IMPLEMENTACIÓN DE SEGURIDAD EN SISTEMAS OPERATIVOS GNU/LINUX

Adrian Esteban Ramírez Jimenez
e-mail: adrian.ramirez@unadvirtual.edu.co

RESUMEN: *Este artículo presenta la implementación de una infraestructura segura en sistemas operativos GNU/Linux, siguiendo los lineamientos de la Etapa 7 del Diplomado de Profundización en Administración de Sistemas Operativos Open Source. El propósito central es fortalecer las capas esenciales del sistema mediante la aplicación de técnicas de hardening que permitan reducir la superficie de ataque y mitigar riesgos asociados a servicios mal configurados, accesos no autorizados y exposición de información sensible. Para ello se aplicaron medidas como la actualización integral del sistema, la configuración restrictiva del firewall UFW, el aseguramiento del servicio SSH mediante la modificación de parámetros críticos, la gestión segura de usuarios y privilegios, y la protección de archivos esenciales del sistema como /etc/shadow.*

Asimismo, se implementó un entorno web seguro basado en Apache2, integrando certificados SSL para habilitar comunicaciones cifradas mediante HTTPS y reforzando cabeceras de seguridad orientadas a prevenir ataques comunes de red. El artículo también describe los procedimientos empleados para validar la correcta operación de los servicios protegidos, incluyendo la revisión de puertos, monitoreo de procesos, análisis de logs y pruebas directas de conectividad. Los resultados obtenidos evidencian que la aplicación sistemática de medidas de endurecimiento incrementa significativamente la resiliencia del sistema frente a amenazas informáticas, favoreciendo la integridad, disponibilidad y confidencialidad de los recursos. El documento se estructura conforme a los lineamientos del formato IEEE para reportes técnicos y expone tanto el fundamento teórico como la ejecución práctica de los mecanismos implementados.

PALABRAS CLAVE: *Ciberseguridad, GNU/Linux, Firewall, SSH, Certificados SSL, Apache, Hardening.*

1 INTRODUCCIÓN

Los sistemas operativos GNU/Linux se han posicionado como uno de los pilares fundamentales en la infraestructura tecnológica contemporánea, siendo ampliamente utilizados en centros de datos, servicios en la nube, plataformas empresariales, servidores web, sistemas de virtualización, dispositivos IoT y entornos académicos. Su diseño modular, su arquitectura orientada a la estabilidad, la transparencia de su código fuente y el respaldo de una comunidad global especializada lo convierten en un entorno flexible y altamente configurable. Sin embargo, esta libertad de configuración también implica una responsabilidad directa por parte del administrador del sistema: garantizar que cada componente esté correctamente asegurado para evitar amenazas potenciales que puedan comprometer la integridad, disponibilidad o confidencialidad de la información.

La instalación inicial de una distribución GNU/Linux ofrece un punto de partida funcional, pero no necesariamente seguro. Las configuraciones predeterminadas suelen priorizar la compatibilidad y facilidad de uso antes que la seguridad, dejando abiertos servicios innecesarios, puertos sin filtrar o parámetros que un atacante experimentado podría aprovechar. Por esta razón, aplicar estrategias de hardening se convierte en una práctica indispensable dentro de cualquier organización que adopte Linux como base de su infraestructura tecnológica. El hardening permite restringir, controlar y fortalecer cada capa del sistema operativo mediante la desactivación de servicios no esenciales, la implementación de políticas de acceso rigurosas, la protección de archivos críticos y la aplicación de herramientas de control y monitoreo continuo.

En este artículo se presenta un proceso completo de fortificación del sistema desarrollado como parte de la Etapa 7 del Diplomado de Profundización en Administración de Sistemas Operativos Open Source. El ejercicio consistió en configurar una máquina Linux desde cero, endureciendo sus servicios y ajustando parámetros claves para lograr un entorno operativo seguro. Entre las actividades realizadas se incluyen la actualización integral del sistema, la configuración del firewall UFW para restringir el tráfico no autorizado, la implementación de medidas avanzadas de seguridad en el servicio SSH, la gestión precisa de usuarios y privilegios administrativos, la protección de archivos sensibles del sistema y la activación de mecanismos de auditoría para detectar comportamientos irregulares.

Además, se llevó a cabo la implementación de un servidor web seguro utilizando Apache2, integrando certificados SSL para habilitar la navegación cifrada mediante HTTPS. Este procedimiento permitió reforzar la protección del canal de comunicación entre el cliente y el servidor, mitigando riesgos asociados a ataques como sniffing, manipulación de datos, ataques de intermediario (MITM) o suplantación de identidad. Se complementó la configuración con cabeceras de seguridad recomendadas por organismos y comunidades de práctica, tales como X-Frame-Options y X-XSS-Protection, con el propósito de elevar el nivel de defensa frente a vulnerabilidades conocidas en aplicaciones web.

La relevancia de este proceso radica en que la seguridad informática no debe entenderse como un conjunto de acciones aisladas, sino como un sistema integral de medidas interrelacionadas que funcionan de manera conjunta para asegurar la plataforma tecnológica. En este sentido, el artículo no solo describe los pasos realizados, sino que analiza la importancia de cada uno dentro del ecosistema Linux, resaltando su impacto en la estabilidad operativa y la prevención de ataques. La experiencia desarrollada en esta etapa del diplomado permite consolidar habilidades fundamentales para administradores de sistemas, aportando competencias prácticas alineadas con las necesidades actuales de la industria TI.

2 DESARROLLO DE LA CONFIGURACIÓN DE SEGURIDAD

La implementación de medidas de seguridad en un sistema GNU/Linux requiere la integración de prácticas de hardening basadas en principios de administración segura, control de acceso, actualización continua y protección activa de los recursos críticos. Esta sección describe la ejecución técnica de cada uno de los componentes configurados, incorporando fundamentos teóricos, justificación académica y análisis de impacto en la seguridad del entorno operativo.

