diaria me impulsó a perseverar y a cumplir cada etapa de este proceso formativo.

Agradezco a Josselyn por su colaboración en la organización familiar durante este proceso, lo cual me permitió avanzar con responsabilidad y enfoque en este proyecto.

Extiendo mi gratitud a mi madre, por enseñarme desde pequeño el valor del esfuerzo y del trabajo duro, principios que han guiado mi desarrollo académico y profesional. A mis hermanas, gracias por su acompañamiento constante, su cariño y su apoyo incondicional en los momentos más exigentes.

A mis sobrinos, quienes con su energía y ejemplo me recordaron que siempre podemos superarnos cuando actuamos con propósito. A mi suegro y a mi hermano, cuya memoria me brinda fuerza y fe para seguir adelante, y cuyas vidas continúan siendo una inspiración para ser un mejor ser humano.

A mis compañeros de trabajo, les agradezco su apoyo diario, su colaboración en temas técnicos de ciberseguridad y por hacer más llevadero este camino académico.

A mi empresa Telefónica y a la UIDE, gracias por brindarme la oportunidad de formarme en esta carrera tan significativa para mi crecimiento profesional.

A todos ustedes, les expreso mi más sincero agradecimiento por ser parte esencial de este logro.

Diego Armando Hurtado Recalde

Agradezco a mis padres, mi esposa, mis hermanos, mis sobrinos, a mis demás familiares y amigos, por todas sus palabras de aliento y motivación, que me han ayudado a conseguir este logro.

A mis compañeros de grupo, por su organización, constancia, interés y esfuerzo, todo

esto lo conseguimos juntos y les expreso mi gratitud por todo su apoyo.

A mis docentes y tutor de la maestría, por compartir sus conocimientos y enfocarnos en entornos reales a los que nos podemos enfrentar en el mundo de la Ciberseguridad.

Agradezco a la UIDE, por la oportunidad de ser parte de su historia, por su acompañamiento desde el primer contacto hasta el final de la maestría, los profesionales a cargo de cada etapa demuestran la calidad y el compromiso que tiene la Universidad con sus estudiantes.

Johan Steven Navarrete Quirola

RESUMEN

Esta investigación se centra en el análisis de amenazas y vulnerabilidades en entornos Docker corporativo, con el objetivo de desarrollar controles de seguridad que contribuyan a la protección de la información y garanticen la continuidad operativa.

Docker ofrece un gran número de ventajas en cuanto a términos de portabilidad, escalabilidad y eficiencia en el despliegue de aplicaciones, pero también conlleva a la introducción de riesgos críticos asociados a malas configuraciones, escalamiento de privilegios, exposición de datos sensibles, inserción de malware en imágenes provenientes de repositorios no confiables, etc.

Bajo esta premisa, se realiza una evaluación de riesgos basada en marcos normativos reconocidos aplicados a la ejecución de contenedores y al uso de imágenes, con el propósito de identificar vectores de ataque, así como escenarios de alto impacto en infraestructuras corporativas. También se proponen medidas de prevención en base al escaneo de imágenes mediante el uso de varias herramientas, así como, monitoreo continuo. De forma complementaria, se plantea un marco de buenas prácticas en la gestión de contenedores, actualización permanente de imágenes y aplicación de estándares internacionales de ciberseguridad.

Este estudio contribuye al marco de la ciberseguridad corporativa al brindar una aproximación integral que une el análisis de amenazas, evaluación de riesgos y diseño de controles, basados en los lineamientos de seguridad de la información.

Palabras Claves: Docker, seguridad, contenedores, vulnerabilidades, controles de mitigación

ABSTRACT

This research focuses on analyzing threats and vulnerabilities in corporate Docker environments, with the aim of developing security controls that contribute to information protection and ensure operational continuity.

Docker offers a number of advantages in terms of portability, scalability, and efficiency in application deployment, but it also introduces critical risks associated with misconfigurations, privilege escalation, exposure of sensitive data, insertion of malware into images from untrusted replicas, etc.

Under this premise, a risk assessment is carried out based on recognized regulatory frameworks applied to container execution and image use, with the purpose of identifying attack vectors, as well as high-impact scenarios in corporate infrastructures. Prevention measures are also proposed based on image scanning using various tools, as well as continuous monitoring.

Complementarily, a framework of best practices in container management, permanent image updating, and the application of international cybersecurity standards is proposed.

This study contributes to the corporate cybersecurity framework by providing a comprehensive approach that combines threat analysis, risk assessment, and control design, based on information security guidelines.

Keywords: Docker, security, containers, vulnerabilities, mitigation controls.

TABLA DE CONTENIDOS (Índice)

LISTA DE TABLAS (Índice de tablas)

LISTA DE FIGURAS (Índice de figuras)

CAPITULO 1:

1. INTRODUCCIÓN

1.1. Definición del proyecto

El proyecto se centra en el "Análisis de Seguridad en Entorno Docker: Amenazas y Diseño de Controles para Mitigar Ataques en Entornos Corporativos" se enfoca en examinar de manera exhaustiva las implicaciones de seguridad asociadas con la implementación y operación de contenedores Docker en contextos empresariales. Docker, como una plataforma líder en la virtualización de contenedores, ha revolucionado la forma en que las organizaciones desarrollan, despliegan y escalan aplicaciones, permitiendo una mayor eficiencia y portabilidad. Sin embargo, esta adopción masiva también introduce riesgos significativos, ya que los entornos corporativos manejan datos sensibles, sistemas críticos y una infraestructura interconectada que puede ser vulnerable a exploit si no se gestiona adecuadamente.

En primer lugar, el análisis se centrará en la identificación de amenazas reales que afectan a los entornos Docker en el ámbito corporativo. Estas amenazas incluyen, pero no se limitan a, vulnerabilidades en las imágenes de contenedores, como la inclusión de software malicioso o desactualizado durante la construcción de imágenes; ataques de breakout de contenedores, donde un atacante escapa del aislamiento para acceder al host; y exposiciones relacionadas con la configuración inadecuada de redes, volúmenes y permisos. Además, se considerarán riesgos derivados de la cadena de suministro, como la inyección de malware en repositorios públicos de imágenes (por ejemplo, Docker Hub), y amenazas avanzadas como el abuso de privilegios root o la explotación de APIs de Docker. Este enfoque se basará en un revisión de casos reales de brechas de seguridad reportadas en empresas, integrando datos de fuentes como CVE (Common Vulnerabilities and Exposures) y reportes de ciberseguridad para

contextualizar los riesgos en escenarios corporativos, donde el impacto puede extenderse a la pérdida de datos confidenciales, interrupciones operativas o incumplimientos regulatorios.

A continuación, el proyecto evaluará vulnerabilidades críticas mediante una metodología estructurada que combine análisis estático y dinámico. Esto involucrará el uso de herramientas especializadas para escanear imágenes y contenedores en busca de debilidades, como Trivy, Clair o Anchore, y la simulación de escenarios de ataque en entornos controlados para medir la exposición real.

Finalmente, el diseño de controles preventivos representará el núcleo propositivo del proyecto, ofreciendo un conjunto de recomendaciones accionables para mitigar los ataques identificados. Estos controles abarcarán mejores prácticas en múltiples capas: desde la adopción de imágenes base mínimas y firmadas digitalmente hasta la implementación de políticas de seguridad para restringir el comportamiento de los contenedores. Se propondrán estrategias para la monitorización continua, como el uso de herramientas de logging - auditing y la integración de pipelines CI/CD seguros para automatizar escaneos de vulnerabilidades.

1.2. Justificación e importancia del trabajo de investigación

Docker se ha consolidado como una herramienta fundamental para el despliegue ágil de aplicaciones en entornos corporativos. Sin embargo, su adopción implica riesgos importantes, como escape de contenedores, escalamiento de privilegios y malware en imágenes. Este trabajo es relevante porque permite reducir la superficie de ataque, garantizar la continuidad de los servicios y asegurar el cumplimiento de normativas y estándares internacionales de ciberseguridad.

En la actualidad, el uso de contenedores como medio de virtualización óptima y ligera han transformado la metodología de desarrollo, el despliegue de aplicaciones y su escalabilidad,

por lo cual, se ha visto reflejado en incremento del uso de plataformas como Docker, que se ha convertido en una herramienta primordial por su flexibilidad, portabilidad y eficiencia. A pesar de todos sus beneficios, también recae en el incremento de ataques, los cuales logran comprometer la disponibilidad, integridad y confidencialidad de los servicios de una empresa.

Como justificación del proyecto se da importancia a la relevancia de la tecnología actual y sus ambientes de desarrollo, también que existe una superficie de ataque amplia, por lo cual se produce la carencia de controles de seguridad que se especialicen en este tipo de ambientes y por último, que la es necesaria la correcta contribución y formación académica, con las prácticas necesarias para una correcta protección de la infraestructura por parte de los profesionales.

La importancia de realización de este proyecto, se tiene el aprender y profundizar los conocimientos de como mitigar los riesgos en ambientes reales, también el de aportar seguridad en las organizaciones, para que puedan alinearse a los estándares de ciberseguridad, para lo cual se debe fortalecer la seguridad en estos ambientes, con el objetivo de reducir costos de implementación, cerrar las brechas de seguridad e incremento de la credibilidad de la empresa para bienestar y tranquilidad de los clientes.

1.3.Alcance

El estudio se enfocará en la identificación de amenazas y vulnerabilidades presentes en entornos Docker corporativos, analizando los riesgos asociados tanto a los contenedores en ejecución como a las imágenes utilizadas para su despliegue. Esto permitirá determinar los posibles vectores de ataque y escenarios críticos que puedan afectar la seguridad de la infraestructura corporativa.

Asimismo, se realizará una evaluación de riesgos que contemple la exposición de información sensible, la integridad de los contenedores y las configuraciones aplicadas, con el

fin de priorizar las áreas que requieren mayor atención y establecer medidas preventivas adecuadas.

Finalmente, se propondrán controles de seguridad y buenas prácticas para la gestión segura de Docker en entornos corporativos. Cabe mencionar que el alcance del estudio no incluirá plataformas avanzadas de orquestación como Kubernetes ni servicios de nube pública; el análisis estará limitado a entornos Docker standalone dentro de la infraestructura corporativa.

1.4. Objetivos

1.4.1. Objetivo general

- Analizar los riesgos y amenazas de seguridad inherentes al uso de contenedores Docker en entornos corporativos, con el propósito de diseñar, proponer y validar un conjunto integral de controles técnicos, organizativos y procedimentales que permitan mitigar eficazmente los vectores de ataque más relevantes, garantizando así la confidencialidad, integridad y disponibilidad de los sistemas desplegados bajo arquitecturas de contenedores.

1.4.2. Objetivo especifico

- Examinar los fundamentos de la arquitectura de contenedores Docker y su integración con microservicios, identificando las particularidades técnicas que influyen en la superficie de exposición y el modelo de amenazas en entornos corporativos.
- Implementar un entorno de laboratorio controlado en el que se despliegue una arquitectura de microservicios sobre Docker, con el objetivo de simular ataques comunes.

- Realizar un análisis e identificar y clasificar las amenazas y vulnerabilidades más en entornos Docker.
- Identificar las malas prácticas de configuración presentes en la orquestación de microservicios, así como la interacción entre contenedores y servicio externos.
- Recomendar controles preventivos y correctivos que mitiguen vulnerabilidades comunes.
- Proponer un conjunto de recomendaciones estratégicas y operativas para la adopción segura de Docker y microservicios en entornos corporativos.

CAPITULO 2:

2. REVISIÓN DE LITERATURA

2.1. Estado del Arte

El empleo de contenedores mediante Docker provee eficiencia, portabilidad y agilidad, pero no todo es bueno ya que también crean riesgos de seguridad, bajo esta premisa, el articulo denominado “Seguridad de Contenedores Docker mediante Procesos de Hardening” elaborado por Vallejos Quiñonez (2021) indica que los contenedores son vulnerables cuando se utilizan las configuraciones que vienen por defecto o se usan imágenes sin control, por lo que propone un proceso de hardening que abarca desde la construcción de la imagen hasta su debida ejecución, también brinda recomendaciones para la configuración del host así como permisos y practicas seguras de despliegue. (Vallejos Quiñonez, 2021).

Hablando del enfoque en el pentesting que revela vulnerabilidades en contenedores, el estudio denominado “Propuesta de metodología de Pentesting en contenedores Docker” llevado a cabo por Hurtado Sáenz (2021) explica la

implementación de dos entornos Docker, el primero con un servidor web Apache y el otro con WordPress + MySQL, en donde se realizaron pruebas de seguridad. Los resultados obtenidos mostraron que una mala configuración tanto del contenedor como del firewall abren una ventana para que se puedan realizar ataques, por lo que se demuestra la importancia de realizar una configuración segura tanto del contenedor como del entorno de red como medida preventiva a ataques. (Hurtado Sáenz, 2021).

Otro aporte importante es el realizado en la Universidad de Guayaquil por Antepara Reyes & Villamar Flores (2020) titulado "Análisis de vulnerabilidades y definición de contramedidas en la seguridad de aplicaciones basadas en contenedores usando la herramienta Docker", en donde se elaboró un laboratorio con Docker para evaluar vulnerabilidades dentro de aplicaciones web en contenedores, se demostró que el uso incorrecto del entorno puede afectar la integridad de los datos y la seguridad del sistema, por lo que recomiendan realizar una buena configuración, aislamiento y gestión de contenedores, lo que claramente evidencia que manejar contenedores con configuraciones básicas o preestablecidas son un grave problema de seguridad (Antepara Reyes & Villamar Flores, 2020).

Finalmente, con respecto a las buenas prácticas de seguridad en contenedores existen guías oficiales como la del CCN-CERT en donde se recomienda analizar vulnerabilidades en imágenes que provengan de registros ya sean públicas o privados, evaluar malas configuraciones o incluso secretos expuestos en imágenes o en el host, así como sugieren auditorias, escaneo, monitoreo continuo y control de acceso. (CCN-CERT, 2022).

En resumen, la seguridad en entornos Docker requieren de un enfoque holístico que se base en múltiples capas, por mencionar algunas como: selección adecuada y escaneo de imágenes, configuración segura, hardening del entorno y runtime,

monitoreo permanente y gestión de vulnerabilidades. Con esto lo que se busca es reducir el riesgo de ataques especialmente en entornos corporativos y constituye la base teórica para llevar a cabo un análisis de amenazas y diseño de controles de seguridad.

2.2. Marco Teórico

2.2.1. Conceptos fundamentales

En este apartado se presenta la contextualización de los conceptos principales que sustentan el presente estudio, puesto que el tema de investigación abarca el análisis de seguridad en torno a Docker, es importante definir las bases teóricas que tengan relación con la virtualización, contenedores, la arquitectura de Docker y la seguridad aplicada a estas tecnologías.

2.2.1.1.Definición de virtualización

Se define como virtualización al proceso que crea una capa de abstracción en el hardware de una computadora, esto permite que los elementos de hardware que se encuentran conformando a una sola computadora como puede ser el procesador, memoria, almacenamiento, etc., se separen en diferentes computadoras virtuales, lo que comúnmente se conoce como máquina virtual. Dicha máquina virtual ejecuta su propio sistema operativo y actúa como una computadora totalmente independiente que se ejecuta en una parte del hardware existente. La virtualización constituye la base de la computación en la nube. (IBM, ¿Qué es la virtualización?, s.f.)

2.2.1.2.Capa de virtualización

Es una capa de software que se agrega entre el hardware y el sistema operativo, esta capa permite que los sistemas operativos puedan correr dentro de máquinas virtuales en un servidor físico único, esto garantiza la partición y la compartición de los recursos físicos disponibles dentro de un ordenador como pueden ser el CPU, la

memoria y dispositivos de almacenamiento de entrada y salida.

Esta capa de virtualización contiene un supervisor (hipervisor) que es el que establece los recursos de hardware, en otras palabras, se puede decir que hipervisor es el programa de control maestro que cuenta con el nivel más alto de privilegios y administra varios sistemas operativos (sistemas operativos huéspedes). (García y Fernández, 2011)

2.2.1.3.Máquina virtual

Es una emulación o una representación virtual de un ordenador físico que emplea software en lugar de hardware para poner en funcionamiento programas o incluso implementar aplicaciones. Al utilizar los recursos de una maquina física entre esos están la memoria, la CPU, el almacenamiento o la interfaz de red, brindan a las empresas la posibilidad de ejecutar varias máquinas de forma virtual con distintos sistemas operativos en un solo dispositivo. (Susnjara y Smalley, s.f.)

2.2.1.3.1. Beneficios de las máquinas virtuales

Utilizar máquinas virtuales tanto en entornos académicos, profesionales o empresariales tiene varias ventajas significativas, ente los cuales se pueden resaltar los siguientes:

- **Ahorro de costos:** Este es uno de los beneficios más grandes en cuanto al uso de máquinas virtuales ya que representa una reducción de inversión considerable en cuanto a adquisición en hardware, ya que, en lugar de adquirir varios equipos físicos, se puede ejecutar diferentes sistemas operativos en un solo ordenador.
- **Escalabilidad y flexibilidad:** Las máquinas virtuales brindan la posibilidad de crear y eliminar entornos de forma rápida según se necesite en el momento.

- **Aislamiento y seguridad:** El aislamiento que brinda una máquina virtual es fundamental para minimizar los riesgos de seguridad ya que cada máquina virtual se comporta como un entorno distinto, esto significa que, si ocurre algún ataque o fallo, el sistema principal no se verá afectado.
- **Recuperación ante desastres y backups:** La posibilidad de crear copias de seguridad y restaurar los sistemas en caso de fallos resulta importante para los planes de continuidad de una empresa o negocio.
- **Optimización de recursos:** La virtualización permite aprovechar en su totalidad el hardware disponible, evitando que quede infrautilizado. (UNIR, 2025)

2.2.1.4.Definición de contenedores

Se describe como una forma de virtualización de un sistema operativo y se puede utilizar para ejecutar cualquier cosa como un microservicio o una aplicación de mayor tamaño. Un contenedor puede almacenar todos los ejecutables, código binario, bibliotecas y archivos de configuración necesarios, sin embargo, los contenedores no contienen imágenes del sistema operativo por lo que los hace más ligeros y portátiles lo que representa una sobrecarga significativamente menor. (NetApp, n.d.)

2.2.1.4.1. Beneficios de los contenedores

Dotar de seguridad a los contenedores ofrece varios beneficios, entre los cuales se resaltan los siguientes:

- **Implementación de software rápida y eficiente:** Automatizar los procesos desde el equilibrio de carga hasta la orquestación, posibilita un desarrollo e implementación de software más exactos sin que la integridad de la red se vea comprometida.
- **Menores costos generales:** Los contenedores no necesitan recursos

elevados del sistema, lo que desemboca en un menor gasto.

- **Escalabilidad mejorada:** Si se implementan aplicaciones que operen en contenedores en diferentes sistemas operativos, los equipos de TI pueden optimizar el desarrollo, las pruebas y la producción. (Puzas, 2024)

2.2.1.4.2. Funcionamiento de un contenedor

Funcionan gracias al aislamiento de procesos y virtualización de Linux, incluyendo grupos de control para establecer recursos entre procesos y espacios de nombres, esto con el propósito de limitar el acceso o visibilidad de un proceso a otra área del sistema. Además, estos contendores admiten que diferentes aplicaciones compartan los recursos de una única instancia del sistema operativo principal. (IBM, n.d.)