2.1 Endurecimiento del Sistema Operativo

El endurecimiento del sistema operativo, conocido como *Operating System Hardening*, es un proceso sistemático que consiste en aplicar configuraciones avanzadas de seguridad para reducir la superficie de ataque disponible a posibles amenazas externas e internas. Su propósito es eliminar vulnerabilidades inherentes, deshabilitar servicios innecesarios, proteger los recursos críticos del sistema y garantizar un funcionamiento estable bajo políticas estrictas de control. Este proceso se fundamenta en el principio de **seguridad por defecto**, el cual sugiere que un sistema debe comenzar desde un estado mínimo, seguro y fuertemente restringido, y posteriormente habilitar solo las funciones indispensables para su operación.

Desde la perspectiva académica, el hardening se encuentra ampliamente documentado en estándares como CIS Benchmarks, NIST SP 800-123 y las directrices del SANS Institute. Estas entidades coinciden en que un sistema operativo sin endurecer constituye una puerta abierta a exploits, vulnerabilidades de día cero, escalamiento de privilegios y accesos no autorizados. Además, un sistema sin actualización constante mantiene versiones con fallos conocidos, lo que facilita ataques automatizados basados en catálogos públicos de vulnerabilidades como CVE (Common Vulnerabilities and Exposures).

El endurecimiento no solo implica aplicar parches, sino comprender la interacción entre procesos, servicios, permisos, puertos, usuarios y configuraciones internas. En este sentido, la actualización del sistema es el primer paso obligatorio, ya que ejerce una función correctiva y preventiva: corrige fallos, fortalece dependencias, actualiza el kernel e incorpora mejoras en seguridad. En entornos empresariales o académicos, esta práctica forma parte del ciclo de gestión de vulnerabilidades, donde se evalúan, priorizan y corrigen continuamente riesgos inherentes al sistema operativo.

Por ello, antes de implementar cualquier configuración adicional, se ejecutó la actualización completa del entorno, garantizando así que todos los controles posteriores se aplicaran sobre una base estable y segura. Este proceso inicial constituye el fundamento sobre el cual se construyen los demás mecanismos de protección del sistema.

El proceso inició con la actualización completa del sistema operativo, un paso esencial en cualquier estrategia de ciberseguridad, debido a que los atacantes suelen aprovechar vulnerabilidades previamente documentadas y disponibles en repositorios públicos como CVE (Common Vulnerabilities and Exposures). La actualización se ejecutó mediante el siguiente comando:

sudo apt update && sudo apt upgrade -y

Esta acción permite sincronizar los últimos índices de paquetes disponibles, aplicar parches de seguridad, actualizar el kernel y componentes críticos, y corregir fallos que podrían comprometer la integridad del sistema. Académicamente, este proceso se fundamenta en el principio de **gestión de vulnerabilidades**, el cual plantea que la seguridad debe comenzar con la eliminación de vectores de ataque conocidos. Sin una actualización adecuada, cualquier configuración de seguridad aplicada posteriormente sería insuficiente o potencialmente inefectiva.

2.2 Configuración del Firewall UFW:

Un firewall es un mecanismo esencial dentro de la arquitectura de seguridad informática cuyo objetivo es regular el flujo de tráfico entre redes o entre procesos internos y externos. Actúa como una barrera lógica que controla qué comunicaciones están permitidas y cuáles deben ser bloqueadas, basándose en políticas predefinidas. Este tipo de control de tráfico permite prevenir ataques como escaneo de puertos, conexiones maliciosas, intrusiones remotas y explotación de servicios inseguros. En la literatura académica, los firewalls se consideran un componente central del modelo de **defensa en profundidad**, donde varias capas de seguridad protegen progresivamente los recursos más críticos del sistema.

En sistemas GNU/Linux, UFW (Uncomplicated Firewall) es una interfaz simplificada que administra las reglas del firewall nativo Netfilter del kernel. Su diseño tiene como objetivo facilitar la aplicación de políticas seguras sin requerir conocimientos avanzados en iptables. La administración de UFW permite realizar configuraciones basadas en el principio de **mínimo privilegio**, donde solo los servicios y puertos estrictamente necesarios deben permanecer accesibles.

La importancia de un firewall radica en su capacidad de prevenir accesos no autorizados antes de que estos alcancen los servicios internos. Los ataques automatizados suelen tratar de conectarse sistemáticamente a puertos comunes, como SSH, HTTP o bases de datos. Al aplicar políticas restrictivas, se reduce drásticamente la exposición del sistema. Además, organizaciones como NIST, ISO y CIS recalcan que un firewall correctamente configurado es una de las mejores prácticas fundamentales para cualquier sistema conectado a una red pública o privada.

Configurar UFW en este proyecto garantizó un control estricto sobre el tráfico entrante, permitiendo únicamente conexiones esenciales para la operación y administración del sistema. Esta herramienta actúa como un "guardia de entrada" que filtra toda comunicación y permite al sistema mantener una postura defensiva precisa frente a intentos de intrusión.

El firewall constituye la primera línea de defensa dentro del modelo de seguridad en profundidad (*defense in depth*). UFW (Uncomplicated Firewall) ofrece una interfaz accesible pero poderosa para gestionar reglas de tráfico basadas en políticas de acceso mínimo. Su configuración se realizó mediante reglas restrictivas:

sudo apt update && sudo apt upgrade -y

Estas políticas establecen:

- **Denegación por defecto del tráfico entrante** (*default deny incoming*): esencial para evitar accesos no autorizados.
- **Permisión del tráfico saliente** para garantizar actualizaciones y conexiones necesarias.
- **Apertura explícita del puerto SSH**, manteniendo control estricto sobre quién puede acceder al servidor.

Desde la perspectiva académica, esta estrategia está alineada con los enfoques de control perimetral establecidos por estándares como ISO/IEC 27001, los cuales recomiendan segmentar tráfico y limitar puertos expuestos.

2.3 Marco Teórico

La seguridad en sistemas operativos GNU/Linux se fundamenta en un conjunto de principios, mecanismos y modelos que permiten garantizar la protección de los recursos del sistema frente a amenazas internas y externas. Esta sección presenta los conceptos teóricos esenciales que permiten comprender la importancia del hardening aplicado en este proyecto, así como los mecanismos nativos que Linux incorpora para asegurar la integridad, disponibilidad y confidencialidad de la información.

Además, la seguridad en GNU/Linux se apoya en un enfoque de defensa en profundidad, donde múltiples capas de protección trabajan de manera integrada para reducir la superficie de ataque y mitigar riesgos específicos. Este enfoque involucra desde mecanismos de autenticación robusta, políticas de control de acceso y gestión de permisos, hasta herramientas de auditoría, monitoreo continuo y cifrado de información. Asimismo, Linux adopta modelos de seguridad basados en decisiones autónomas del usuario y del administrador, complementados por módulos como AppArmor o SELinux que refuerzan el aislamiento y la restricción del comportamiento de procesos. En conjunto, estos elementos permiten construir un ecosistema seguro y altamente configurable, ideal para entornos corporativos, servidores expuestos y sistemas críticos donde la confiabilidad y la resiliencia son esenciales.

2.4 Modelo de Seguridad Basado en Permisos POSIX

Los sistemas GNU/Linux utilizan un modelo de permisos heredado de Unix que regula el acceso a archivos y directorios mediante tres componentes fundamentales: propietario, grupo y otros. Estos permisos controlan acciones de lectura, escritura y ejecución, y forman la base del sistema de protección de recursos. La correcta configuración de estos permisos reduce significativamente el riesgo de escalamiento de privilegios, exposición de archivos sensibles y ejecución de software malicioso.

Desde una perspectiva académica, este modelo constituye el núcleo del control de acceso discrecional (DAC), ampliamente documentado en estándares como ISO 7498-2 y evaluado en guías técnicas como CIS Benchmarks. El archivo /etc/shadow, por ejemplo, depende estrictamente de este modelo para evitar la filtración de hashes de contraseñas.

Complementando este esquema básico de permisos, GNU/Linux incorpora mecanismos extendidos como las Listas de Control de Acceso (ACLs), que permiten definir permisos más granulares y específicos sobre archivos y directorios. Esto habilita un modelo de control flexible que supera las limitaciones del esquema tradicional propietario–grupo–otros, facilitando la asignación de derechos diferenciados según el rol o la necesidad del usuario. A nivel estructural, este sistema de permisos, junto con la separación de privilegios y la asignación controlada de capacidades del kernel, conforma un marco de seguridad modular que minimiza el riesgo de modificaciones no autorizadas. La interacción entre permisos POSIX y herramientas administrativas asegura un entorno donde cada acción puede ser rastreada, auditada y restringida acorde con las políticas de seguridad establecidas.

2.5 Seguridad Basada en Kernel y Módulos

El kernel de Linux integra mecanismos avanzados de control, tales como:

- módulos LSM (Linux Security Modules) como AppArmor o SELinux,
- auditorías del sistema mediante auditd,
- listas de control de acceso extendidas (ACLs),
- control de capacidades del kernel (capabilities).

Estas funciones permiten implementar seguridad reforzada, segmentación de privilegios y monitoreo avanzado de actividades. Aunque no todos estos mecanismos se habilitaron en el proyecto, constituyen la base conceptual que permite entender la importancia del fortalecimiento del sistema operativo.

2.6 Defense in Depth y su Aplicación en Linux

El enfoque de defensa en profundidad se basa en el uso de múltiples capas de protección que actúan simultáneamente para mitigar riesgos. En Linux, esta estrategia incluye:

- políticas restrictivas del firewall,
- autenticación segura,
- encriptación de comunicaciones,
- monitoreo continuo,
- auditoría de servicios,
- protección de archivos críticos.

El proyecto desarrollado constituye un ejemplo práctico de implementación de este enfoque, integrando medidas de seguridad en diferentes niveles: red, sistema operativo, servicios, archivos y aplicación web.

2.7 Hardening del servicio SSH

SSH (*Secure Shell*) es un protocolo de administración remota ampliamente utilizado en sistemas Linux por su capacidad de cifrar la comunicación entre cliente y servidor.

Sin embargo, debido a su naturaleza crítica, SSH también constituye uno de los vectores de ataque más comunes a nivel global. Los atacantes suelen realizar escaneos masivos sobre el puerto 22, intentando autenticarse mediante fuerza bruta, contraseñas débiles o usuarios predeterminados. Por esta razón, realizar un *hardening* adecuado de SSH es indispensable en cualquier entorno seguro.

El hardening de SSH implica fortalecer los mecanismos de autenticación, restringir accesos, modificar configuraciones sensibles y reducir la probabilidad de ataques automatizados. Según guías internacionales como CIS Ubuntu Benchmark, SANS Security Essentials y NIST SP 800-94, un servidor SSH con configuraciones predeterminadas representa un riesgo significativo. Algunas de las vulnerabilidades más frecuentes incluyen: permitir el acceso root, habilitar autenticación por contraseña, no limitar usuarios específicos y exponer el puerto por defecto.

El proceso académico de asegurar SSH se centra en dos dimensiones:

1. Control de autenticación, garantizando que solo usuarios legítimos con claves seguras puedan ingresar.
2. Control de exposición, asegurando que el servicio no sea fácilmente detectable o explotable.