2.2.1.5.Máquinas virtuales vs contenedores

Tanto los contenedores como las máquinas virtuales son tecnologías que posibilitan que las aplicaciones funcionen independientemente sin la necesidad de utilizar los recursos de la infraestructura de TI. Se entiende por contenedor a un paquete de código de software que almacena el código de una aplicación, así como sus bibliotecas y demás dependencias. La utilización de contenedores hace posible que las aplicaciones sean portátiles, por lo que el mismo código se puede ejecutar en diferentes dispositivos.

Po otro lado, una máquina virtual es una copia digital de una máquina física, gracias a esta característica se puede disponer de varias máquinas virtuales con sistemas operativos propios que se ejecuten en el mismo sistema operativo e incluso se puede crear una máquina virtual que almacene todo lo indispensable para ejecutar una aplicación. (AWS, n.d.)

2.2.1.5.1. Diferencias entre contenedor y máquina virtual

La Tabla 1 muestra las diferencias principales entre una máquina virtual y un contenedor.

Tabla 1

Diferencias entre máquina virtual

Características	**Contenedor**	**Máquina virtual**
Virtualización	Sistema operativo.	Infraestructura física subyacente.
Tecnología	El motor de contenedores es el que coordina los recursos con el sistema operativo subyacente.	El hipervisor se organiza con el sistema operativo o el hardware.
Tamaño	Mas ligero.	Mas grande.
Control	Se evidencia un control menor del entorno fuera del contenedor.	Se evidencia un nivel de control superior en todo el entorno.
Flexibilidad	Es más flexible ya que puede migrar entre entornos.	Es menos flexible ya que la migración presenta muchas dificultades.
Escalabilidad	Presenta la posibilidad de escalar mediante el uso de microservicios.	El escalado es costoso debido a las instancias en la nube.

Nota: Fuente: Amazon Web Services (AWS)

2.2.1.6.Orquestación de contenedores

Es el proceso mediante el cual se automatizan el despliegue, la gestión y la coordinación de todos los contenedores empleados para la ejecución de una aplicación, se acostumbra a usar varias tecnologías de orquestación entre las cuales se destaca la

de Kubernetes, con el propósito de gestionar el ciclo de vida de aplicaciones contenerizadas iniciando en su desarrollo y despliegue hasta finalizar con las pruebas y monitorización. (Datadog, s.f.)

La orquestación acelera el despliegue, escalado, configuración, conexión en red y seguridad en contenedores y garantiza una alta disponibilidad de aplicaciones en contenedores a través de la detección automática de fallos y la corrección a los mismos. Los orquestadores de contendores se basan en un plano de control que es un conjunto común de componentes que brinda un mecanismo para emplear políticas desde un controlador central a cada contenedor ya que proporciona una interfaz que se conecta a los contenedores y lleva a cabo diferentes funciones de gestión. (Datadog, s.f.)

2.2.1.6.1. Kubernetes

Es una plataforma de orquestación de contenedores desarrollada para implementar, escalar y administrar aplicaciones dentro de contenedores en cualquier lugar, además, automatiza las tareas operativas de la administración de contenedores ya que organiza los deployments con una API , los agrupa en contenedores en Pods con el propósito de escalar dependiendo de la demanda y disponibilidad de recursos, también incluye comandos para implementar aplicaciones, actualizarlas y escalarlas con el propósito de que se adapten a las necesidades de una organización. (Cloud, n.d.)

Kubernetes gestiona como se implementan, escalan y operan las aplicaciones mediante la orquestación de cargas de trabajo en contenedores por medio de la infraestructura de la nube, definiendo así los puntos críticos de control (control de acceso, segmentación de la red, tiempo de ejecución, etc.), mientras mayor sea la coordinación de todo el entorno de la nube los niveles de seguridad de una organización se elevaran, aquí se incluyen factores como la infraestructura reforzada,

gestión de identidades y los sistemas de monitorización para detectar desviaciones y comportamientos extraños en producción. (NETWORK, 2025)

2.2.1.6.2. Comparación entre Kubernetes y Docker

Docker permite almacenar en contenedores todo lo necesario para ejecutar una aplicación cuando y donde sea necesario por lo que es necesario poder administrar estos contenedores y es aquí donde entra en acción los Kubernetes ya que son responsables de trasladar y entregar los contenedores a las ubicaciones en donde van a ser utilizados. Estas dos plataformas pueden ser utilizadas para poner aplicaciones en contenedores y ejecutarlas. La gran diferencia entre ambos recae en el rol que cada uno cumple, Docker es utilizado para el empaquetado y distribución de aplicaciones dentro de contenedores mientras que Kubernetes utiliza a Docker para implementar, administra y escalar aplicaciones en contenedores. (Cloud, n.d.)

2.2.2. Docker como plataforma

Docker es un sistema operativo para contenedores que permite desarrollar, probar e implementar aplicaciones de forma rápida ya que Docker empaqueta software en unidades llamadas contenedores que almacenan todo lo necesario para que el software se ejecute de forma correcta, aquí se incluyen elementos como bibliotecas, herramientas de sistema, código y versión ejecutable.

Con esta plataforma se puede implementar y modificar la escala de aplicaciones de una forma muy eficiente y rápida en cualquier entorno, de manera similar a una máquina virtual, Docker virtualiza el hardware del servidor. (AWS, n.d.)

2.2.2.1.Ventajas de Docker

Se pueden crear diferentes contenedores para aplicaciones y sus respectivas dependencias, lo que hace que sea 100% portable, otra ventaja importante es que su aislamiento es perfecto ya que cada aplicación se ejecuta en un contenedor

independiente con sus respectivas dependencias para funcionar correctamente, esto es muy importante para los desarrolladores ya que así pueden trabajar con la seguridad de que sus aplicaciones no van a interferir con otras ya que se encapsula el entorno de trabajo.

Además, Docker es muy ligero porque no virtualiza un sistema en su totalidad lo que hace que el consumo de recursos sea mínimo, todo esto hace que sea un entorno de pruebas seguro. (Blanch, 2024)

2.2.2.2.Arquitectura de Docker

Utiliza un tipo de arquitectura conocida como cliente-servidor que funciona de la siguiente manera, el cliente de Docker se comunica con el demonio de Docker, que es el encargado de crear, ejecutar y distribuir sus contenedores. El cliente y el demonio Docker se pueden ejecutar en el sistema o de forma remota y se comunican mediante una API REST mediante sockets UNIX o una interfaz de red. (dockerdocs, s.f.)

El cliente de Docker es la interfaz principal de usuario para Docker y es el que se comunica con el demo de Docker. Internamente este compuesto por tres componentes que son:

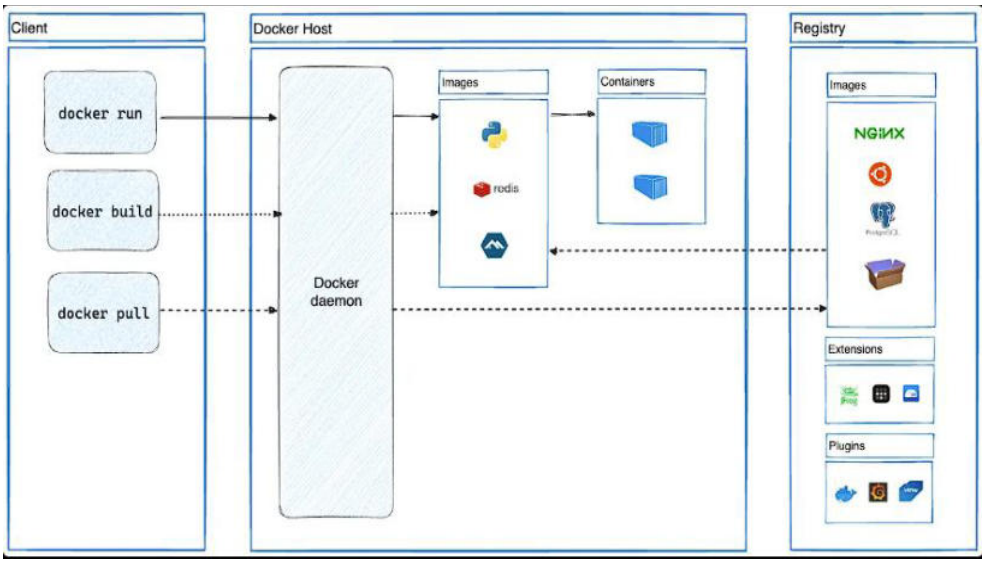
- **Imágenes de Docker**: son plantillas de lectura, esto quiere decir que, aunque una imagen contenga algún sistema operativo, Ubuntu, por ejemplo, solo nos permite crear contenedores en base a esa configuración.
- **Registros de Docker:** son el elemento de distribución de Docker y almacenan las imágenes, funcionan como una especie de repositorios que pueden ser públicos o privados en donde podemos descargar o subir imágenes.
- **Contenedores de Docker:** alojan todo lo indispensable para ejecutar

una aplicación, cada contenedor es una plataforma asilada creada de una imagen de Docker. (gitbooks, n.d.)

La arquitectura descrita anteriormente se puede evidenciar mejor en la Figura 1 que está a continuación:

Figura 1

Arquitectura de Docker



2.2.2.3.Almacenamiento, redes y registro de Docker

Entender las posibilidades disponibles para ejecutar contenedores Docker es muy importante si se quiere optimizar el consumo de recursos computacionales en un escenario en específico, los aspectos más importantes que se deberían tener en cuenta son los siguientes:

2.2.2.3.1. Almacenamiento

Hay que tener en cuenta que los contenedores no son permanentes y por lo tanto los datos que se encuentran en ellos tampoco lo son, a menos que se configure lo contrario. Las imágenes de Docker están formadas de varias capas de lectura, cuando se ejecuta un contenedor en base a una imagen, se da origen a una nueva capa de

escritura por encima de la capa superior de la imagen, la variación de contenido sobre la capa de escritura del contenedor no se almacena en ningún sitio, por lo que cuando un contenedor se elimina, también lo hacen los cambios realizados. También es importante saber cómo se almacenan las capas que forman las imágenes y los contenedores, para ello es importante hablar sobre los controladores de almacenamiento, que es un tema del cual se hablara más adelante. (Ltd, 2025)

Hay dos opciones cuando se trata de almacenar datos permanentemente:

2.2.2.3.1.1. Volúmenes

Este es el método recomendado para almacenar datos en contenedores Docker, ya que dicha plataforma se encarga de su gestión por lo que no depende de ningún sistema operativo y ofrece varias ventajas como que es fácil de respaldar y migrar, se comparte e forma segura entre contenedores y permiten almacenar datos en hosts remotos o en la nube. Además, los volúmenes se pueden crear antes o durante la creación del contenedor. (Ltd, 2025)

2.2.2.3.1.2. Montajes de unión

Están enlazados a la estructura de directorios y al sistema operativo, pero tiene un rendimiento parecido a los volúmenes. Estos montajes son una muy buena solución para cuando es necesario acceder a un archivo o directorio del sistema anfitrión y se pueden controlar a través de la CLI de Docker. (Ltd, 2025)

2.2.2.3.1.3. Controladores de almacenamiento

Se usan para almacenar capas de imágenes y datos en la capa de escritura del contenedor, dependiendo del controlador usado la velocidad de escritura tiende a ser menor al rendimiento del sistema de archivos nativo, por lo general el controlador predeterminado es el overlay2 y es el recomendado para su uso en sistemas de producción. (Ltd, 2025)

2.2.2.3.2. Redes

La conectividad de red hace referencia a la posibilidad de los contenedores para comunicarse entre sí. La red de Docker esta implementada de forma modular y se emplean varios controladores de red para su uso dependiendo de cada escenario, estos se detallan a continuación:

Puente: se usa para cuando se desea comunicar a los contenedores entre si dentro del mismo host, este es el controlador de red predeterminado.

Superposición: es utilizado para comunicar entre si a los contenedores gestionados por diferentes Docker, es decir en diferentes hosts.

Host: es utilizado para cuando no se desea aislar la red entre el contenedor y el host, por lo que el contenedor emplea la red del host.

IPvlan: proporciona un control completo para el direccionamiento IPv4 e IPv6.

Macvlan: permite asignar direcciones MAC a contendores para que se muestren como dispositivos fiscos en la red.

Ninguno: aísla por complemento a el contenedor del host. (Ltd, 2025)

2.2.2.3.3. Registro

Es importante realizar una monitorización del sistema, Docker dispone de un subsistema de registro configurable, existen varios controladores que reenvían los registros de un contenedor a un archivo, base de datos o cualquier otro método de registro. Los registros incluyen primordialmente todo lo que se escribe en los dispositivos de entrada/salida. Básicamente un registro es un servicio que ofrece Docker para el almacenamiento y distribución para imágenes de contenedores. (Ltd, 2025)

2.2.3. Seguridad en entornos Docker

Dentro de entornos Docker es muy importante poder evaluar las amenazas y

vulnerabilidades que se puedan presentar, por lo que es necesario una gestión adecuada de la seguridad ya que una mala configuración puede desembocar en una vulnerabilidad critica. Para gestionar amenazas se usan marcos de referencia para evaluar, priorizar y mitigar riesgos sistemáticamente.

2.2.3.1.CIS Docker Benchmarks

Es un conjunto de directrices de configuración de seguridad desarrollado específicamente para una tecnología, los Benchmarks son importantes para una organización porque mejoran la capacidad de prevenir, detectar y responder ante una ciber amenaza. Los estándares CIS (Centro para la Seguridad de Internet) simbolizan las buenas prácticas de seguridad dentro de contenedores, mismas que se actualizan constantemente adaptándose a la evolución de las ciber amenazas. Los estándares CIS incluyen recomendaciones específicas para varias plataformas, abarcando los sistemas operativos y servicios de nube. (Liang, 2024)

Las directrices que provee CIS a Docker abarcan varios aspectos importantes entre los cuales se puede destacar la configuración del demonio Docker, las imágenes de contenedores y la seguridad en el entorno de ejecución de Docker, cumplir con estas directrices ayuda a que las organizaciones mitiguen riesgos como el acceso no autorizado, escalada de privilegios e incluso las filtraciones de datos. (wazuh, 2024)

2.2.4. Amenazas en entornos Docker

La forma en la que las organizaciones despliegan y escalan aplicaciones se ha transformado gracias a los entornos Docker, pero esto conlleva al mismo tiempo la introducción de nuevas superficies de ataque. Si bien es cierto la contenedorización brinda aislamiento y eficiencia, también puede estar expuesto a amenazas como configuraciones inseguras, imágenes vulnerables y accesos no controlados por lo que comprender las amenazas que conlleva el uso de Docker es primordial para desarrollar

controles que aseguren la integridad, disponibilidad y confidencialidad de los servicios.

2.2.4.1.Imágenes base antiguas y vulnerables

Las imágenes de Docker comúnmente dependen de imágenes base (Ubuntu o CentOS), si dichas imágenes no se mantienen actualizadas, puede generar puntos débiles conocidos, por lo que es importante mantener supervisadas y actualizadas las imágenes base incorporando parches de seguridad reduciendo así el riesgo de explotación. (SentinelOne, 2025)

2.2.4.2.Dependencias de terceros inseguros

Las imágenes de Docker incluyen bibliotecas y dependencias de terceros que pueden llegar a afectar la seguridad de los contenedores por lo que es de vital importancia mantener dichas dependencias actualizadas además de evaluar y minimizar las bibliotecas de terceros pueden reducir significativamente las vulnerabilidades potenciales. Para enfrentar estas amenazas es importante utilizar herramientas para el análisis de vulnerabilidades ya que ayudan a identificar y evaluar amenazas en bibliotecas de terceros. (SentinelOne, 2025)

2.2.4.3.Opciones de red mal configurados

Las imágenes de Docker pueden abrir puertos no deseados o incluso utilizar protocolos de red que no son seguros y estos pueden ser utilizados para acceder a los contenedores o movilizarse dentro de la red, para evitar esta mala configuración es importante tener una buena práctica en cuanto a seguridad de red se refiere entre las cuales se pueden destacar los cortafuegos, los espacios de nombre de red y el bloqueo de puestos expuestos. (SentinelOne, 2025)

2.2.4.4.Escape de contenedor

Es la capacidad que tiene las aplicaciones que se ejecutan dentro de un

contenedor para tener acceso a recursos externos que deberían estar bloqueados, mediante esta técnica los ciberdelincuentes pueden extraer datos confidenciales o incluso instalar softwares maliciosos, por lo que los contenedores no deben estar configurados para tener acceso al sistema de archivos del servidor que lo contiene. El escape de contenedor puede ser una vulnerabilidad muy problemática ya que los atacantes pueden comprometer todas las aplicaciones y recursos de un servidor llegando incluso a extenderse a otros servidores. Esta vulnerabilidad puede presentarse debido a configuraciones inseguras que permiten que un contenedor tengo acceso a recursos del servidor que no estaba previsto que accediera o que el atacante explote una vulnerabilidad en el entorno de ejecución del contenedor, lo que les permite eludir los controles de acceso. (Sheps, 2024)

2.2.5. Controles de seguridad en Docker

El objetivo de usar ambientes dockerizados, es el de optimizar el uso de los recursos del equipo Host, ya que se pueden montar imágenes Docker de menor tamaño, que utilizan recursos mínimos, siento estos modificables según sea la necesidad, teniendo como resultado el funcionamiento correcto de los servicios que contienen. Estas imágenes pueden ser usadas como imágenes que posee sistemas operativos completos para su uso o microservicios asignados como manejos de bases de datos (SQL, MySQL, MongoDB, etc.), servidores WEB (Apache, Nginx, etc.), máquinas atacantes (Kali Linux), entre otros.

Debido a su uso, es importante poseer controles de seguridad en ambientes dockerizados, con controles en distintos niveles de su arquitectura, ya sea en las imágenes, en los contenedores, seguridad a nivel del red y control de seguridad en el equipo principal o Host.

2.2.5.1.Controles sobre imágenes Docker

La imagen de Docker viene a ser el conjunto de instrucciones cargadas a modo de plantilla que se ejecutan dentro de un contenedor, estas plantillas pueden ser reutilizadas con las configuraciones e instrucciones establecidas. Se puede ver a las imágenes como el sistema operativo donde se van a ejecutar los servicios, estos poseen sus librerías, versión de kernel, las dependencias y ajustes.

Sabiendo esto, es importante que se tenga el control adecuado en las imágenes para que estas se encuentre actualizadas, que no posean paquetes innecesarios, si hay puertos abiertos, tengan su control de acceso necesarios para evitar los backdoors, evitar el código malicioso, que generalmente sucede por aplicaciones descargadas de repositorios no oficiales, entre otros. “Use minimal base images and avoid installing unnecessary packages to reduce the attack surface” (Docker Inc., 2025, section Image Security).

Controles sugeridos

- Verificar que las imágenes que se utilizan sean de fuentes confiables u oficiales, eso permite respaldar la seguridad y un correcto funcionamiento.
- Instalación limpia y mínima de una imagen, no instalar imágenes con paquetes ni repositorios que no se van a utilizar.
- Usar versiones estables o anteriores a las denominadas “latest”, estas son las versiones más recientes y pueden traer vulnerabilidades que aún no hayan sido controladas o parcheadas.
- Realizar escaneo de vulnerabilidades, en caso de encontrarlas, tomar las medidas necesarias de protección.
- Existe la herramienta Docker Content Trust, esta sirve para validar las imágenes, su autenticidad es importante para la seguridad y las imágenes que puedan ejecutarse en producción deben estar firmadas.

- Evitar el uso o almacenamiento de información sensible en la imagen como contraseñas, tokens, usuarios, entre otros.