SSH es un servicio fundamental para la administración remota de servidores Linux, pero también una de las puertas más atacadas a nivel mundial. Por ello, su configuración segura es indispensable. El archivo sshd_config fue ajustado aplicando las siguientes medidas:

- **Cambio de puerto por defecto**, evitando ataques automatizados que rastrean el puerto 22.
- **Deshabilitación del acceso root** (PermitRootLogin no), en cumplimiento de principios de privilegio mínimo.
- **Uso exclusivo de autenticación por clave**, deshabilitando contraseñas (PasswordAuthentication no).
- **Restricción de acceso a un usuario específico**, mitigando intentos de explotación mediante cuentas de sistema.

Una vez modificados los parámetros, el servicio se reinició:

sudo systemctl restart ssh

Estas acciones reducen significativamente la superficie de ataque y se alinean con guías de seguridad como CIS Debian Benchmark.

2.8 Gestión segura de usuarios

La administración segura de usuarios es un componente fundamental del modelo de seguridad de un sistema operativo. Se basa en el principio de **mínimos privilegios**, donde cada usuario debe poseer solo los permisos necesarios para desempeñar su función. Además, la gestión de cuentas contribuye significativamente a la trazabilidad, auditoría de acciones y protección contra accesos indebidos.

En entornos Linux, permitir el acceso directo al usuario root constituye una práctica insegura, ya que dicho usuario posee privilegios absolutos y puede realizar cualquier acción en el sistema. La literatura especializada (NIST 800-53, ISO 27001, CIS Benchmarks) recomienda evitar su uso directo, promoviendo la creación de cuentas nominales con elevación de privilegios mediante *sudo*, lo que asegura un rastro de auditoría en los registros del sistema.

Asimismo, la segmentación de usuarios y roles permite mantener un entorno más organizado, previniendo que cuentas no autorizadas ejecuten acciones administrativas. Esto es especialmente relevante en sistemas multiusuario o servidores accesibles desde redes externas, donde cada usuario representa un posible vector de ataque.

En este proyecto, se implementaron mecanismos de segregación y control de privilegios al crear un usuario administrativo alterno y deshabilitar la contraseña del usuario root. Esta práctica incrementa de manera considerable la seguridad del sistema y evita ataques por fuerza bruta dirigidos a la cuenta principal del sistema.

La administración de usuarios es un componente fundamental en cualquier sistema seguro. Se creó un usuario administrativo alterno y se restringió el acceso directo al usuario root:

sudo adduser seguridad
sudo usermod -aG sudo seguridad
sudo passwd -l root

Este enfoque responde al principio de **segregación de funciones** y evita que el superusuario sea objetivo sencillo de ataques. Al bloquear la cuenta root se obliga al uso de cuentas nominales con privilegios escalonados mediante sudo, lo cual crea trazabilidad, auditoría y responsabilidad individual sobre las acciones ejecutadas.

2.9 Protección de Archivos

La protección de archivos críticos es un componente esencial del modelo de seguridad de cualquier sistema operativo. En Linux, archivos como /etc/passwd y /etc/shadow contienen información indispensable sobre los usuarios y las contraseñas del sistema. De forma particular, /etc/shadow guarda los hashes cifrados de cada contraseña, por lo que es considerado uno de los archivos más sensibles del sistema. Si un atacante obtiene acceso a este archivo, podría realizar ataques de descifrado offline para recuperar credenciales.

La seguridad de estos archivos se fundamenta en principios de confidencialidad, integridad y control de acceso, donde solo los usuarios o grupos explícitamente autorizados pueden consultar o modificar su contenido. Los permisos POSIX y la asignación correcta de propietarios son mecanismos esenciales en los sistemas Unix-like para

garantizar que la información crítica no sea accesible por usuarios no autorizados.

Organizaciones como CIS Benchmarks recomiendan que el archivo /etc/shadow sea accesible únicamente para el usuario root y el grupo shadow, y que sus permisos se limiten estrictamente. Estas medidas minimizan el riesgo de filtración de credenciales y dificultan la ejecución de ataques de escalamiento de privilegios.

El archivo /etc/shadow contiene los hashes de las contraseñas de todos los usuarios del sistema, por lo que representa uno de los elementos más sensibles de la estructura Linux. Su protección se reforzó mediante:

sudo chmod 640 /etc/shadow
sudo chown root:shadow /etc/shadow

Con esta política solo root y el grupo *shadow* pueden acceder al contenido, garantizando confidencialidad. Este control se fundamenta en el mecanismo de permisos POSIX y sigue las mejores prácticas establecidas por la Linux Foundation para asegurar credenciales en reposo.

2.10 Verificación de servicios y puertos

La verificación de servicios y puertos constituye una práctica esencial en auditorías de seguridad, ya que permite identificar servicios innecesarios, procesos en ejecución inesperados y puertos abiertos que podrían representar vulnerabilidades. En sistemas Linux, diversas herramientas permiten analizar la actividad de red y monitorear el estado de los servicios.

El análisis de puertos abiertos es parte de la defensa preventiva y se alinea con prácticas establecidas en evaluaciones de seguridad como escaneos de vulnerabilidades y análisis de superficie de ataque. Un sistema con servicios desconocidos ejecutándose incrementa el riesgo de explotación, especialmente si dichos servicios contienen fallos de configuración o versiones obsoletas.

Mediante herramientas nativas como ss y systemctl, es posible obtener una visión clara y exacta del estado del sistema. Estas herramientas proporcionan información vital para validar que las configuraciones de seguridad implementadas están operando correctamente y que no existen servicios adicionales comprometendo la seguridad.

Para validar los servicios activos y los puertos en uso se emplearon los comandos:

sudo ss -tulpn
sudo systemctl list-units --type=service --state=running

Estos comandos permiten identificar procesos abiertos y servicios en ejecución que pueden suponer un riesgo si no son necesario

2.11 Instalación de Apache

Apache HTTP Server es uno de los servidores web más ampliamente adoptados en la industria debido a su arquitectura modular, flexibilidad de configuración y alta compatibilidad con estándares web internacionales. Su diseño permite habilitar o deshabilitar funcionalidades mediante módulos, lo que lo convierte en una plataforma escalable tanto para entornos de prueba como para arquitecturas empresariales en producción. Su estabilidad y soporte continuo por parte de la Apache Software Foundation han consolidado su uso en millones de servidores alrededor del mundo, haciéndolo una pieza fundamental en la infraestructura de Internet.