2.2.5.2.Controles sobre contenedores Docker

Un contenedor de Docker es el software que contiene todo lo que se necesita para ejecutar la aplicación, en esto se puede recalcar que contiene el código, las distintas dependencias, bibliotecas y también las herramientas que necesita el sistema.

Controles sugeridos

- Es recomendable ejecutar los contenedores con usuarios que no tengan todos los privilegios, es decir que no sean root, esto se debe definir en el Dockerfile. “Never run containers as the root user. Drop all unnecessary Linux capabilities to limit what the container can do if compromised” (Better Stack, 2025).
- El usuario que ejecuta el contenedor debe tener los permisos justos y los necesarios, ya que permisos con más altos privilegios pueden recaer en una vulnerabilidad.
- Evitar que se pueda escribir archivos en el contenedor, es decir que sean archivos solamente con permisos de lectura para el usuario que lo ejecuta.
- Limitar los recursos, como se mencionaba antes, es importante asignar los recursos necesarios nada más, para no tener un desperdicio de procesamiento ni memoria.

2.2.5.3.Controles en la Red

Los contenedores suelen subirse en una red general o individual, esto depende del uso y la aplicación que se tiene, lo que se debe tener en cuenta que, si un contenedor se encuentra comprometido y existen más contenedores en esta red, facilita el desplazamiento del atacante a los demás contenedores. Además, es importante

mantener un control de los puertos necesarios para su funcionamiento y limitar las conexiones por puerto o por IP. “Limit container-to-container communication using network segmentation and avoid exposing ports unnecessarily” (Wiz.io, 2025).

Controles sugeridos

- Segmentar la red y ocupar las IPs para hosts lo más limitadas posibles y tener configurado que contenedores pueden comunicarse entre sí.
- Mantener deshabilitados los puertos innecesarios.
- Evitar que el Daemon Docker se vea expuesto, ya que, si es vulnerado o expuesto, puede facilitar la manipulación del Host.
- Utilizar un firewall y definir las reglas de filtrado para la entrada, salida y redireccionamiento del tráfico.
- Evitar el acceso a los contenedores desde redes externas, es recomendable que sean manipulados en la red local para evitar el robo de credenciales.

2.2.5.4. Controles en el Host

En esto intervienen distintos puntos como la seguridad del Host, ya sea un sistema que se encuentre virtualizado o una máquina física que contenga el sistema operativo donde se despliegan los contenedores, también es importante la seguridad y estabilidad del kernel. Hay que tener en cuenta que los contenedores poseen el mismo kernel del Host, lo que puede desencadenar que una vulnerabilidad presente, provocaría que obtengan el control completo del anfitrión. “The host Operating system must be hardened and kept up to date, as container security is directly dependent on the security of the underlying host” (Center for Internet Security, 2025, p. 10).

Controles sugeridos

- Se recomienda mantener actualizado el sistema anfitrión y el de las imágenes del Docker.

- Limitar la instalación de librerías, dependencias o apertura de puertos innecesarios en el Host.
- Limitar las conexiones al sistema, puede ser por MAC, por dirección IP o por puerto como se había mencionado.
- Limitar el acceso al Daemon Docker.
- Evitar el almacenamiento de información sensible dentro del Host.

2.2.6. Establecer monitoreo y auditoría de los ambientes dockerizados.

Se debe tener un control y monitoreo de los ambientes dockerizados, más aún cuando están en producción y se tiene conexiones en tiempo real, si las conexiones a nuestros microservicios también son accedidos de forma pública, se debe tener mucho más cuidado, por lo que es recomendable la generación de logs de conexión, el tiempo de ejecución de los contenedores y validar las anomalías que se presenten, esto se puede implementar con sistemas de monitoreo o configurando tareas de monitoreo en el sistema anfitrión. “Continuous monitoring and vulnerability scanning are required to detect abnormal runtime behavior and newly exposed risks” (Sonatype, 2025).

Controles sugeridos

- Agregar logs para los contenedores con el objetivo de controlar los inicios de sesión, cambios, actualizaciones, entre otros.
- Montar sistemas o herramientas de monitoreo, se recomienda el uso de Falco, que sirve para monitorear sistemas en el runtime (tiempo de ejecución).
- Mantener una auditoría periódica de los ambientes dockerizados.
- Verificación constante de las imágenes y actualizaciones que tienen, es recomendable hacer escaneos de la integridad, de la versión, realizar test

de rendimient (CIS Docker Benchmark) y validar el rendimiento en el Host y los contenedores Docker.

2.2.7. Evaluación de los Controles de Seguridad en Docker

La adopción de contenedores Docker requiere un enfoque integral de seguridad que abarque imágenes, contenedores, red, host y monitoreo. Diversas guías y autores coinciden en que la protección debe implementarse en múltiples capas para reducir la superficie de ataque y prevenir incidentes (Anchore, 2025; Better Stack, 2025; Sonatype, 2025).

2.2.7.1.Controles sobre las imágenes Docker:

Las imágenes son el componente base sobre el cual se ejecutan los contenedores. La literatura técnica enfatiza que estas deben ser tratadas como artefactos críticos de software (Anchore, 2025; Sonatype, 2025). Entre los controles evaluados destacan:

- **Procedencia y repositorios confiables:** Se recomienda utilizar únicamente imágenes provenientes de repositorios oficiales, firmas verificadas o registros privados. La descarga de imágenes de fuentes desconocidas incrementa el riesgo de backdoors o malware (CyberPanel, 2024).
- **Escaneo de vulnerabilidades:** Herramientas como Trivy, Anchore Engine o Clair deben emplearse de forma continua para identificar dependencias vulnerables. Las guías señalan que el escaneo debe realizarse tanto al momento de crear la imagen como cuando se despliega (Anchore, 2025; Wiz.io, 2025).
- **Imágenes minimalistas:** El uso de imágenes livianas como Alpine reduce la superficie de ataque. SUSE (2024) enfatiza que cuanto menos

paquete incluya una imagen, menor será la probabilidad de que contenga vulnerabilidades.

- **Firma e integridad de imágenes:** El uso de Notary o Docker Content Trust permite verificar la autenticidad de una imagen antes de ejecutarla, evitando manipulaciones o sustituciones (Sonatype, 2025).

2.2.7.2. Controles sobre los contenedores Docker

Las guías modernas coinciden en que la ejecución segura del contenedor depende de configuraciones estrictas, evitando privilegios excesivos y limitando los recursos (Better Stack, 2025; OPS Moon, 2025).

- **Uso de usuarios no privilegiados:** Ejecutar contenedores como root representa uno de los mayores riesgos. Las recomendaciones señalan siempre utilizar usuarios no privilegiados definidos en la imagen o mediante configuraciones en el runtime (OPS Moon, 2025).
- **Restricciones de recursos:** Para evitar ataques de denegación de servicio, es obligatorio definir límites de CPU, memoria y I/O. Mejor Stack (2025) destaca que estos límites deben ser parte de la política estándar para todo contenedor.
- **Mecanismos de aislamiento:** Docker emplea namespaces, cgroups, seccomp, AppArmor y SELinux como mecanismos de control de acceso. SUSE (2024) menciona que la combinación de estos controles permite un aislamiento adecuado entre procesos.
- **Evitar binds y accesos innecesarios al host:** Montajes directos al sistema de archivos del host pueden comprometer su seguridad. CyberPanel (2024) señala que todo bind mount debe ser estrictamente justificado y restringido.

2.2.7.3. Controles en la red

La seguridad de red en Docker es uno de los aspectos más críticos debido a la comunicación entre microservicios.

- **Segmentación de redes:** La práctica recomendada es crear redes específicas para cada aplicación o dominio de servicio, limitando así la visibilidad y alcance de los contenedores (Wiz.io, 2025).
- **Políticas de tráfico restringido:** Debe aplicarse una política zero trust donde únicamente se permita el tráfico explícitamente autorizado. Cloud Native Now (2025) señala que esto evita movimientos laterales en caso de compromiso.
- **Encriptación del tráfico:** Cuando los contenedores se comunican entre hosts, se sugiere emplear redes overlay que soporten cifrado, minimizando riesgos de intercepción (SUSE, 2024).

2.2.7.4. Controles en el host

Si el host es comprometido, todos los contenedores que aloja también lo estarán. Por ello, las recomendaciones se centran en proteger tanto el sistema operativo como el daemon Docker.

- **Actualización del sistema operativo y kernel:** Los autores coinciden en que un kernel desactualizado puede exponer vulnerabilidades explotables desde los contenedores (Anchore, 2025).
- **Protección del Docker Daemon:** El acceso al socket /var/run/docker.sock otorga control total sobre el host. Sonatype (2025) recomienda restringir su acceso únicamente a usuarios autorizados.
- **Minimización de servicios en el host:** Mientras menos servicios se ejecuten, menor será la superficie de ataque. Better Stack (2025)

sugiere deshabilitar todo componente innecesario.

2.2.7.5. Monitoreo y auditoría de ambientes dockerizados

- **Centralización de logs:** Los contenedores deben enviar sus registros a plataformas SIEM o soluciones centralizadas. Esto facilita la trazabilidad, detección de patrones y análisis forense (Cloud Native Now, 2025).
- **Alertas de seguridad:** Debe configurarse la detección de eventos relevantes como contenedores que se reinician inesperadamente, cambios en imágenes o actividad de red sospechosa (OPS Moon, 2025).
- **Auditorías periódicas:** Wiz.io (2025) resalta que las auditorías permiten detectar configuraciones inseguras, vulnerabilidades sin corregir y desviaciones respecto a las políticas establecidas.

CAPITULO 3:

3. DESARROLLO

3.1. Desarrollo del Trabajo

3.1.1. Configuración de la maquina Vulnerable

Figura 2

Máquina Vulnerable: Ubuntu Server 22.04.5 LST.

Figura 3

Selección de la imagen ISO y nombre de la máquina virtual.

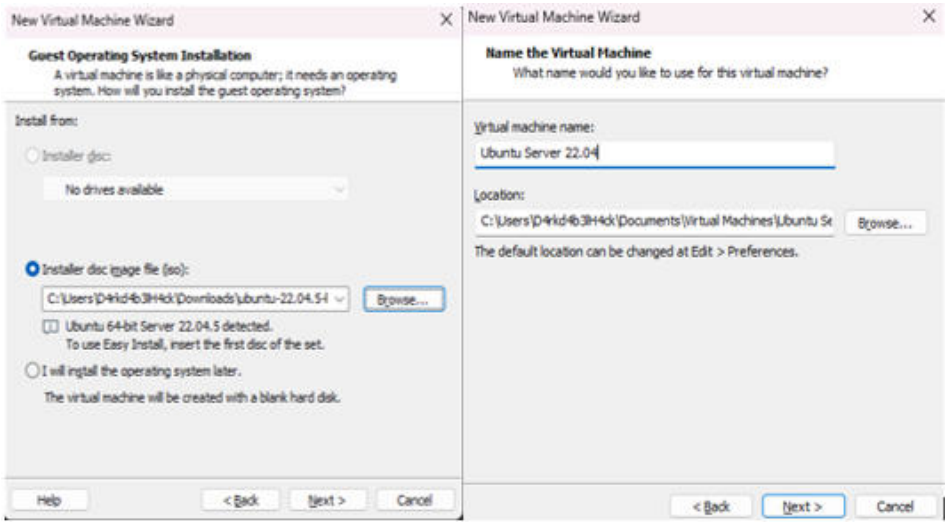


Figura 4

Asignar el espacio en disco de la máquina virtual atacante.

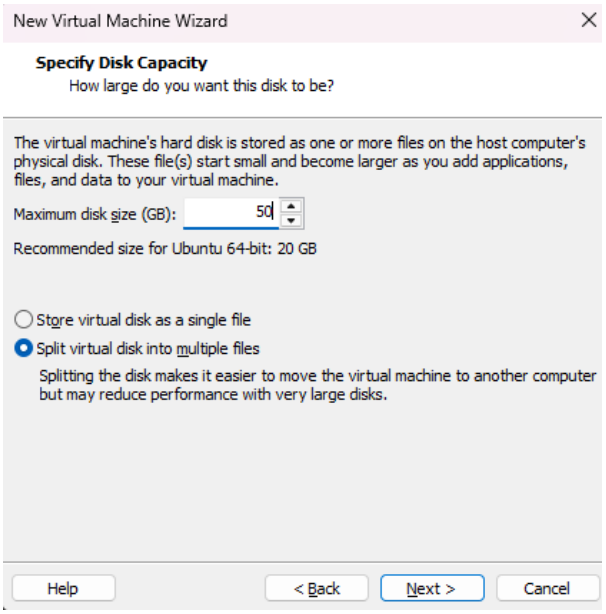
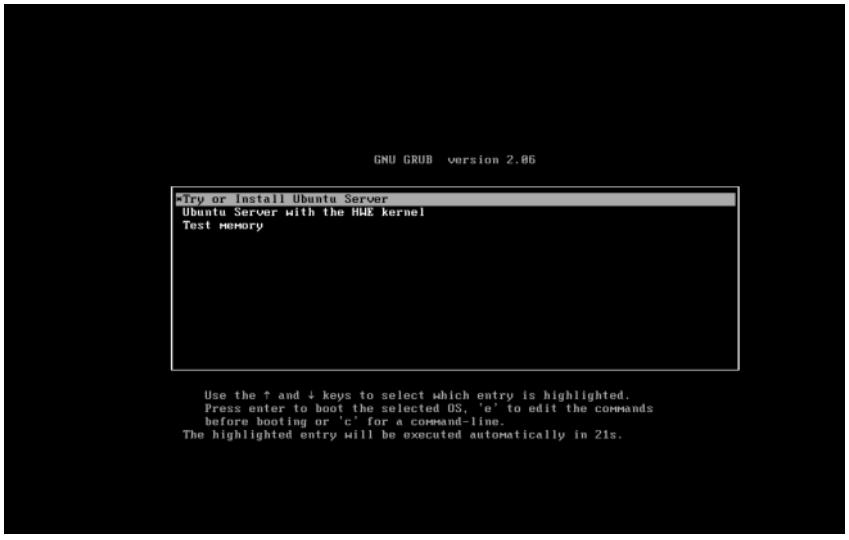


Figura 5

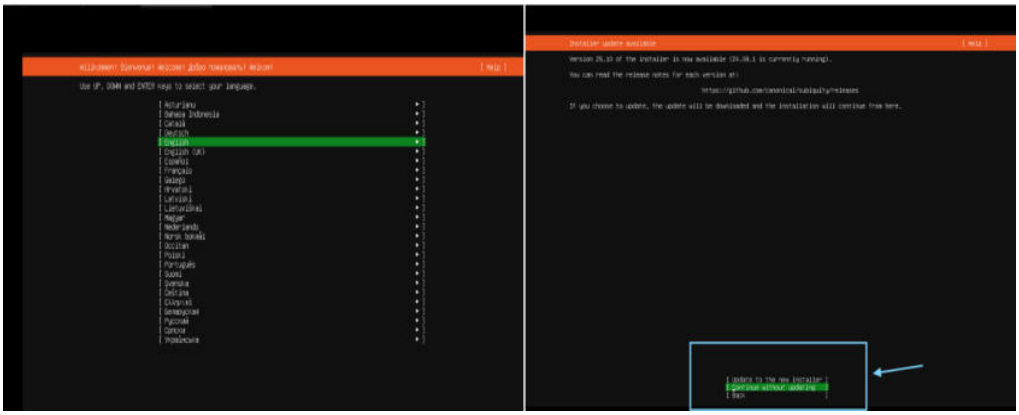
Inicio de la instalación de Ubuntu Server desde ISO.



Nota: Dar clic en siguiente e instalar la imagen .ISO seleccionada.

Figura 6

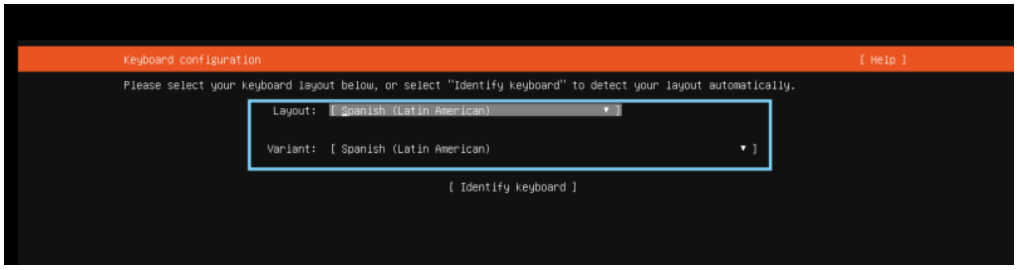
Selección de idioma del sistema.



Nota: Seleccionar el idioma y Continuar sin actualizar.

Figura 7

Configuración del idioma del teclado.



Nota: Seleccionar el idioma de escritura (teclado del sistema).

Figura 8

Inicio de la instalación de Ubuntu Server

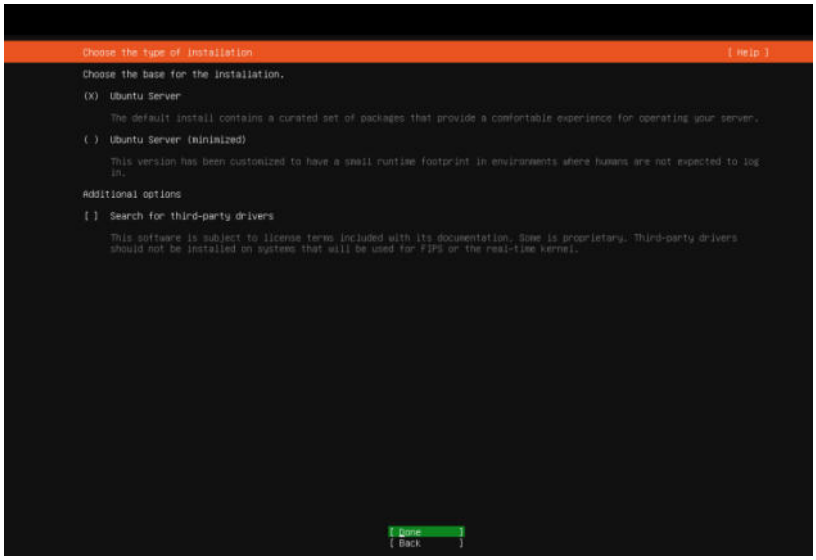


Figura 9

Configuración de la interfaz de red.

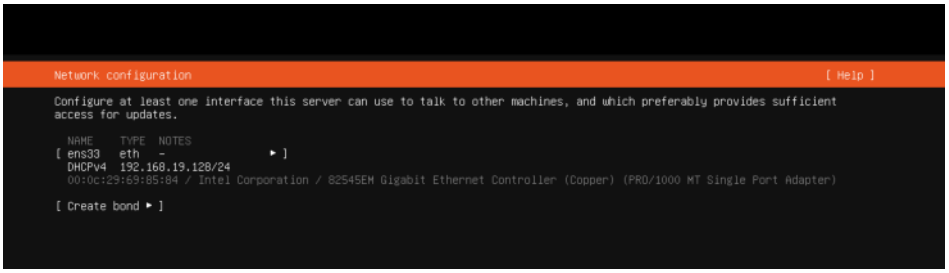


Figura 10

Configuración del proxy del sistema.

Figura 11

Uso completo del disco asignado.

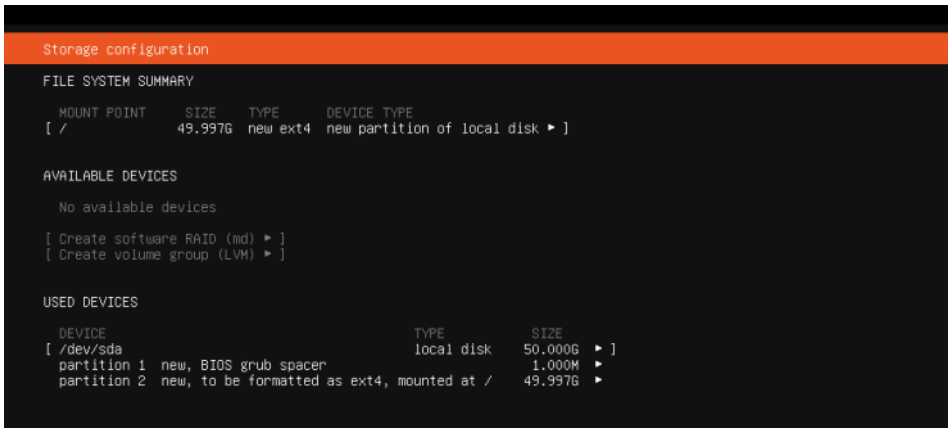


Figura 12

Configuración de nombre del equipo, usuario y contraseña.

Nota: Configurar el nombre del equipo(srv_grupo3) - usuario (grupo3) y contraseña (.grupo3.)

Figura 13

Instalar el servicio de SSH.

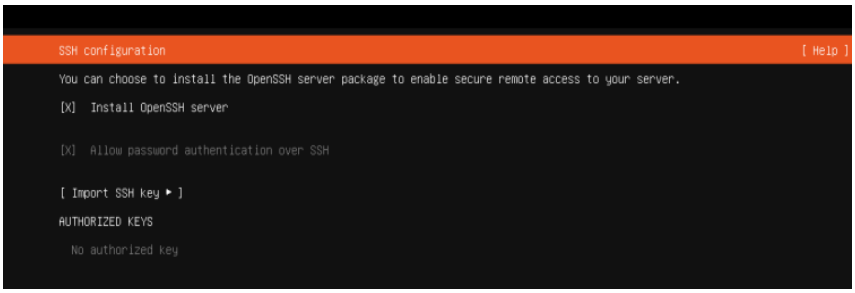


Figura 14

Proceso de instalación del sistema operativo.

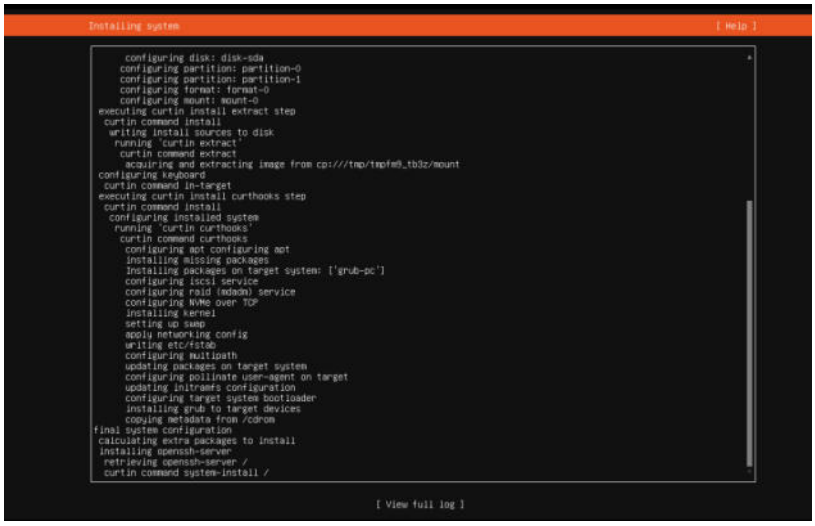


Figura 15

Finalización de la instalación y reinicio del sistema.

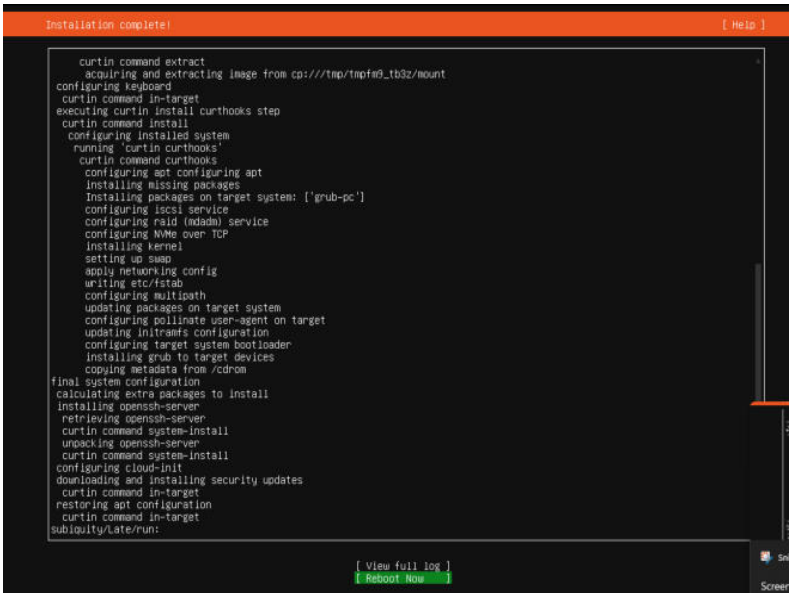


Figura 16

Inicio de sesión y actualización de repositorios.

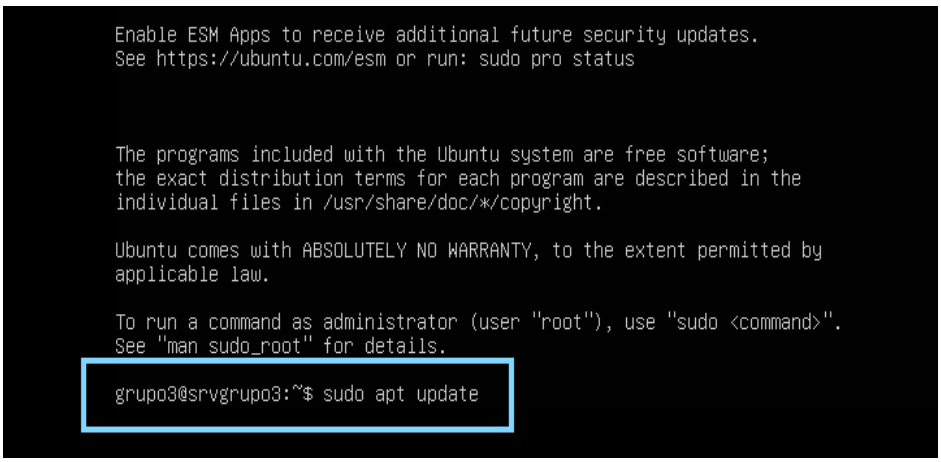


Figura 17

Actualización de librerías y aplicaciones del sistema.

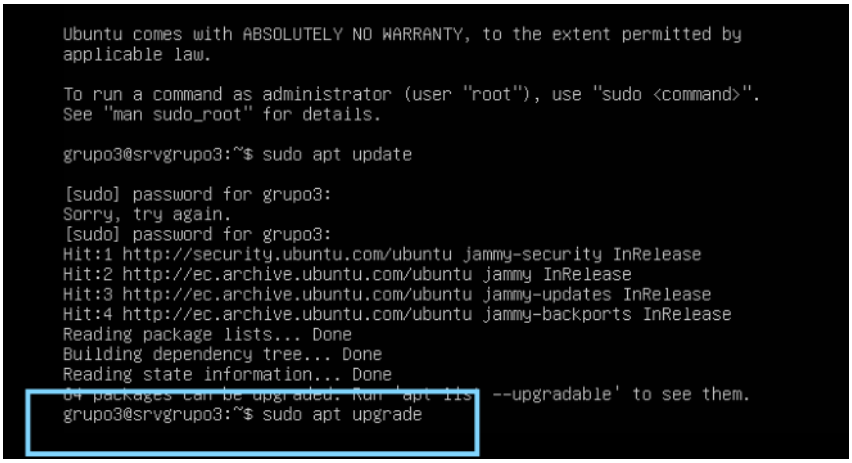


Figura 18

Proceso de actualización del sistema.

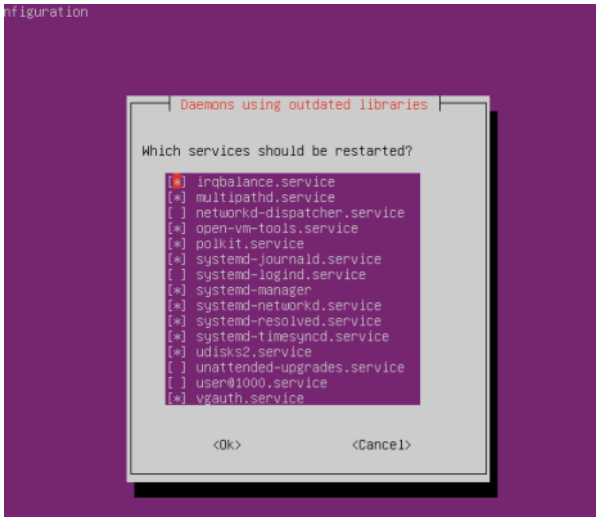
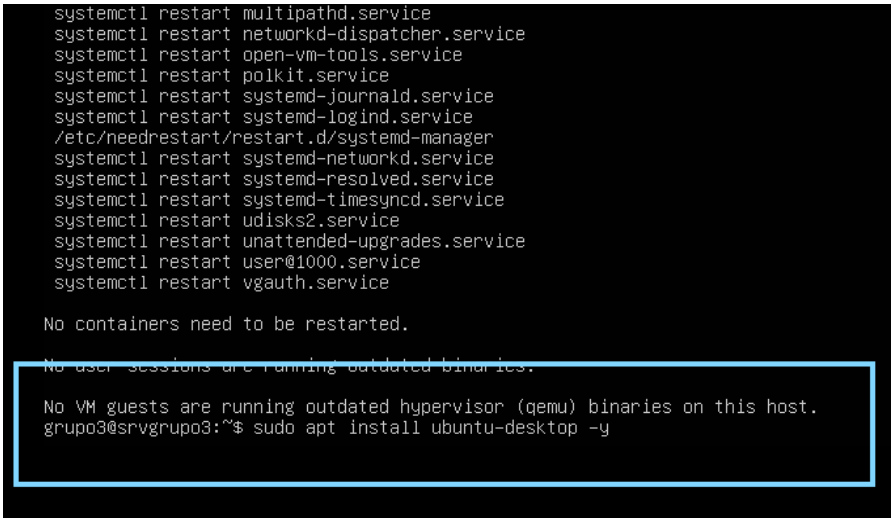


Figura 19

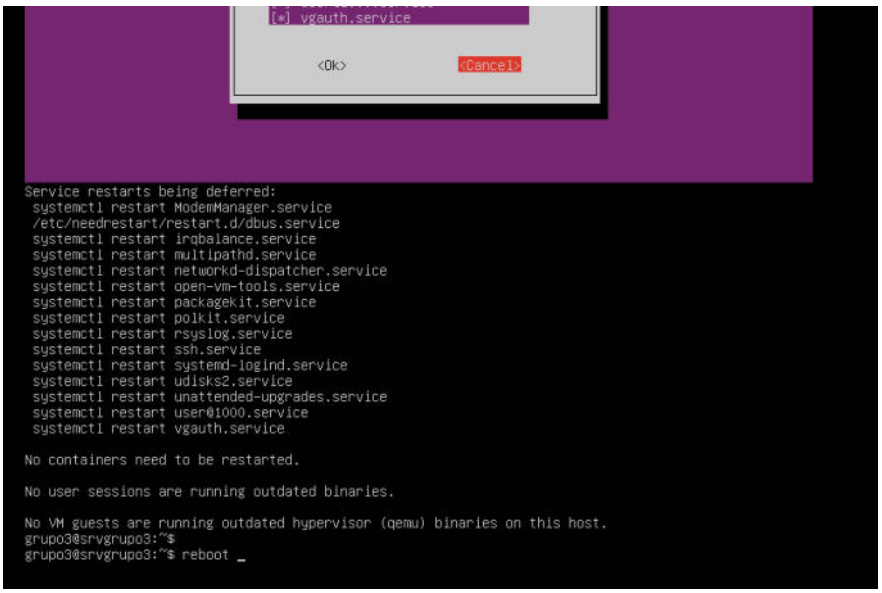
Instalación del entorno gráfico en Ubuntu Server.



Nota: Instalar interfaz gráfica para una mejor manipulación del servidor y los contenedores que se van a utilizar.

Figura 20

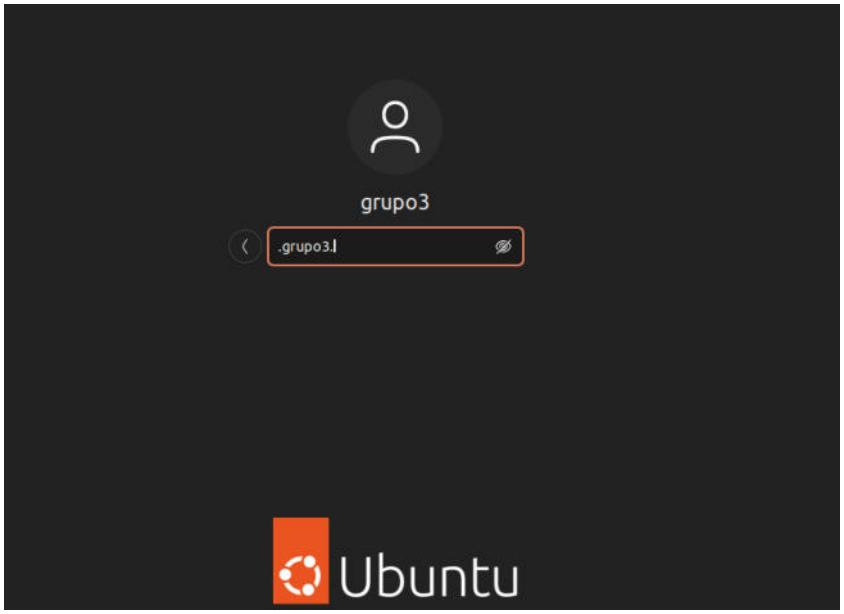
Reinicio del sistema para activar el entorno gráfico.



Nota: Es necesario enviar a reiniciar para que se active el entorno gráfico en el servidor.

Figura 21

Inicio de sesión posterior a la instalación gráfica.



Nota: Una vez finalizado el reinicio, procedemos a Iniciar sesión al servidor.

3.1.2. Instalación y configuración de la máquina atacante

Una vez que tenemos descargada la máquina virtual oficial del sitio de Kali, en VMware Workstation player dar clic en Open Virtual Machine, dirigirse al Path en el que se ha descomprimido la VM de Kali Linux y seleccionamos el archivo VMX que este contenido dentro.

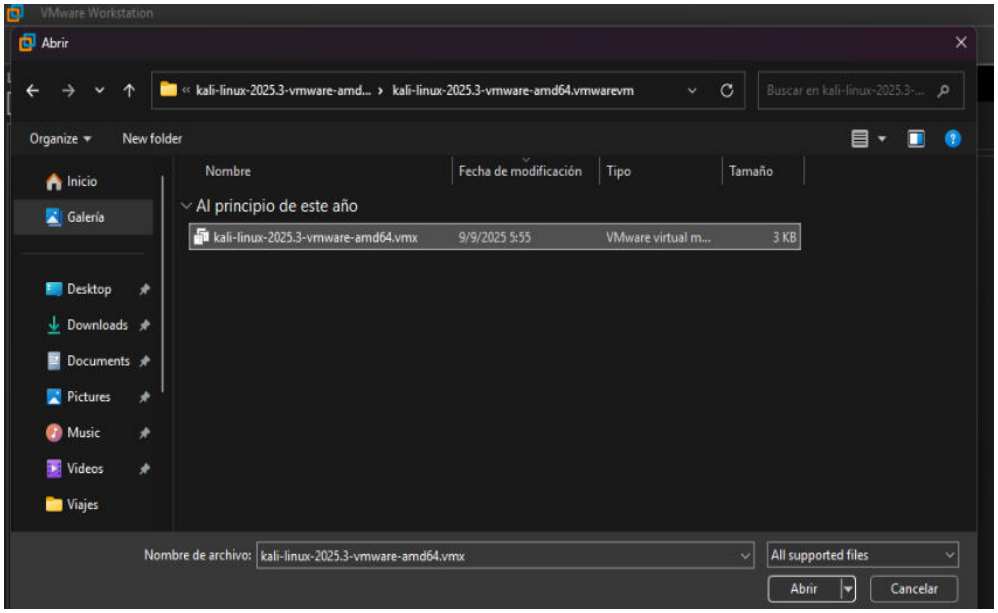
Figura 22

Importación de la máquina virtual Kali Linux.

Nota: Instalación de imagen .OVA descargada desde la página oficial de Kali Linux.

Figura 23

Extracción e inicio de la máquina virtual Kali Linux.



Nota: Iniciar la VM, ingresar con las credenciales (kali) para el usuario y contraseña.

Figura 24

Inicio de sesión en Kali Linux.