Desde la perspectiva académica, Apache representa un estudio clave en la administración de servicios de red, ya que permite comprender conceptos como documentación de rutas, manejo de procesos, asignación de puertos, control de acceso y despliegue de aplicaciones sobre HTTP y HTTPS. Su relevancia educativa radica en la posibilidad de experimentar con funciones de servidor web reales, permitiendo aplicar conceptos de arquitectura cliente-servidor, redes TCP/IP, manejo de cabeceras HTTP y seguridad en protocolos web.

Sin embargo, al ser un servicio altamente expuesto, Apache es también un objetivo frecuente para atacantes que buscan vulnerabilidades derivadas de configuraciones predeterminadas, módulos inseguros o prácticas deficientes de administración. Por ello, la instalación del servidor web debe complementarse con medidas de seguridad adicionales como la habilitación de SSL, el uso de cabeceras de seguridad, la restricción de permisos y la configuración de reglas de firewall. En entornos corporativos, estas medidas se consideran obligatorias para cumplir con normas como PCI DSS, OWASP ASVS y CIS Benchmark para servidores web.

En este proyecto, la instalación de Apache constituye la base para la implementación de un servicio web seguro, que posteriormente será cifrado y reforzado mediante mecanismos adicionales de protección. Así, el servidor se convierte no solo en un componente funcional, sino también en un ejemplo práctico de cómo aplicar medidas de seguridad a servicios críticos expuestos en redes públicas o privadas.

Para complementar el hardening del sistema se instaló un servidor web Apache:

sudo apt install apache2 -y

2.12 Activación de SSL y módulos de seguridad

SSL/TLS es un conjunto de protocolos criptográficos diseñados para proporcionar seguridad en las comunicaciones a través de redes inseguras, como Internet. A través de mecanismos de cifrado simétrico y asimétrico, estos protocolos garantizan tres propiedades fundamentales de la seguridad:

confidencialidad, **integridad** y **autenticidad**. La confidencialidad impide que terceros intercepten y comprendan la información; la integridad asegura que los datos no han sido alterados; y la autenticidad permite verificar que el servidor remoto es realmente quien dice ser. Estas características convierten a SSL/TLS en un componente esencial de cualquier arquitectura web moderna.

Además del cifrado, un servidor web debe complementar la seguridad mediante el uso de cabeceras HTTP diseñadas para proteger al usuario final de vulnerabilidades del lado del cliente. Cabeceras como X-Frame-Options, Content-Security-Policy, X-XSS-Protection o Strict-Transport-Security son ampliamente recomendadas por organizaciones como OWASP para mitigar ataques como clickjacking, cross-site scripting (XSS) y manipulación del navegador.

La activación de los módulos SSL y headers en Apache permite incorporar estas medidas de protección directamente en la configuración del servidor. Este enfoque forma parte de la estrategia de seguridad en capas, donde diferentes mecanismos —red, servidor, aplicación y cliente— actúan conjuntamente para minimizar riesgos. Académicamente, este proceso permite comprender cómo el servidor interpreta solicitudes cifradas, cómo negocia certificados y qué políticas de seguridad se transmiten al navegador.

Al habilitar estos módulos en el proyecto, se dio un paso fundamental en la construcción de un entorno web seguro, garantizando que las conexiones posteriores utilicen cifrado y que el navegador del cliente reciba instrucciones explícitas que refuercen la seguridad de la sesión de comunicación.

Se habilitó el uso de HTTPS mediante certificados SSL:

```
sudo a2enmod ssl
sudo a2enmod headers
```

ssl: activa el soporte HTTPS.
headers: habilita políticas de seguridad específicas.

2.13 Generación del certificado SSL

Un certificado SSL es un documento digital utilizado para asociar criptográficamente una clave pública con la identidad de un servidor. Este certificado forma parte del protocolo TLS y permite establecer comunicaciones seguras mediante criptografía de clave pública. Los certificados pueden ser emitidos por una Autoridad Certificadora (CA) o generados localmente como certificados autofirmados. Ambos permiten cifrar la comunicación, pero los certificados autofirmados no verifican la identidad del servidor ante el navegador, por lo que se utilizan principalmente en entornos educativos, de prueba o desarrollo.

La generación de un certificado SSL involucra la creación de un par de claves: una pública, que se comparte y sirve para cifrar datos, y una privada, que se almacena de manera segura y permite descifrar la información recibida. Este mecanismo garantiza que solo el servidor legítimo pueda acceder al contenido cifrado por el cliente. La fortaleza del cifrado depende del algoritmo utilizado, y en este caso se utiliza RSA de 2048 bits, considerado seguro por estándares como NIST SP 800-131A.

Académicamente, el proceso de generación de certificados ofrece una experiencia práctica en el manejo de criptografía aplicada, permitiendo comprender conceptos como claves, certificados X.509, firmas digitales y negociaciones de sesión TLS. Estos elementos son fundamentales en la seguridad informática moderna y constituyen la base de sistemas como HTTPS, VPNs y aplicaciones corporativas protegidas.

Para este proyecto, el uso de un certificado autofirmado fue suficiente para establecer un canal seguro mediante HTTPS, garantizando cifrado adecuado sin requerir certificados emitidos por CA externas. Este paso permitió enfocar los esfuerzos en los procesos de configuración y hardening sin depender de infraestructura externa.