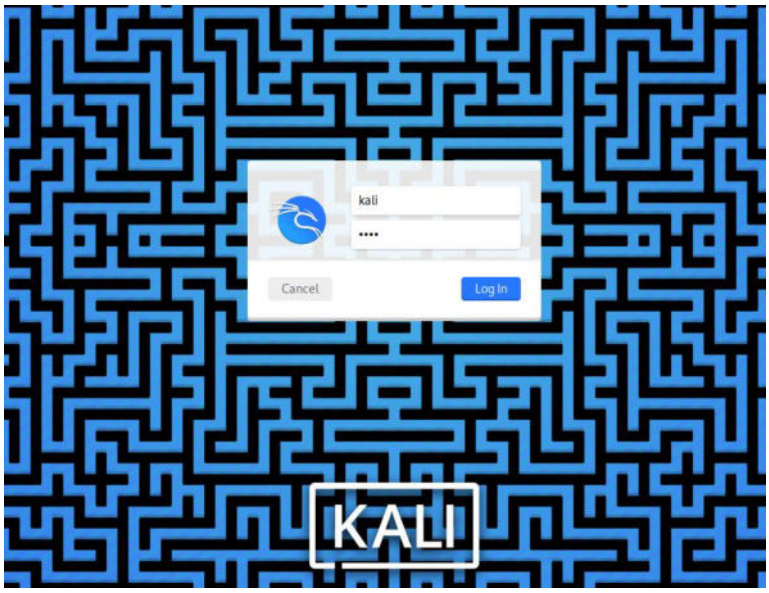
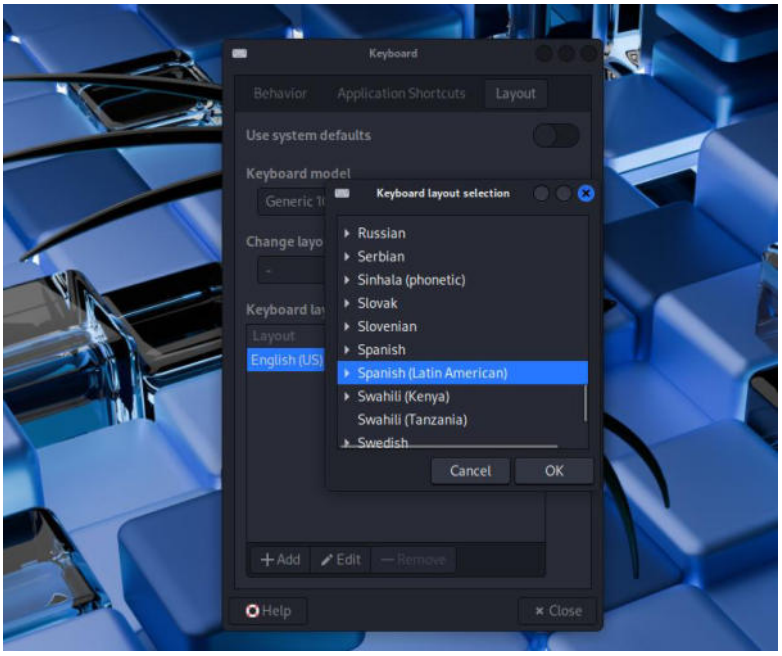


Figura 25

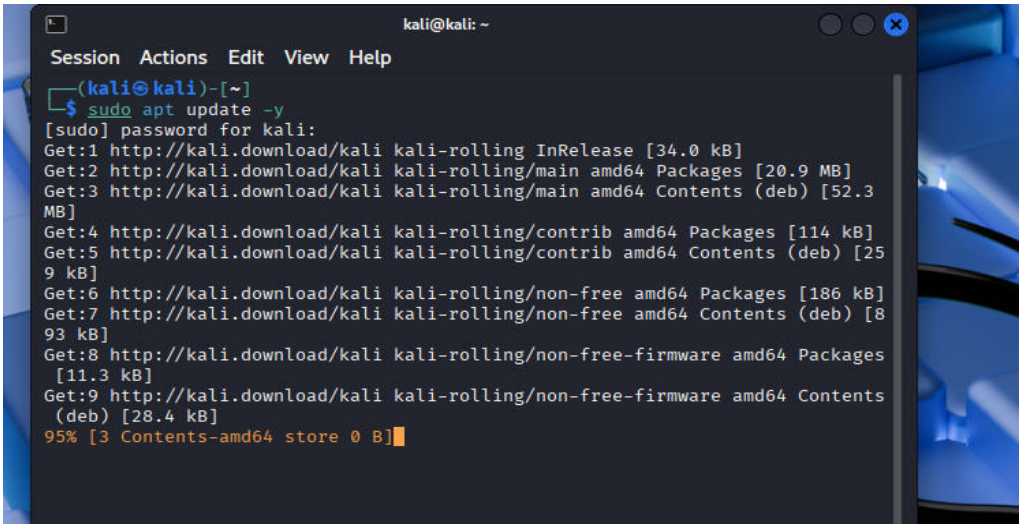
Configuración del idioma del teclado en Kali Linux.



Nota: Como recomendación, se debe cambiar el idioma del teclado del Kali Linux.

Figura 26

Actualización de repositorios en Kali Linux.



Nota: Actualizar los repositorios mediante la terminal sudo apt update -y.

3.1.3. Instalación de Nessus en Kali Linux

Figura 27

Descarga del instalador de Nessus.

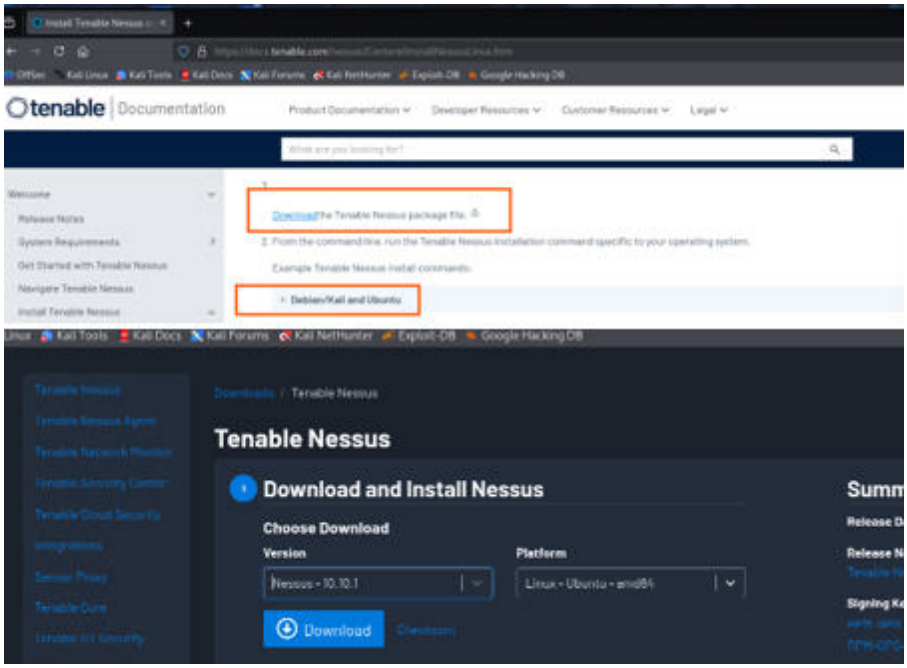


Figura 28

Instalación del paquete Nessus en Kali Linux.

Figura 29

Inicio del servicio Nessus.

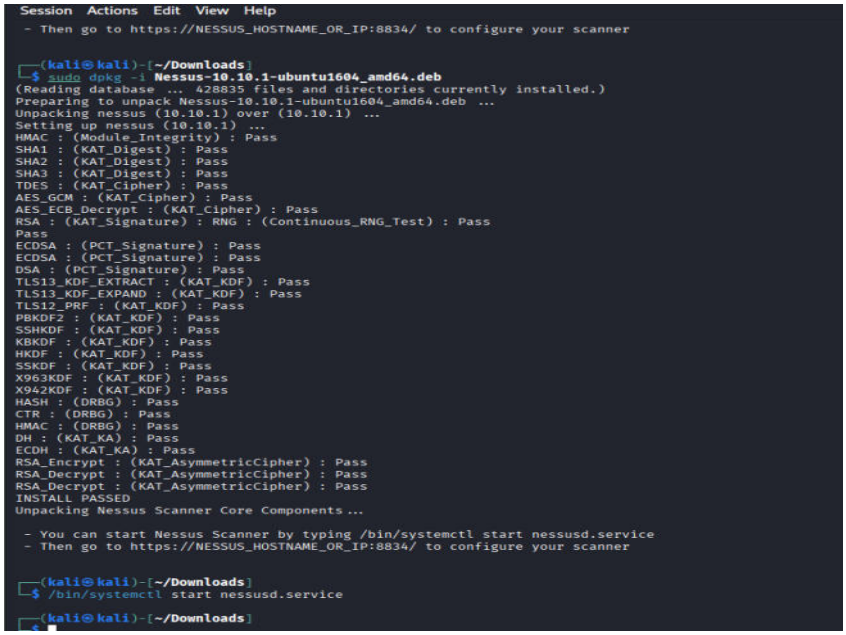
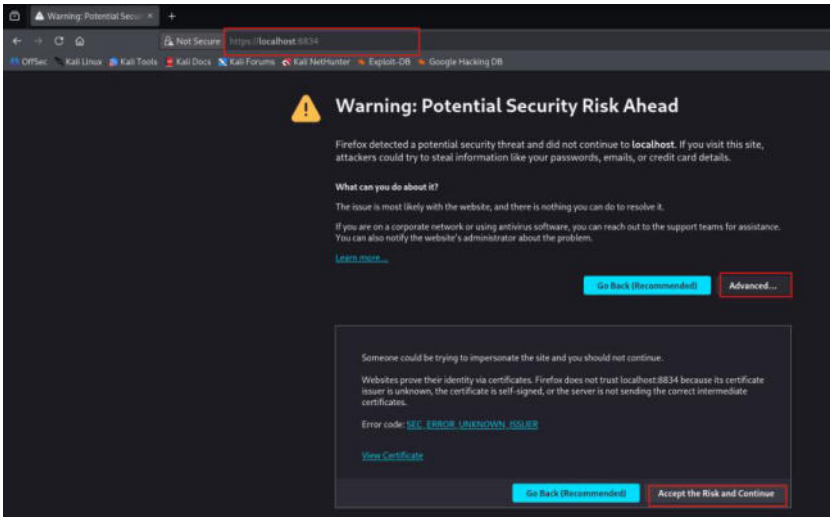


Figura 30

Acceso a la consola web de Nessus.



Nota: Ingresar por el navegador con la dirección https://localhost:8834.

Figura 31

Activación de Nessus Essentials.

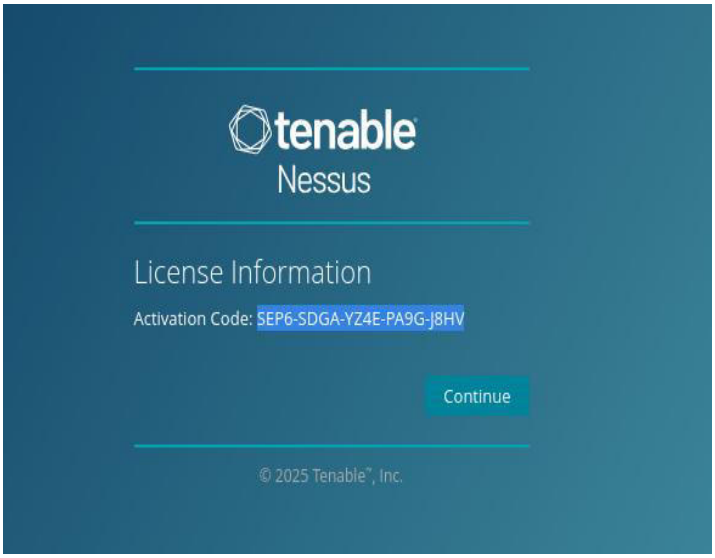


Figura 32

Proceso de inicio de sesión de Nessus Essentials.

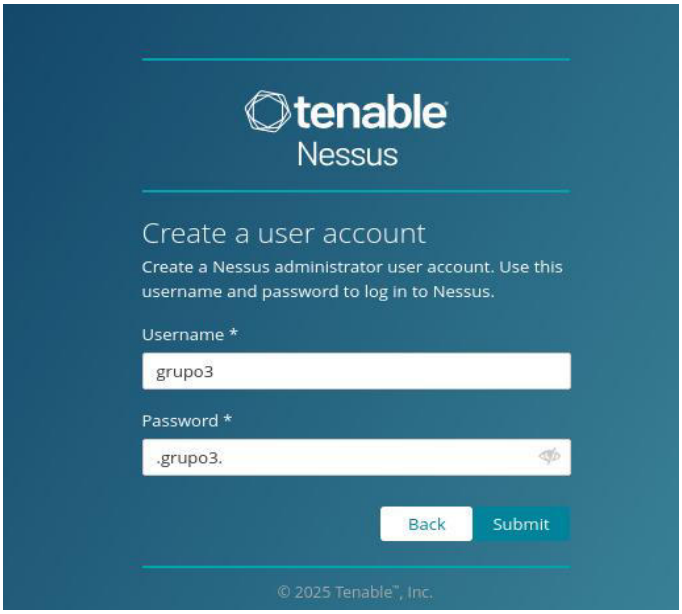


Figura 33

Descarga de plugins de seguridad.

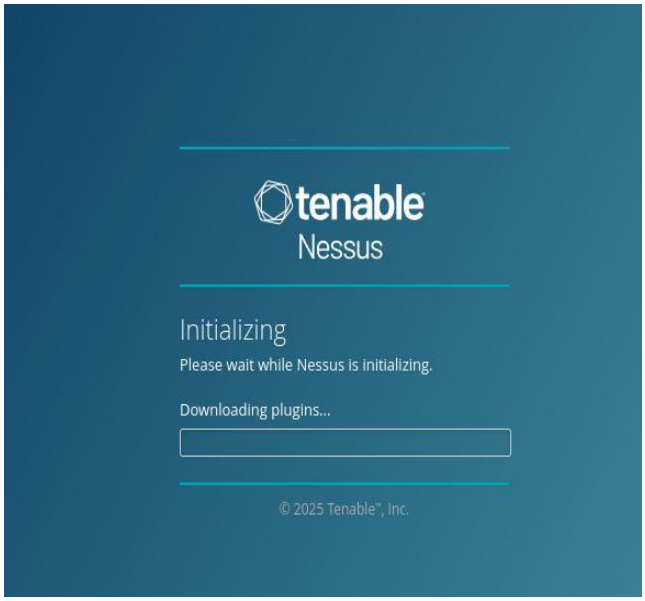
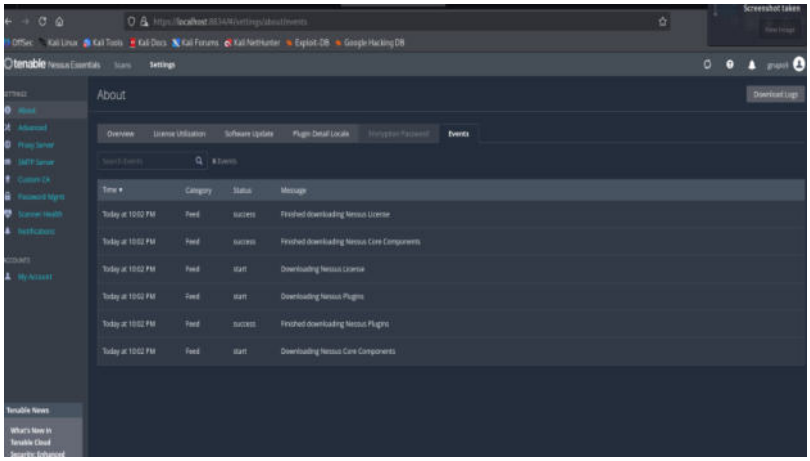


Figura 34

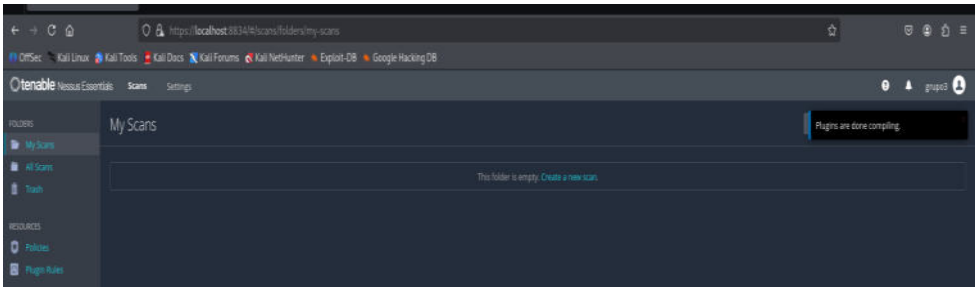
Verificación de componentes de Nessus.



Nota: Una vez inicializado, veremos la consola, sin embargo; debemos esperar a la finalización del proceso de compilación de complementos.

Figura 35

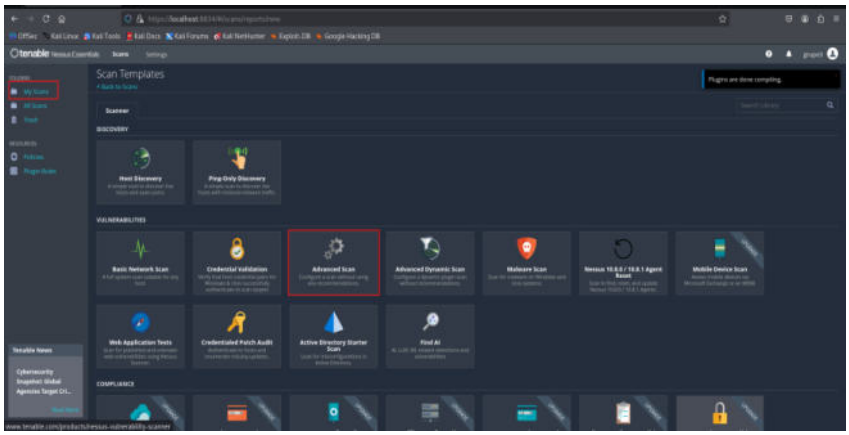
Ejecución de escaneos de seguridad.



Nota: Ejecutar scans, una vez finalizado este proceso, tendremos acceso a la herramienta y podremos ejecutar scans, es decir, análisis de seguridad, en nuestro caso, al servidor que instalamos anteriormente.

Figura 36

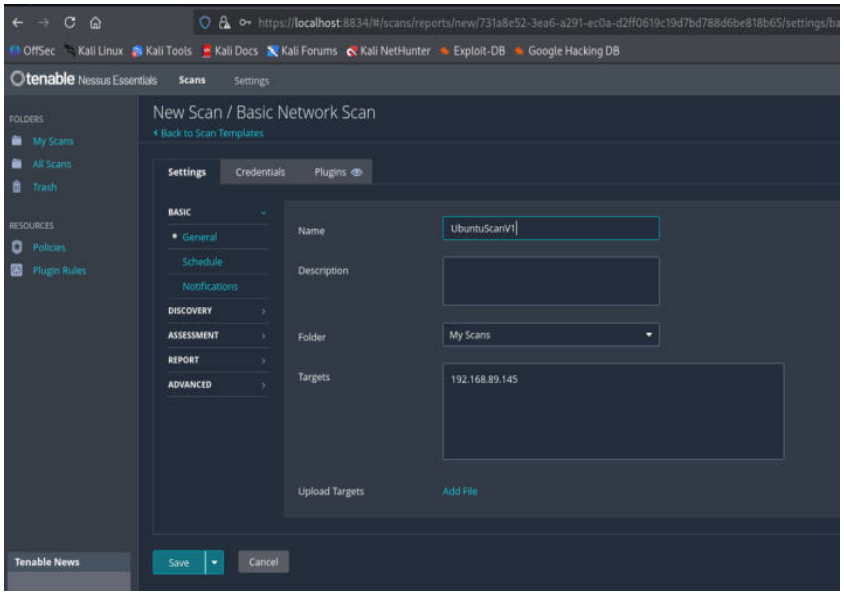
Creación de un nuevo análisis de red.



Nota: Damos clic en crear un nuevo scan, luego seleccionamos la opción "Basic Network Scan".

Figura 37

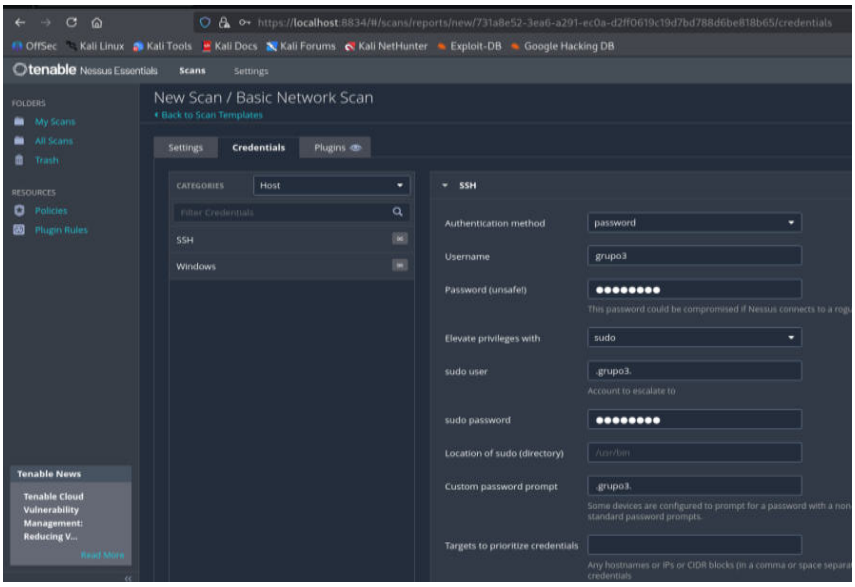
Configuración del objetivo del escaneo.



Nota: Ingresamos la información del servidor a analizar.

Figura 38

Configuración de credenciales para escaneo autenticado



Nota: En nuestro caso configuraremos las credenciales del servidor, esto permite que Nessus haga un Credentialed Scan, que es mucho más completo.

Configuramos el apartado Credentials de la siguiente manera:

Authentication method: *Password*

Username*: (tu usuario SSH del servidor)*

Password: *la contraseña correspondiente*

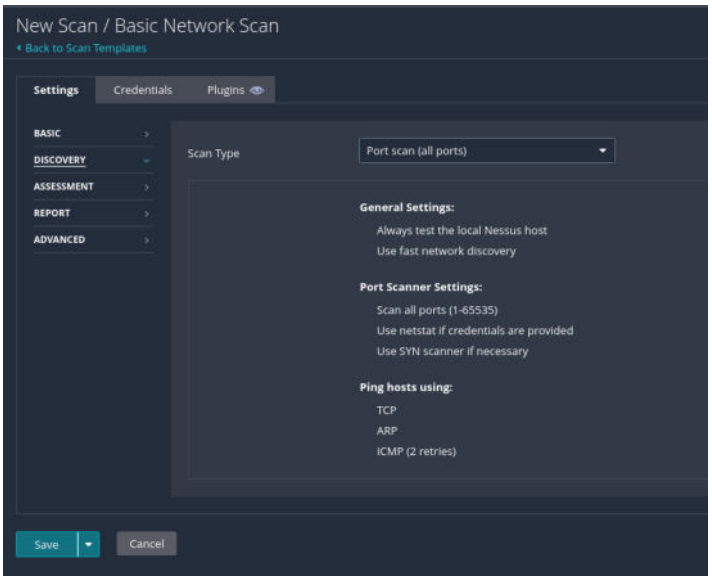
Elevate privileges with: *sudo (si el usuario tiene permisos sudo)*

Sudo password: *contraseña para sudo*

Port: *22*

Figura 39

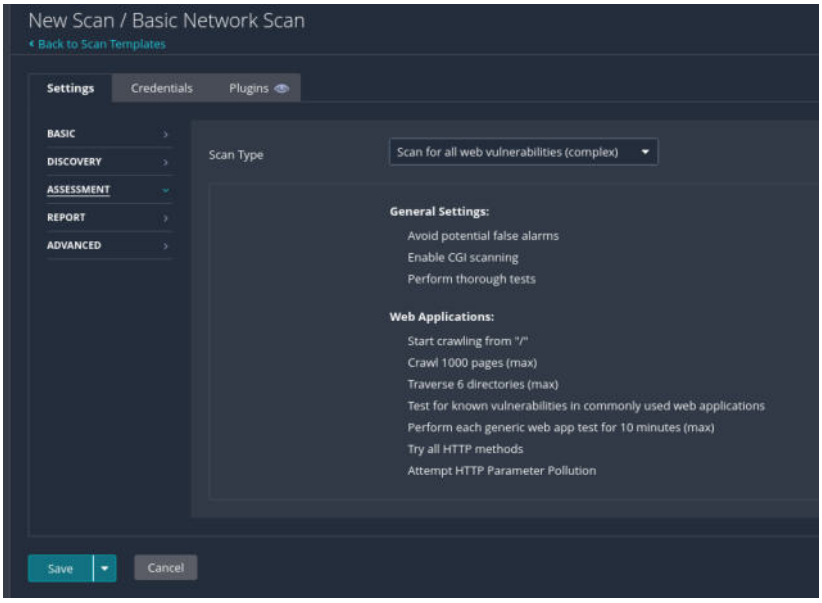
Configuración de descubrimiento de puertos



Nota: En Discovery seleccionamos "All ports".

Figura 40

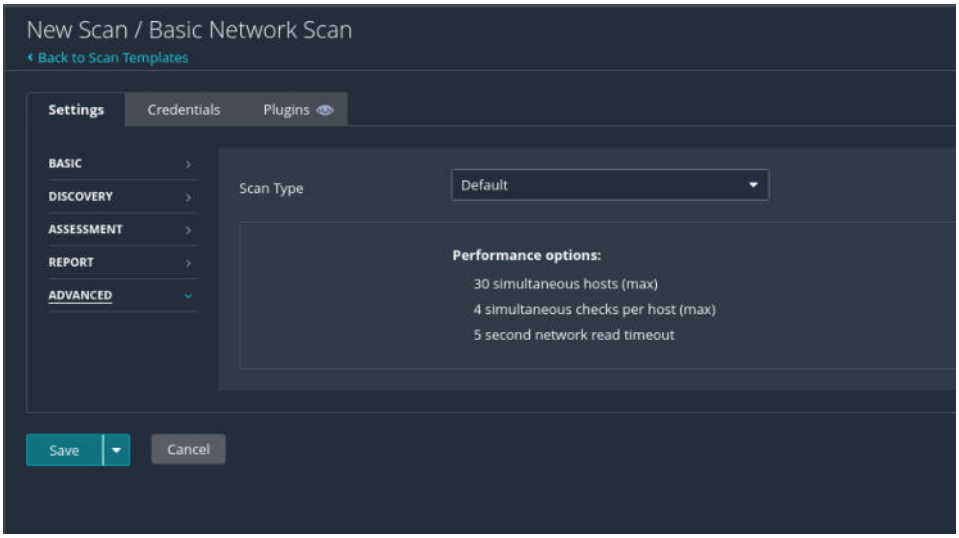
Selección del tipo de evaluación de vulnerabilidades.



Nota: En Assesment, seleccionamos "Scan for all web vulnerabilities (complex)".

Figura 41

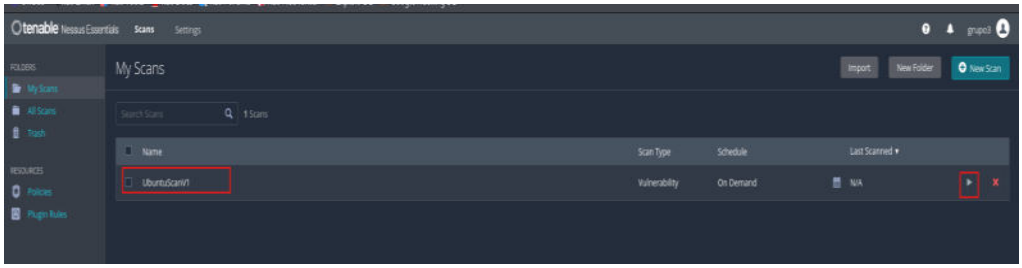
Configuración avanzada del escaneo.



Nota: En Advanced dejamos seleccionado "Default" y guardamos los cambios.

Figura 42

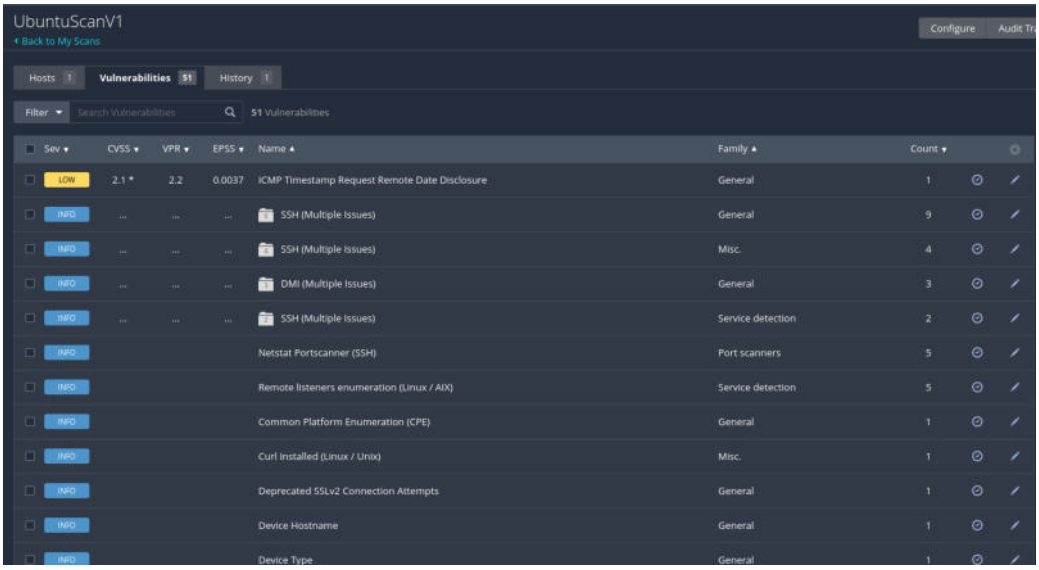
Ejecución del escaneo al servidor Ubuntu



Nota: Guardamos y lanzamos el análisis/escaneo del servidor Ubuntu.

Figura 43

Visualización de resultados del escaneo

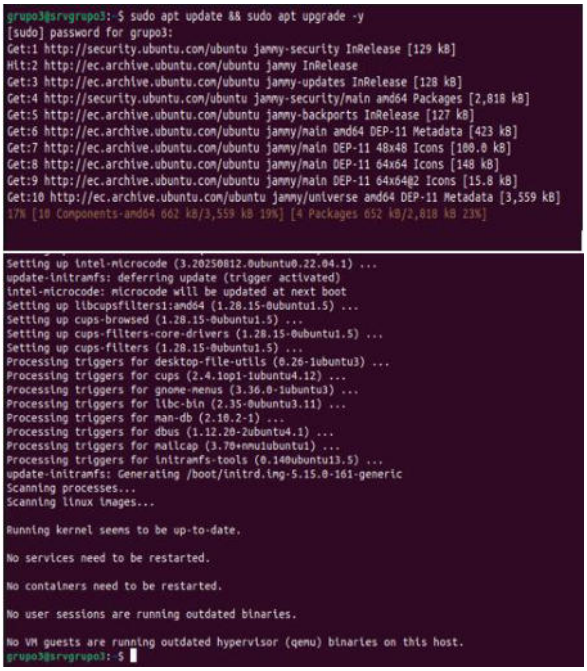


Nota: Esperamos a que el análisis finalice para poder visualizar los resultados o hallazgos que se hayan identificado.

3.1.4. Desplegando Docker sobre Ubuntu Server

Figura 44

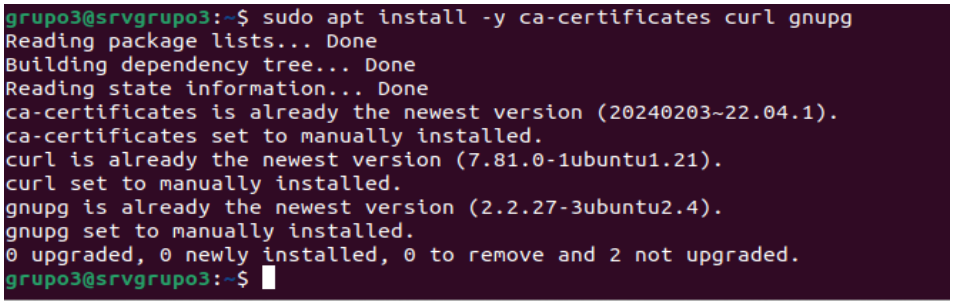
Actualización del sistema Ubuntu Server.



Nota: Para esto, el primer paso es actualizar el servidor Ubuntu, utilizamos el comando sudo apt update && sudo apt upgrade -y.

Figura 45

Instalación de dependencias para Docker



Nota: Una vez finalizada la actualización, continuamos con la instalación de Docker. Ubuntu no viene con Docker por lo que lo instalaremos con el siguiente comando sudo apt install -y ca-certificates curl gnupg.

Figura 46

Instalación de repositorios y llaves de Docker

```
root@srvgrupo3:~# # Add Docker's official GPG key:
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Suites: $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF

sudo apt update
```

Figura 47

Instalación de paquetes de Docker

```
Hit:1 http://ec.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://ec.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://ec.archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
14 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203~22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.21).
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
Types: deb
URIs: https://download.docker.com/linux/ubuntu
Suites: jammy
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:2 https://download.docker.com/linux/ubuntu jammy InRelease [48.5 kB]
Hit:3 http://ec.archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://ec.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://ec.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:6 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [62.9 kB]
Fetched 111 kB in 2s (47.9 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
14 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Figura 48

Instalación de paquetes y componentes de Docker