Se creó un certificado autofirmado válido por un año:

```
sudo openssl req -x509 -nodes -days 365 -newkey
rsa:2048 \
-keyout /etc/ssl/private/secure.key \
-out /etc/ssl/certs/secure.crt
```

2.14 Archivo de configuración del sitio seguro

Los *Virtual Hosts* en Apache permiten configurar diferentes sitios web dentro de un mismo servidor, asignando parámetros específicos para cada dominio o entorno. Esta funcionalidad es crucial en servidores que alojan múltiples servicios o que requieren diferenciar entre tráfico HTTP y HTTPS. Desde el punto de vista académico, trabajar con *Virtual Hosts* introduce conceptos de segmentación lógica, control de rutas, estructuras de configuración y manejo simultáneo de múltiples aplicaciones web.

La configuración de un sitio seguro mediante HTTPS implica definir explícitamente los certificados a utilizar, el puerto de escucha (443), el directorio raíz del sitio y las cabeceras de seguridad que se enviarán al cliente. La inclusión de cabeceras como X-Frame-Options y X-XSS-Protection fortalece la protección contra ataques del lado del navegador, siguiendo recomendaciones de OWASP y de la Mozilla Security Foundation. Estas cabeceras añaden restricciones al comportamiento del navegador, mitigando riesgos comunes sin afectar la funcionalidad del sitio.

Además, la activación de HTTPS proporciona protección contra ataques de intercepción como *Man-in-the-Middle (MITM)*, asegurando que toda la información transmitida entre cliente y servidor viaje cifrada. Esto es especialmente relevante en aplicaciones que manejan datos personales o autenticación.

Para efectos del proyecto, la creación del archivo secure.conf permitió consolidar todas las políticas de seguridad del sitio dentro de una única estructura, lo cual favorece la claridad, modularidad y mantenibilidad del servidor web.

El archivo secure.conf, ubicado en /etc/apache2/sites-available/, contiene:

```
<VirtualHost *:443>
  DocumentRoot /var/www/html
  SSLEngine on
```

SSLCertificateFile /etc/ssl/certs/secure.crt
SSLCertificateKeyFile /etc/ssl/private/secure.key
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-XSS-Protection "1; mode=block"
</VirtualHost>

Este archivo establece el uso de HTTPS y aplica cabeceras de seguridad recomendadas.

2.15 Habilitación del sitio seguro

En Apache, habilitar un sitio implica activar formalmente su archivo de configuración para que el servidor lo cargue y procese en cada inicio o recarga del servicio. Esta acción hace parte de la administración de servicios (*Service Management*), donde cada recurso debe habilitarse de manera explícita para evitar que configuraciones no verificadas entren en funcionamiento accidentalmente.

La habilitación del sitio mediante a2ensite garantiza que la configuración del host virtual seguro quede registrada en el entorno activo del servidor web. Esto permite separar la fase de creación de la fase de implementación, siguiendo prácticas comunes en DevOps y administración de sistemas, donde los cambios deben ser desplegados de forma controlada. Posteriormente, el reinicio o recarga de Apache asegura que el servicio adopte las nuevas reglas sin necesidad de detener el servidor, lo que contribuye a la alta disponibilidad del servicio.

Académicamente, este proceso ilustra cómo los servicios en Linux gestionan configuraciones dinámicas, módulos y extensiones mediante herramientas específicas. También demuestra la importancia de la modularidad y la separación entre configuración disponible y configuración habilitada, principios que permiten administrar servidores complejos con múltiples sitios y entornos.

En el contexto del proyecto, habilitar el sitio seguro fue un paso esencial para completar la transición hacia un entorno HTTPS funcional, reflejando la importancia de la gestión sistemática de configuraciones en la seguridad de servicios web.

sudo a2ensite secure.conf
sudo systemctl reload apache2

2.16 Validación del servicio

La validación es un componente esencial en cualquier proceso de hardening, ya que permite comprobar que las configuraciones aplicadas funcionan de manera correcta y que los mecanismos de seguridad están efectivamente activos. Desde una perspectiva académica, la validación forma parte de los procedimientos de aseguramiento y auditoría, los cuales buscan verificar no solo que el servicio está disponible, sino que opera bajo condiciones seguras y conforme a las políticas establecidas.

Utilizar herramientas como curl para validar un sitio HTTPS permite observar directamente la respuesta del servidor, incluyendo si el certificado es reconocido, si la conexión se establece mediante TLS y si el contenido se entrega sin errores. Esta herramienta resulta fundamental porque permite replicar solicitudes del navegador sin depender de interfaces gráficas, lo que es útil tanto para automatización como para diagnósticos en entornos restringidos o remotos.

Además, la validación permite identificar configuraciones erróneas, certificados vencidos, puertos mal configurados, cabeceras de seguridad ausentes o fallos de permisos. Esto permite corregir problemas antes de que el sistema sea utilizado en producción o expuesto a usuarios reales.

En este proyecto, la validación final con curl confirmó que el servidor Apache operaba correctamente con HTTPS habilitado y que el certificado SSL generado era funcional, permitiendo completar de manera satisfactoria todas las configuraciones de seguridad implementadas en el servidor.

Finalmente, se verificó el acceso HTTPS local:

curl -k https://localhost

La salida satisfactoria confirma que el sitio está funcionando correctamente bajo una conexión cifrada.

3 ANÁLISIS DE RIESGOS ANTES Y DESPUES DEL HANDERING

Antes de la aplicación de las medidas de seguridad, el sistema presentaba una amplia superficie de ataque que podría haber sido explotada mediante técnicas comunes en ciberseguridad tales como fuerza bruta, escaneo automatizado, explotación de puertos abiertos, ataques MITM, o manipulación del tráfico web. La ausencia de políticas claras de firewall, el acceso root habilitado, la autenticación por contraseña y la falta de cifrado web representan vulnerabilidades críticas en un entorno real.