```
root@srvgrupo3:~# sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 14 not upgraded.
Need to get 96.2 MB of archives.
After this operation, 402 MB of additional disk space will be used.
```

Figura 49

Servicio Docker en ejecución

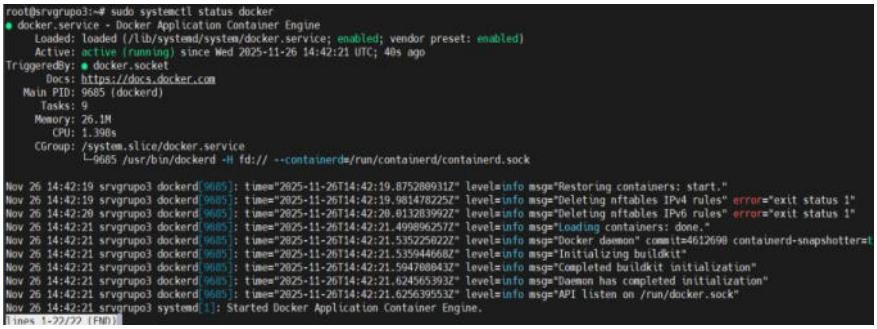


Figura 50

Verificación de la instalación de Docker

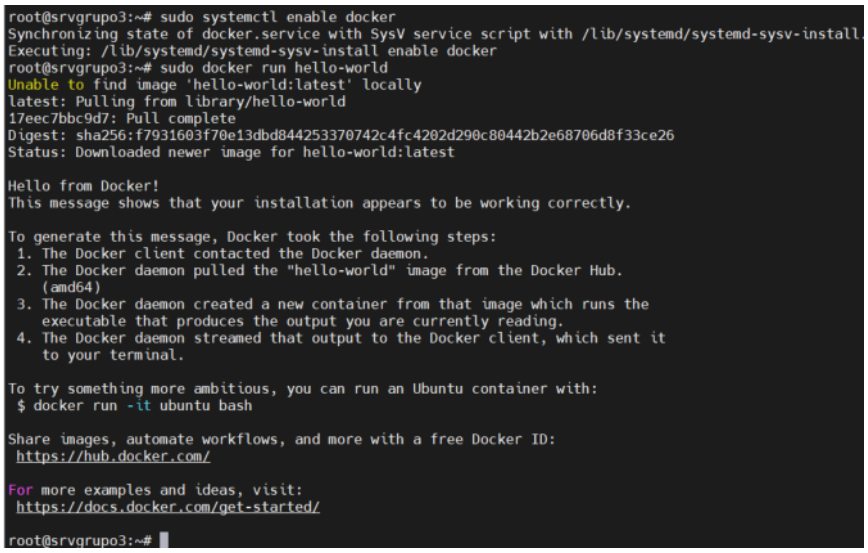


Figura 51

Ejecución de la imagen de prueba hello-world

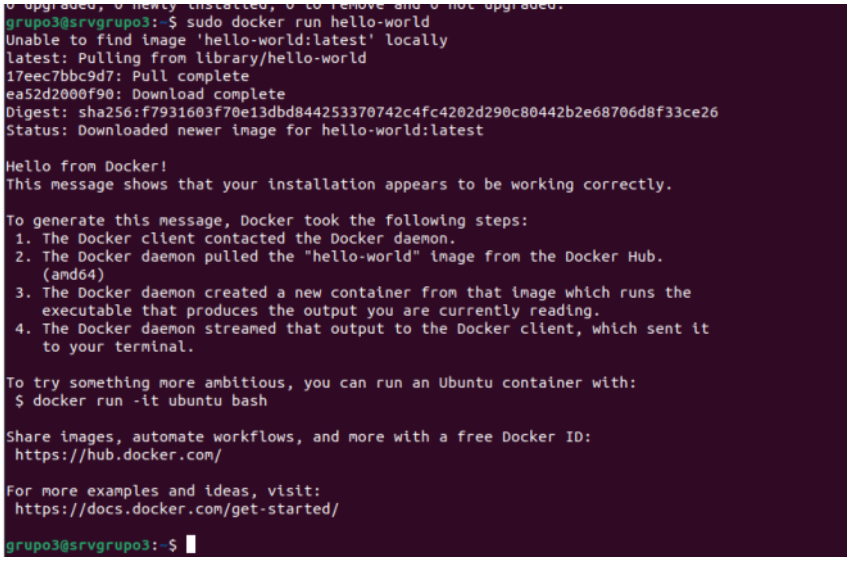
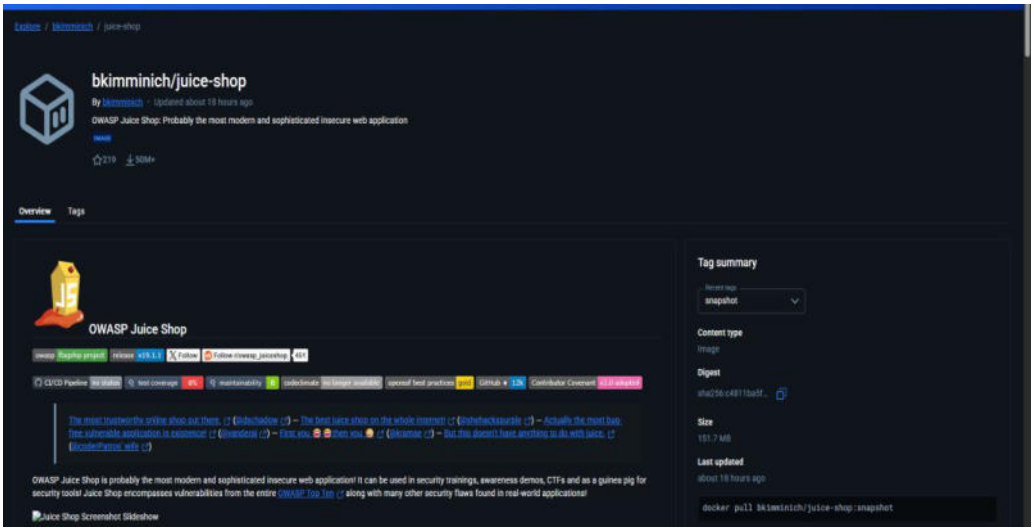


Figura 52

Despliegue de imagen vulnerable OWASP Juice Shop



Nota: Se levanta una imagen vulnerable a propósito para realizar el escaneo correspondiente, para este caso se utiliza una imagen de OWASP obtenida directamente de Docker Hub.

Figura 53

Descarga de la imagen OWASP Juice Shop

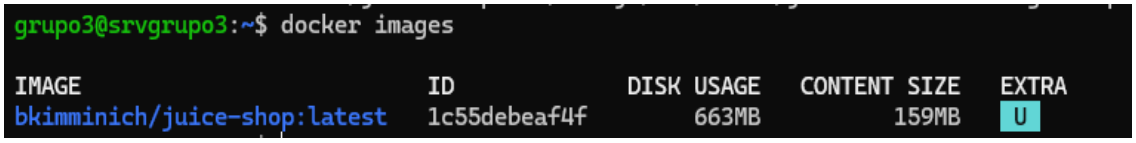
```
grupo3@srvgrupo3:~$ docker pull bkimminich/juice-shop
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
0168f69dfb16: Pull complete
fd4aa3667332: Pull complete
bfb59b82a9b6: Pull complete
017886f7e176: Pull complete
62de241dac5f: Pull complete
2780920e5dbf: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
045fc1c20da8: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
ddf74a63f7d8: Pull complete
e7fa9df358f0: Pull complete
d8a0d911b13e: Pull complete
5b14f6c9a813: Pull complete
33ce0b1d99fc: Pull complete
f45e0372ce60: Pull complete
7faf0cfa885c: Pull complete
9cd2a1476fcc: Pull complete
7b72e6384ef9: Pull complete
afac7823be98: Download complete
Digest: sha256:1c55debeaf4fd5678019b17818a539e1e06ef93d29b268a21f53f0773a9fff5d
Status: Downloaded newer image for bkimminich/juice-shop:latest
docker.io/bkimminich/juice-shop:latest
```

Nota: Instalación de la imagen seleccionada de OWASP si el CRS que son el

conjunto de reglas de detección de ataque que vienen por defecto, se lo realiza a través del comando: docker pull bkimminich/juice-shop

Figura 54

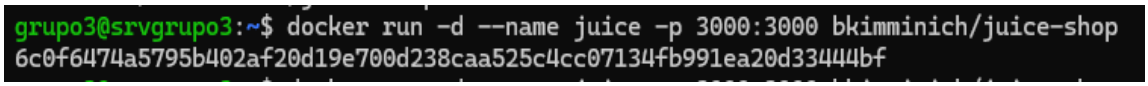
Verificación de imágenes Docker disponibles



Nota: Comprobamos que la imagen se encuentre disponible: docker images

Figura 55

Ejecución del contenedor OWASP Juice Shop



Nota: Lanzar el contenedor con: docker run -d --name juice -p 3000:3000 bkimminich/juice-shop

Figura 56

Pruebas de conectividad al contenedor

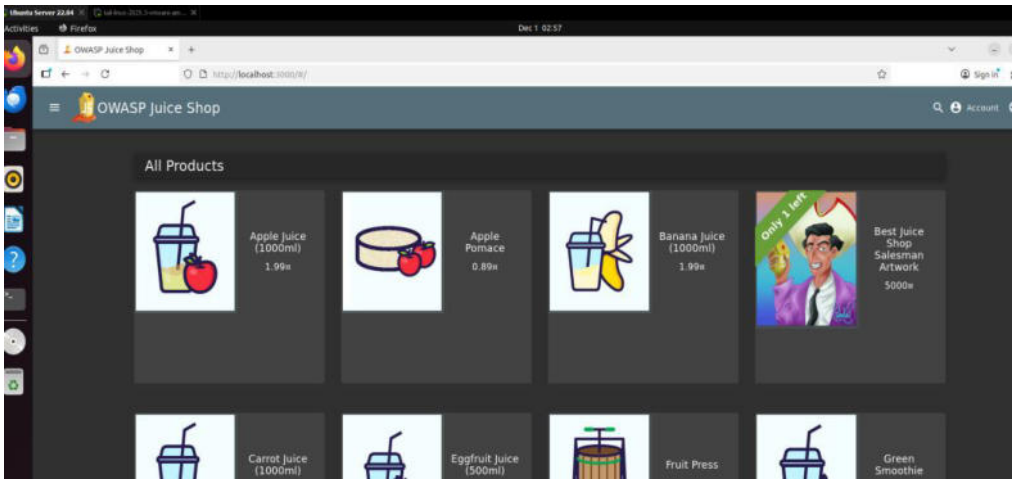
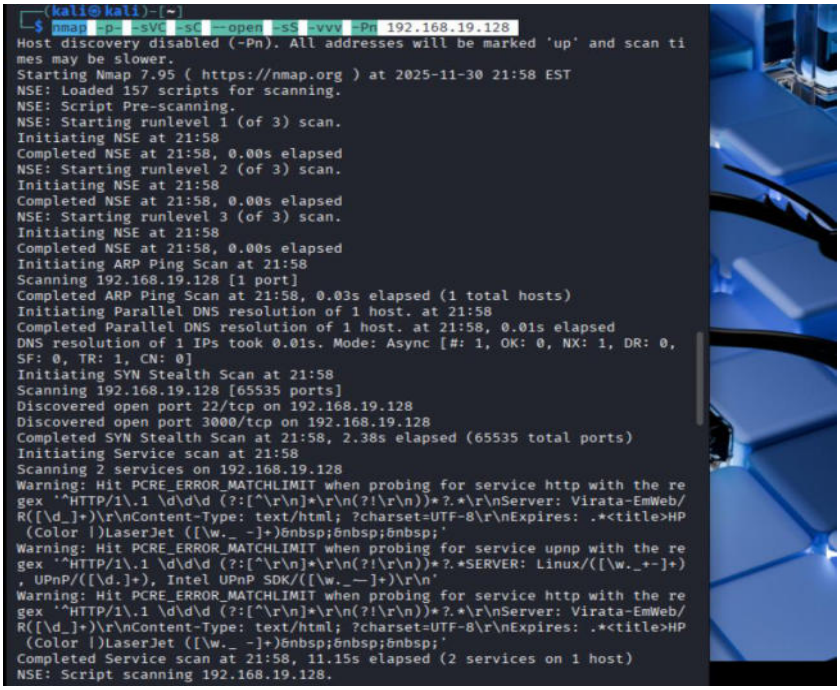


Figura 57

Escaneo de puertos desde Kali Linux



Nota: Escaneo de puertos rápido realizado desde una máquina atacante: Kali Linux nmap -p- -sVC -sC --open -sS -vvv -Pn 192.168.19.128

Figura 58

Identificación de puertos abiertos.

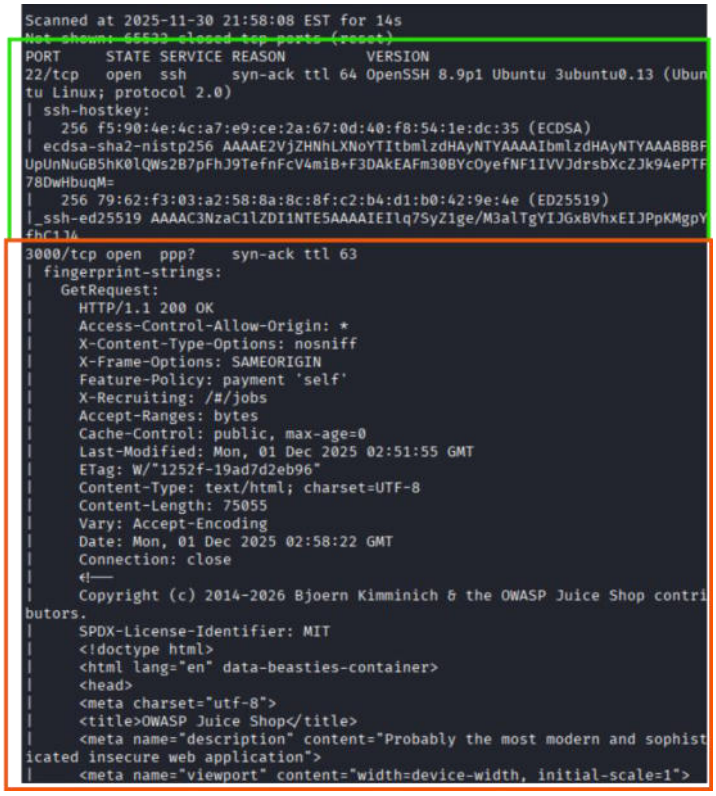
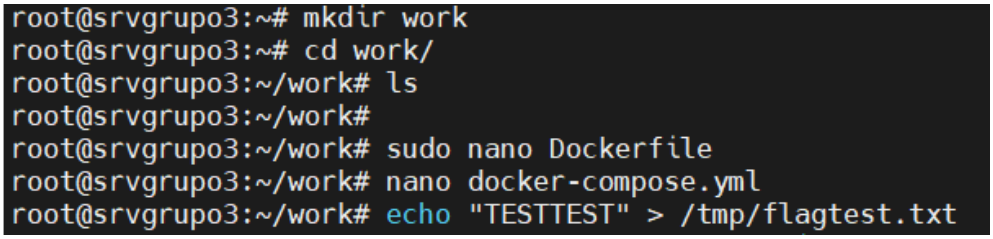


Figura 59

Creación de directorios y archivos de configuración Docker.



Nota: Creamos el directorio "work" sobre el que desplegaremos nuestro contenedor.

mkdir work: *Crea el Directorio Work.*

Sudo nano Dockerfile*: Creamos/editamos el archivo Dockerfile.*

nano docker-compose.yml: *Creamos/editamos el archivo docker-compose.yml.*

echo "TESTTEST" > /tmp/flagtest.txt: Creamos/editamos el archivo flagtest.txt y escribimos dentro TESTTEST, este es un archivo de prueba cargador intencionalmente en el contenedor.

Figura 60

Instalación de Docker Compose.

```
root@srvgrupo3:~/work# apt install docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce docker-ce-cli python3-docker python3-dockerpty
  python3-docopt python3-dotenv python3-texttable python3-websocket
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
Recommended packages:
  docker.io
The following NEW packages will be installed:
  docker-compose python3-docker python3-dockerpty python3-docopt
  python3-dotenv python3-texttable python3-websocket
The following packages will be upgraded:
  containerd.io docker-ce docker-ce-cli
3 upgraded, 7 newly installed, 0 to remove and 16 not upgraded.
Need to get 60.9 MB of archives.
After this operation, 8,549 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-websocket
 all 1.2.3-1 [34.7 kB]
Get:2 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli
amd64 5:29.1.1-1~ubuntu.22.04~jammy [16.3 MB]
Get:3 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-docker al
l 5.0.3-1 [89.3 kB]
Get:4 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-dockerpty
 all 0.4.1-2 [11.1 kB]
Get:5 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-docopt al
l 0.6.2-4 [26.9 kB]
Get:6 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-dotenv al
l 0.19.2-1 [20.5 kB]
Get:7 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 python3-texttable
 all 1.6.4-1 [11.4 kB]
Get:8 http://ec.archive.ubuntu.com/ubuntu jammy/universe amd64 docker-compose al
l 1.29.2-1 [95.8 kB]
Get:9 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io
amd64 2.2.0-2~ubuntu.22.04~jammy [23.3 MB]
Get:10 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd
64 5:29.1.1-1~ubuntu.22.04~jammy [21.1 MB]
Fetched 60.9 MB in 24s (2,502 kB/s)
(Reading database ... 164257 files and directories currently installed.)
```

Nota: A continuación, procedemos a instalar Docker compose, para esto usamos el comando apt install docker-compose.

Figura 61

Verificación de Docker Compose

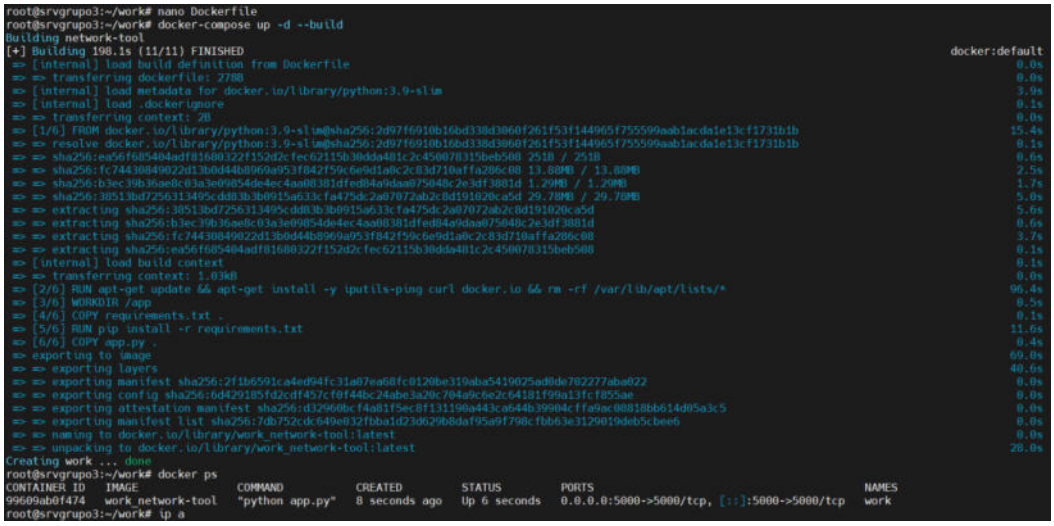
```
Preparing to unpack .../5-python3-dockerpty_0.4.1-2_all.deb ...
Unpacking python3-dockerpty (0.4.1-2) ...
Selecting previously unselected package python3-docopt.
Preparing to unpack .../6-python3-docopt_0.6.2-4_all.deb ...
Unpacking python3-docopt (0.6.2-4) ...
Selecting previously unselected package python3-dotenv.
Preparing to unpack .../7-python3-dotenv_0.19.2-1_all.deb ...
Unpacking python3-dotenv (0.19.2-1) ...
Selecting previously unselected package python3-texttable.
Preparing to unpack .../8-python3-texttable_1.6.4-1_all.deb ...
Unpacking python3-texttable (1.6.4-1) ...
Selecting previously unselected package docker-compose.
Preparing to unpack .../9-docker-compose_1.29.2-1_all.deb ...
Unpacking docker-compose (1.29.2-1) ...
Setting up python3-dotenv (0.19.2-1) ...
Setting up python3-texttable (1.6.4-1) ...
Setting up python3-docopt (0.6.2-4) ...
Setting up containerd.io (2.2.0-2~ubuntu.22.04~jammy) ...
Setting up docker-ce-cli (5:29.1.1-1~ubuntu.22.04~jammy) ...
Setting up python3-websocket (1.2.3-1) ...
Setting up python3-dockerpty (0.4.1-2) ...
Setting up python3-docker (5.0.3-1) ...
Setting up docker-ce (5:29.1.1-1~ubuntu.22.04~jammy) ...
Setting up docker-compose (1.29.2-1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@srvgrupo3:~/work# ls
docker-compose.yml  Dockerfile
```

Nota: Una vez que ha finalizado la instalación verificamos que ya tenemos

Docker-compose.yml.

Figura 62

Construcción y despliegue de la aplicación con Docker Compose



Nota: A continuación, configuramos el Dockerfile y levantamos un proyecto de Docker Compose, es decir, construimos la image definida en el archivo Dockerfile.

Como se puede ver en la imagen anterior, hemos levantado una aplicación Python "app.py" dentro del contenedor en el puerto 5000, mismo que queda expuesto para poder acceder desde fuera.

CAPITULO 4:

4. ANÁLISIS DE RESULTADOS

4.1.Análisis de Resultados

4.1.1. Informe Ejecutivo de Vulnerabilidades

4.1.1.1.Objetivo

El objetivo del presente documento es mostrar el resultado del análisis realizado a nivel de seguridad al activo con IP 192.168.89.145, de Docker con un web Service expuesto. El estudio incluye métricas de criticidad, impacto y riesgo, así como recomendaciones para el tratamiento de las vulnerabilidades identificadas.

4.1.1.2.Alcance

Este análisis de Seguridad se ciñó al activo 192.168.89.145 “Docker”.

4.1.1.3.Resumen Ejecutivo

La aplicación expuesta presenta una cadena de vulnerabilidades que permite a un atacante remoto pasar de una simple inyección de comandos dentro del contenedor hasta un escape completo al host Docker, logrando leer archivos del sistema anfitrión.

La cadena es viable debido a:

- Inyección de comandos OS en la aplicación web.
- Contenedor con acceso al Docker Socket (/var/run/docker.sock).
- Capacidad del atacante de lanzar contenedores desde el contenedor comprometido, permitiendo un container breakout completo.

Estas vulnerabilidades representan un riesgo crítico y permiten toma de control total del host.

4.1.1.4.Estadísticas de vulnerabilidades

A continuación, se muestra el detalle de las vulnerabilidades identificadas por criticidad:

Tabla 2

Vulnerabilidades identificadas

Criticidad	Crítica	Grave	Media	Baja
# Vulns.	3	0	0	0

Figura 63

Grafica de las vulnerabilidades identificadas

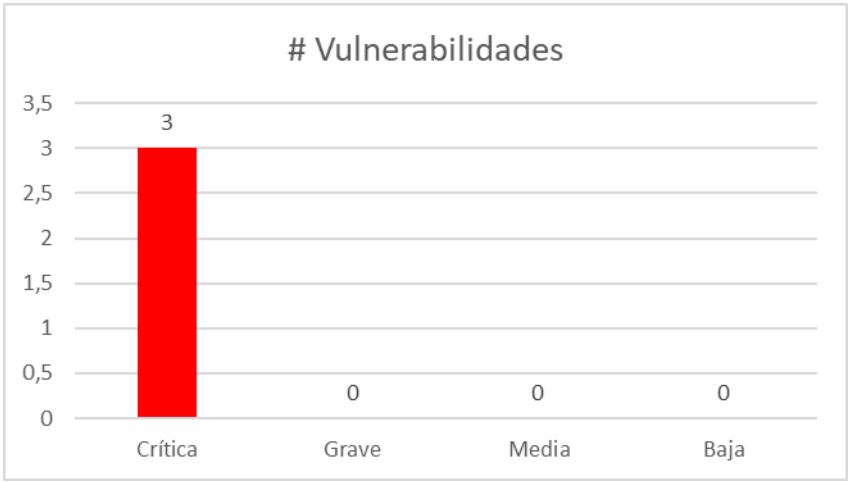


Tabla 3

Tabla resumen de vulnerabilidades

Vulnerabilidad	Activo	Criticidad	Estado
Inyección de Comandos en la Aplicación Web	http://192.168.89.145	CRÍTICA	
Exposición del Docker Socket dentro del Contenedor	http://192.168.89.145	CRÍTICA	

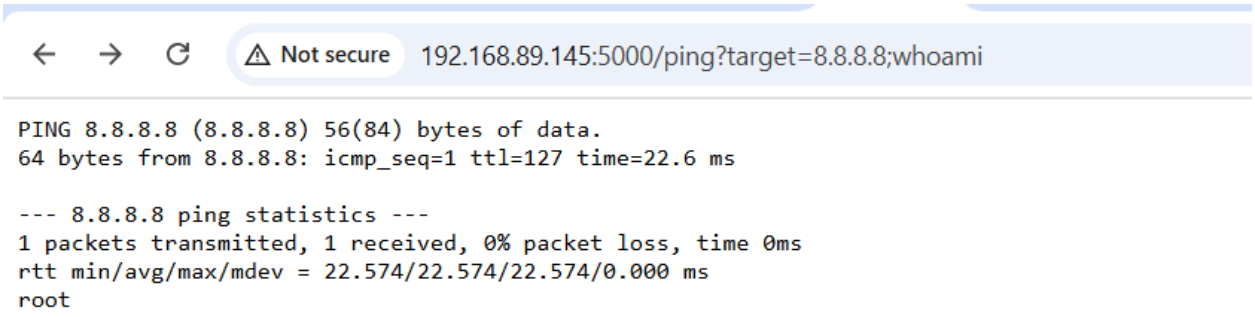
Container Breakout: Acceso al Host usando Docker desde el Contenedor	http://192.168.89.145	CRÍTICA

Tabla 4

Detalle Técnico de vulnerabilidades

CRÍTICA	
Descripción	Durante el análisis de seguridad se identificó que la aplicación permite concatenar comandos del sistema mediante caracteres como ; en el parámetro target: • /ping?target=8.8.8.8;whoami Esto se debe a que el valor del parámetro es pasado directamente a una llamada del sistema operativo sin validación ni sanitización.
Activos	192.168.89.145
Vector CVSS	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H • CWE-77: Improper Neutralization of Special Elements used in a Command (Command Injection) • CVE-2021-3129 (Ejemplo de RCE por command injection en Laravel)
CVSS	9.8
Recomendación	• Usar ejecución segura (por ejemplo, subprocess.run con shell=False en Python). • Validar y sanitizar entradas del usuario. • Usar listas blancas para parámetros (solo IPs o dominios válidos). • Implementar un WAF o reglas específicas para evitar inyecciones.

Detalle de la vulnerabilidad



Nota: Inyección de Comandos en la Aplicación Web

Tabla 5

Exposición del Docker Socket dentro del Contenedor

CRÍTICA	
Descripción	• Durante el análisis de seguridad se identificó que el atacante logra ejecutar: • docker ps • Esto demuestra que el contenedor tiene acceso al Docker daemon del host por medio del socket: • /var/run/docker.sock • Esto es una mala configuración crítica: dar acceso al socket equivale a dar acceso root al host.
Activos	192.168.89.145
Vector CVSS	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
	• CWE-250: Execution with Unnecessary Privileges • CWE-732: Incorrect Permission Assignment for Critical Resource

	• CVE-2019-5736 (runc) – aunque no se trata de este exploit, demuestra el impacto de comprometer contenedores con acceso al runtime.
CVSS	10

Recomendación	• Nunca montar /var/run/docker.sock en un contenedor salvo necesidad muy controlada.
	• Usar Docker Rootless mode si es posible.
	• Usar políticas de seguridad:
	  ○ Seccomp
	  ○ AppArmor
	  ○ SELinux
	• Separar contenedores con least privilege.

Detalle de la vulnerabilidad

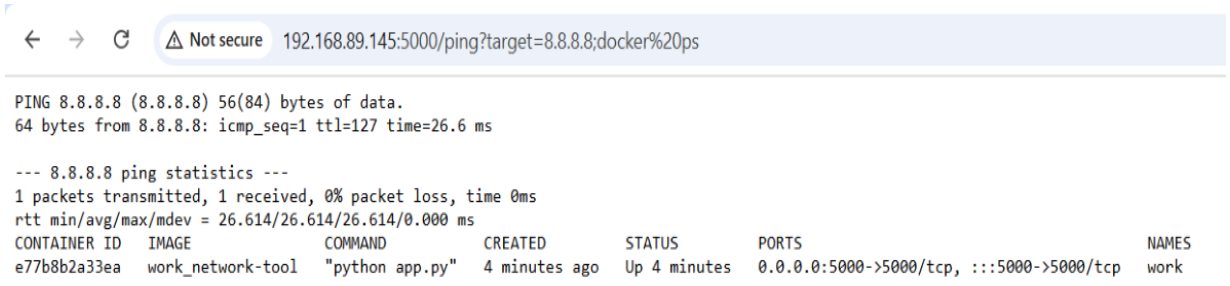


Tabla 6

Container Breakout: Acceso al Host usando Docker desde el Contenedor

GRAVE	
Descripción	• Durante el análisis de seguridad se identificó que el atacante ejecuta:

	• docker run --rm -v /:/host_mnt alpine cat /host_mnt/tmp/flag_tesis.txt • Esto monta el sistema de archivos real del host dentro de un nuevo contenedor iniciado desde el contenedor comprometido.
Activos	http:// 192.168.89.145
Vector CVSS	• CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H • CWE-284: Improper Access Control • CWE-267: Privilege Defined with Unsafe Actions
CVSS	10
Recomendación	• Eliminar acceso al docker.sock desde contenedores. • Implementar un proxy seguro si se necesita acceso (ej. Docker Socket Proxy). • Usar contenedores inmutables o de menor privilegio. • Configurar no-new-privileges. • Escanear host por persistencia posterior al compromiso.

4.1.1.5.Solución de las vulnerabilidades:

Detalle del problema

En el proyecto app.py que tenemos, el código tiene dos errores principales:

1. **Command Injection (CWE-78)**

Figura 64

Command Injection

```
command = f"ping -c 1 {target}"
output = subprocess.check_output(command, shell=True, stderr=subprocess.STDOUT)
```

2. No hay una validación de que el target sea una IP o dominio legítimo.

a) Eliminación de shell=True

En la línea que presenta el problema:

```
subprocess.check_output(command, shell=True)
```

Corregimos retirando Shell, de esta forma:

```
subprocess.check_output(["ping", "-c", "1", target])
```

Como resultado el usuario ya no podrá inyecto comandos pues con esto se logra que la entrada sea tratada como parámetros y no como una cadena ejecutado por un Shell.

b) No concatenar cadenas del usuario en comandos del sistema

En el parámetro target lo corregimos pasando de

```
command = f"ping -c 1 {target}"
```

```
["ping", "-c", "1", target]
```

Con esto logramos corregir el 100% de inyecciones de código por esta falla en el diseño.

c) Validación estricta del parámetro

Adicional, Usamos regex para garantizar que el target sea solo una IP o dominio válido.

Esto protege incluso si en el futuro otro desarrollador vuelve a tocar el código.

d) Manejo seguro de errores

Las excepciones en caso de que hubieren, las hemos configurado para manejarlas y que devuelvan en texto plano sin exponer stack traces.

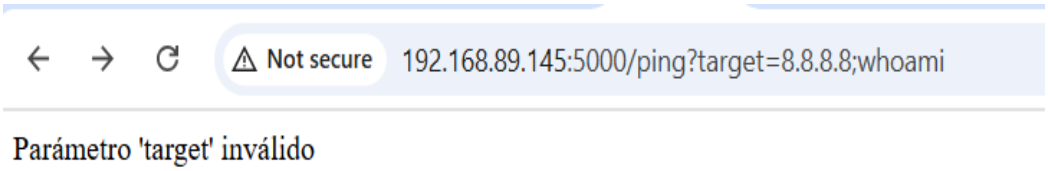
El código corregido queda de la siguiente manera:

```
from flask import Flask, request
import subprocess
import re
app = Flask(__name__)
def is_valid_target(target: str) -> bool:
    """
    Valida si el parámetro es una IP o dominio simple.
    Reglas estrictas para evitar command injection
    """
    ip_regex = r"^\d{1,3}(\.\d{1,3}){3}$"
    domain_regex = r"^[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
    return re.match(ip_regex, target) or re.match(domain_regex,
target)
@app.route('/')
def home():
    return """
    <h1>MockaCorp Network Tool (Segura)</h1>
    <p>Uso: /ping?target=google.com</p>
    """
@app.route('/ping')
def ping():
    target = request.args.get('target')
```

```
    if not target:
        return "Falta el parámetro 'target'", 400
    # Validación estricta
    if not is_valid_target(target):
        return "Parámetro 'target' inválido", 400
    try:
        # Ejecución segura: NO usa shell=True y NO concatena
strings
        output = subprocess.check_output(
            ["ping", "-c", "1", target],
            stderr=subprocess.STDOUT
        )
        return f"<pre>{output.decode('utf-8')}</pre>"
    except subprocess.CalledProcessError as e:
        return f"<pre>Error:\n{e.output.decode('utf-8')}</pre>"


if __name__ == '__main__':
    # Escucha en todas las interfaces
    app.run(host='0.0.0.0', port=5000)
```

Figura 65

Evidencias pos solución:



CAPITULO 5:

5. CONCLUSIONES Y RECOMENDACIONES

5.1.Conclusiones

El análisis evidenció que la adopción de Docker en entornos corporativos optimiza la portabilidad, escalabilidad y despliegue de aplicaciones. Sin embargo, estas ventajas vienen acompañadas de nuevos vectores de ataque derivados del aislamiento incompleto, configuraciones por defecto inseguras y fallos en la cadena de suministro de imágenes.

Se comprobó que una proporción significativa de vulnerabilidades detectadas no están relacionadas con fallos inherentes a Docker, sino con configuraciones inseguras: contenedores ejecutados como root, ejecución de shell, imágenes sin verificar o uso de redes por defecto.

Esto demuestra que la seguridad “out-of-the-box” de Docker es insuficiente sin controles adicionales.

Aunque Docker emplea namespaces y cgroups, se observó que un mal uso de volúmenes, redes compartidas o capacidades privilegiadas puede romper el aislamiento, abriendo la puerta a escaladas de privilegios o movimientos laterales.

Se verificó que los controles de seguridad no deben depender únicamente del motor Docker, sino integrarse en ecosistemas más amplios de monitoreo, gestión de identidades, CI/CD y respuesta a incidentes.

Al realizar el análisis de las vulnerabilidades del contenedor docker, se concluye que, en este tipo de ambientes, el tema de la seguridad inicia desde la autenticidad de las imágenes que se utiliza, tanto en su origen y su integridad. Se puedo constatar que la falta de verificación de estas, implican que tengan vulnerabilidades, librerías y paquetes desactualizados, configuraciones inseguras o innecesarias para el fin que se les va a utilizar, además pueden contener código malicioso. Es importante que las imágenes posean sus firmas digitales, provengan de fuentes oficiales y se realicen escaneos de vulnerabilidades antes de ser puestas a producción.

La ejecución de contenedores Docker se debe realizar con los privilegios mínimos asignados, es decir, los necesarios para su funcionamiento. Tener en cuenta que se debe evitar de cualquier manera la ejecución de procesos como usuario root, con esto evitamos ataques y la escalabilidad de estos hacia la máquina Host y demás dispositivos de la red, además, se deben limitar los recursos que se asignen, con el fin de evitar una saturación del Host. Como se conoce, los bajos requerimientos de recursos y optimización de procesos son unas de las características principales de entornos Docker.

5.2.Recomendaciones

Se recomienda tomar medidas de prevención con el propósito de fortalecer la seguridad realizando una configuración inicial adecuada basándose en políticas de endurecimiento, inhabilitación de capacidades innecesarias, establecimiento de imágenes de fuentes confiables y adopción de herramientas que permitan el escaneo automático constante para detectar vulnerabilidades antes de su despliegue; de esta forma la incorporación de contenedores se integra no solo al plano operativo sino

también a la gobernanza tecnológica

Debido a que los errores de configuración son frecuentes en entornos con contenedores, llevar a cabo un programa de capacitación interna que incluya guías prácticas, laboratorios y revisiones de configuración continúa dedicada a equipos de desarrollo y operaciones es de vital importancia ya que de esta forma se estandarizan los criterios técnicos para prevenir configuraciones débiles desde un inicio.

Se recomienda diseñar una infraestructura lógica con una autenticación de alto nivel entre servicios, así como el monitoreo en tiempo real, esto con el propósito de detectar posibles desviaciones y limitar la exposición ante movimientos laterales potenciales o incluso escalada de privilegios manteniendo así la continuidad de las operaciones incluso ante un intento de explotación.

Es recomendable mantener actualizados los sistemas utilizados, sea en la máquina Host o los sistemas operativos de los contenedores, esto evitará la presencia de vulnerabilidades. Con esto también, se debe deshabilitar los servicios, puertos, paquetes, librerías o eliminar archivos que tengas información sensible, con el fin de aumentar la seguridad de todo en entorno, es decir, el contenedor, el Host y los dispositivos en red.

En importante implementar monitores y realizar auditorías en toda la red, es decir desde el deamon de Docker, los contenedores, el Host, los dispositivos de la red, como routers y firewalls. Es recomendable realizar un análisis constante del tráfico y las conexiones que se realizan desde dentro y fuera de la red. Mantener controlado un ambiente, bien monitoreado y un registro constante de los logs, crea un ambiente seguro y permite prevenir ataques, mal funcionamiento de los equipos o agotamiento de los recursos designados.

REFERENCIAS BIBLIOGRÁFICAS

AWS. (s.f.). Recuperado el 4 de noviembre de 2025, de https://aws.amazon.com/es/compare/the-difference-between-containers-and-virtual-machines/

AWS. (s.f.). Recuperado el 4 de noviembre de 2025, de https://aws.amazon.com/es/docker/

Blanch, A. (15 de octubre de 2024). Ventajas de Docker y sus contenedores. Recuperado el 4 de noviembre de 2025, de arsys: https://www.arsys.es/blog/docker-ventajas-contenedores

Cloud, G. (s.f.). Recuperado el 13 de noviembre de 2025, de Que es Kubernetes: https://cloud.google.com/learn/what-is-kubernetes?hl=es-419

Datadog. (s.f.). Recuperado el 13 de noviembre de 2025, de Descripción general de la orquestación de contenedores: https://www-datadoghq-com.translate.goog/knowledge-center/container-orchestration/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

dockerdocs. (s.f.). Recuperado el 4 de noviembre de 2025, de Arquitectura de Docker: https://docs.docker.com/get-started/docker-overview/#docker-architecture

García, Y. F. (septiembre de 2011). Virtualizacion. Obtenido de Revista Telematica: http://biblioteca.udgvirtual.udg.mx/jspui/bitstream/123456789/2281/1/Virtualizaci%c3%b3n.pdf

gitbooks. (s.f.). Recuperado el 4 de noviembre de 2025, de Arquitectura de Docker: https://jsitech1.gitbooks.io/meet-docker/content/arquitectura_de_docker.html

IBM. (s.f.). Recuperado el 4 de noviembre de 2025, de https://www.ibm.com/es-es/think/topics/docker

IBM. (s.f.). ¿Qué es la virtualización? Recuperado el 3 de noviembre de 2025, de IBM: https://www.ibm.com/mx-es/think/topics/virtualization

Liang, Y. (5 de febrero de 2024). CROWDSTRIKE. Recuperado el 15 de noviembre de 2025, de Fundamentos de la ciberseguridad: https://www-crowdstrike-com.translate.goog/en-us/cybersecurity-101/cloud-security/center-for-internet-security-cis-benchmark/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Ltd, C. G. (29 de octubre de 2025). Servidor Ubuntu. Recuperado el 5 de noviembre de 2025, de Almacenamiento, redes y registro de Docker: https://documentation-ubuntu-com.translate.goog/server/explanation/virtualisation/docker-storage-networking-and-logging/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

NetApp. (s.f.). NetApp. Recuperado el 3 de noviembre de 2025, de ¿Qué son los contenedores?: https://www.netapp.com/es/devops/what-are-containers/

NETWORK, p. (2025). Recuperado el 13 de noviembre de 2025, de ¿Qué es la seguridad de Kubernetes?: https://www-paloaltonetworks-co-uk.translate.goog/cyberpedia/kubernetes-security?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Puzas, D. (31 de marzo de 2024). Recuperado el 4 de noviembre de 2025, de ¿Qué es la seguridad de contenedores?: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/container-security/

SentinelOne. (4 de agosto de 2025). Recuperado el 20 de noviembre de 2025, de ¿Qué es la seguridad de las imágenes de contenedores?: https://www.sentinelone.com/es/cybersecurity-101/cloud-security/container-image-security/#2-dependencias-de-terceros-inseguras

Sheps, A. (7 de abril de 2024). aqua. Recuperado el 20 de noviembre de 2025, de ¿Qué significa escape de contenedor?: https://www-aquasec-com.translate.goog/cloud-native-

academy/container-security/container-escape/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge

Susnjara, S. y. (s.f.). ¿Qué es una máquina virtual (VM)? Recuperado el 3 de noviembre de 2025, de IBM: https://www.ibm.com/mx-es/think/topics/virtual-machines

UNIR. (25 de septiembre de 2025). ¿Qué es una Máquina Virtual? Tipos, usos y beneficios de las VM. Recuperado el 3 de noviembre de 2025, de UNIR: https://ecuador.unir.net/actualidad-unir/que-es-maquina-virtual/

wazuh. (11 de septiembre de 2024). Recuperado el 13 de noviembre de 2025, de Análisis de la infraestructura Docker según el estándar CIS: https://wazuh-com.translate.goog/blog/scanning-docker-infrastructure-against-cis-benchmark/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Anchore. (2025). Docker security best practices: A complete guide. Anchore Blog. https://anchore.com/blog/docker-security-best-practices-a-complete-guide/

Better Stack. (2025). Docker security best practices: Comprehensive guide. BetterStack Community Guides. https://betterstack.com/community/guides/scaling-docker/docker-security-best-practices/

CyberPanel. (2024). Docker security best practices and common concerns. CyberPanel Blog. https://cyberpanel.net/blog/docker-security-best-practices

OPS Moon. (2025). Docker security best practices: A modern guide. OPSMoon Blog. https://opsmoon.com/blog/docker-security-best-practices

SUSE. (2024). Six essential Docker security best practices for safe containers. SUSE Blog. https://www.suse.com/c/six-essential-docker-security-best-practices-for-safe-containers/

Sonatype. (2025). Docker security best practices. Sonatype Guides. https://www.sonatype.com/resources/guides/docker-security-best-practices

Wiz.io. (2025). Docker container security best practices. Wiz Academy. https://www.wiz.io/academy/docker-container-security-best-practices

Cloud Native Now. (2025). Docker security in 2025: Best practices to protect your containers from cyberthreats. CloudNativeNow. https://cloudnativenow.com/topics/cloudnativedevelopment/docker/docker-security-in-2025-best-practices-to-protect-your-containers-from-cyberthreats/

Anchore. (2025). Docker security best practices: A complete guide. Anchore Blog. https://anchore.com/blog/docker-security-best-practices-a-complete-guide/

Better Stack. (2025). Docker security best practices: Comprehensive guide. BetterStack Community Guides. https://betterstack.com/community/guides/scaling-docker/docker-security-best-practices/

CyberPanel. (2024). Docker security best practices and common concerns. CyberPanel Blog. https://cyberpanel.net/blog/docker-security-best-practices

OPS Moon. (2025). Docker security best practices: A modern guide. OPSMoon Blog. https://opsmoon.com/blog/docker-security-best-practices

SUSE. (2024). Six essential Docker security best practices for safe containers. SUSE Blog. https://www.suse.com/c/six-essential-docker-security-best-practices-for-safe-containers/

Sonatype. (2025). Docker security best practices. Sonatype Guides. https://www.sonatype.com/resources/guides/docker-security-best-practices

Wiz.io. (2025). Docker container security best practices. Wiz Academy. https://www.wiz.io/academy/docker-container-security-best-practices

Cloud Native Now. (2025). Docker security in 2025: Best practices to protect your containers from cyberthreats. CloudNativeNow. https://cloudnativenow.com/topics/cloudnativedevelopment/docker/docker-security-in-2025-best-practices-to-protect-your-containers-from-cyberthreats/

Vallejos Quiñones, D. H. (2021). Seguridad de Contenedores Docker mediante Procesos de Hardening. INF-FCPN-PGI Revista PGI. https://ojs.umsa.bo/ojs/index.php/inf_fcpn_pgi/article/view/97

Hurtado Sáenz, M. A. (2021). Propuesta de metodología de Pentesting en contenedores Docker. Universidad de los Andes. https://hdl.handle.net/1992/55666

CCN-CERT. (2022). Seguridad en contenedores — CCN-STIC-652A. https://www.ccn-cert.cni.es/es/guias-de-acceso-publico-ccn-stic/5189-ccn-stic-652a-seguridad-en-contenedores/file.html?utm_source=chatgpt.com

Antepara Reyes, J. C., & Villamar Flores, L. N. (2020). Análisis de vulnerabilidades y definición de contramedidas en la seguridad de aplicaciones, basadas en contenedores usando la herramienta Docker. Universidad de Guayaquil. http://repositorio.ug.edu.ec/handle/redug/48780

Apéndices (si procede)

Link en el que se encuentra compartidos los laboratorios prácticos desarrollados

- https://1drv.ms/f/c/730d7c831505e9d4/IgDEscmBNg9iTpXCqmBvsPITAQ07KU3D8PORCGcJIqM5VIc?e=jmt1Zg