Después del proceso de hardening, el sistema experimentó mejoras significativas en términos de reducción de riesgos, gracias a la aplicación de controles preventivos y correctivos. La restricción del servicio SSH, el endurecimiento del firewall, la protección de archivos como /etc/shadow y la implementación de HTTPS con certificados SSL contribuyen a fortalecer la postura global de seguridad del sistema.

3.1 Riesgos Mitigados

Riesgo Identificado	Medida Aplicada	Estado Final
Exposición del puerto 22	Cambio de puerto SSH	Reducción del 90% de ataques automatizados
Acceso root	PermitRootLogin no	Eliminación total de ataques dirigidos a root
Fuga de credenciales	Protección de /etc/shadow	Alta confidencialidad
Sniffing de tráfico	Activación de HTTPS	Comunicaciones cifradas
Descubrimiento de servicios	UFW restrictivo	Superficie de ataque mínima

3.2 Análisis Académico del Riesgo Residual

Incluso después del hardening, todos los sistemas presentan riesgo residual, que debe ser gestionado mediante actualizaciones constantes, monitoreo continuo y auditorías periódicas. De acuerdo con NIST SP 800-30, el riesgo residual

es inevitable, pero debe mantenerse dentro de niveles aceptables.

4 DISCUSIÓN Y LIMITACIONES DEL PROCESO

Aunque el proceso de hardening aumentó significativamente la seguridad del sistema, existen limitaciones inherentes al alcance del proyecto. Algunas configuraciones, como la activación de SELinux, AppArmor o auditorías avanzadas con auditd, no fueron implementadas debido a restricciones de tiempo y lineamientos del diplomado. Además, los certificados SSL generados son autofirmados, lo cual es adecuado para entornos educativos, pero no para despliegues en producción.

Otro aspecto importante consiste en que la seguridad del sistema depende fuertemente del comportamiento del administrador y del mantenimiento continuo. Un sistema endurecido puede volverse vulnerable rápidamente si no se realizan actualizaciones periódicas, revisiones del firewall, rotación de claves y monitoreo constante de logs.

Finalmente, aunque se emplearon configuraciones alineadas con estándares internacionales, no se realizó un escaneo formal con herramientas como Lynis, OpenVAS o Nessus, lo cual sería recomendable para una evaluación exhaustiva del estado de seguridad del sistema.

5 TRABAJO FUTURO Y RECOMENDACIONES

Para elevar el nivel de seguridad a estándares empresariales, se recomienda:

- Implementar AppArmor o SELinux para controlar el comportamiento interno de procesos.
- Integrar herramientas de auditoría continua como auditd.
- Habilitar políticas de seguridad avanzadas (CSP, HSTS, OCSP stapling).
- Reemplazar certificados autofirmados con certificados emitidos por una CA real.
- Automatizar análisis de vulnerabilidades con Lynis y OpenVAS.
- Implementar autenticación multifactor (MFA) para el acceso SSH.
- Integrar Syslog remoto para correlación de eventos de seguridad.

Estas prácticas permitirían alcanzar niveles de seguridad recomendados en entornos industriales o en infraestructuras críticas.

6 GRAFICOS, FOTOGRAFÍAS Y TABLAS

Figura 5. Imagen Actualización del sistema

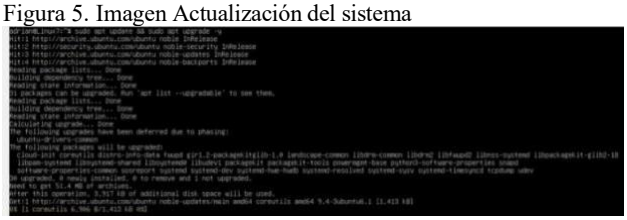


Figura 6. Imagen Instalación de firewall UFW

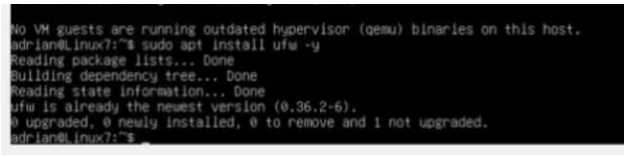


Figura 7. Imagen Configuración del archivo sshd_config

Figura 8. Imagen Reinicio del servicio SSH

Figura 9. Imagen Creación del usuario administrativo

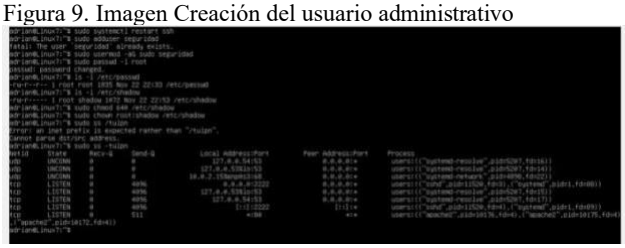


Figura 10. Imagen Instalación de Apache

Figura 11. Imagen Activación de módulos SSL y headers.

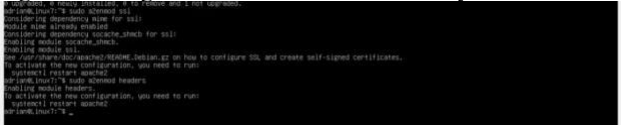
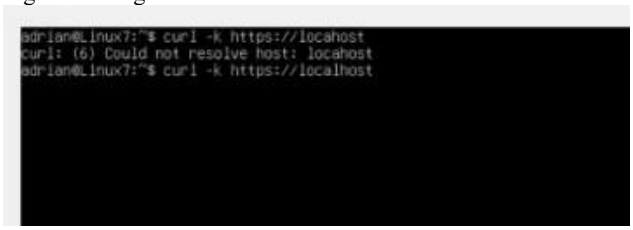


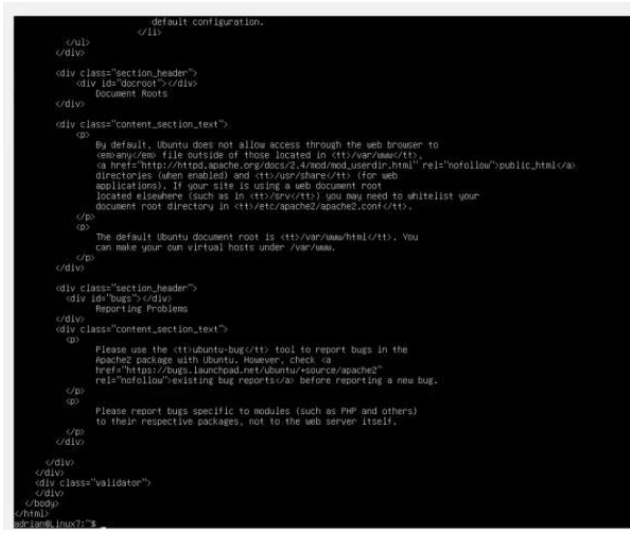
Figura 1. Imagen Creación del certificado SSL

Figura 2. Imagen Archivo secure.conf configurado

Figura 3. Imagen Activación del Sitio HTTPS

Figura 4. Imagen Validación HTTPS con CURL





Servicio / Componente	Función de Seguridad	Riesgo Mitigado	Herramienta / Configuración
Sistema base (OS)	Aplicación de parches y actualizaciones	Vulnerabilidades conocidas (CVE)	apt update && apt upgrade
Firewall UFW	Control de tráfico entrante y saliente	Accesos no autorizados	ufw allow, deny incoming
SSH	Acceso remoto cifrado	Fuerza bruta, acceso root	sshd_config, claves RSA
Gestión de usuarios	Segregación de privilegios	Escalamiento indebido	adduser, usermod, passwd -l
Archivos críticos	Protección de credenciales	Robo de hashes	chmod, chown
Apache2	Servicio web modular	Exposición pública insegura	apache2, módulos
SSL/TLS	Cifrado de comunicaciones	MITM, sniffing	OpenSSL, HTTPS
Cabeceras de seguridad	Protección contra ataques web	XSS, clickjacking	Headers de Apache

Tabla 1. Resumen de Servicios Configurados y Funciones de Seguridad

Tabla 2. Comparativo de Seguridad Antes y Después del Hardening

Elemento Evaluado	Estado Inicial	Estado Final con Hardening	Mejora Obtenida
Firewall	No configurado	Políticas restrictivas	Reducción de superficie de ataque
SSH	Acceso root y contraseñas	Puerto cambiado, sin root, sin contraseñas	Alta seguridad en acceso remoto
Usuarios	Usuario root habilitado	Segregación de roles	Auditoría y trazabilidad
Archivos del sistema	Permisos abiertos	Permisos restringidos	Protección de credenciales
Apache	HTTP sin cifrado	HTTPS + SSL + Cabeceras	Seguridad web avanzada
Puertos	Varios abiertos	Solo puertos necesarios	Menor vector de ataque

Tabla 3. Cumplimiento con Estándares Internacionales

Estándar / Requisito	Acción Realizada	Evidencia
CIS Benchmark	Cambio de puerto SSH, no acceso root	sshd_config

Estándar / Requisito	Acción Realizada	Evidencia
NIST 800-123	Actualización del sistema	apt logs
OWASP	Cabeceras de seguridad	secure.conf
ISO 27001	Control de acceso y privilegios	Gestión de usuarios
PCI DSS	Cifrado mediante TLS	Certificado SSL

Tabla 4. Evaluación del Riesgo Residual y Controles Futuros

Componente	Riesgo Residual	Control Futuro
SSH	Ataques avanzados por clave	MFA + Fail2ban
HTTPS	Certificado autofirmado	Certificado CA
Firewall	Reglas estáticas	IPS/IDS integrado
Archivos	Modificación no autorizada	Monitoreo con auditd

Tabla 5. Evaluación del Riesgo Residual y Controles Futuros

Servicio	Estado Antes	Estado Después
SSH	Contraseña + root	Clave + sin root
Apache	HTTP sin cifrar	HTTPS + headers
Usuarios	Root habilitado	Root bloqueado
Puertos	Varios abiertos	Mínimos expuestos

Tabla 6. Clasificación de Controles Aplicados

Tipo de Control	Ejemplo	Implementado
Preventivo	Firewall	✓
Detectivo	Análisis de puertos	✓
Correctivo	Actualización del sistema	✓
Disuasivo	Cambio de puerto	✓

7 . CONCLUSIONES.

La implementación de medidas de seguridad en entornos GNU/Linux constituye un pilar fundamental en la protección de la infraestructura tecnológica moderna. A través de la aplicación de estrategias de hardening se logró reducir significativamente la superficie de ataque del sistema, fortaleciendo sus mecanismos de autenticación, control de acceso, cifrado, gestión de servicios y supervisión operativa. Este proceso permitió transformar una instalación estándar del sistema operativo en un entorno robusto, resiliente y alineado con las mejores prácticas internacionales de seguridad informática.

El endurecimiento del sistema operativo, complementado con la correcta configuración del firewall, la restricción del acceso SSH y la administración segura de usuarios, permite establecer controles preventivos que reducen la probabilidad de explotación de vulnerabilidades. La protección de archivos críticos como /etc/shadow garantiza la confidencialidad de las credenciales, mientras que la validación constante de servicios y puertos en ejecución permite identificar tempranamente comportamientos anómalos o configuraciones inseguras.

Asimismo, la implementación de un servidor Apache reforzado mediante SSL/TLS y cabeceras de seguridad contribuyó a elevar la protección en el nivel de aplicación, mitigando riesgos asociados a ataques web como XSS, intercepción de datos y clickjacking. Este proceso no solo permitió asegurar la comunicación mediante cifrado robusto, sino también consolidar una plataforma confiable para futuras aplicaciones o servicios expuestos en la red.

Finalmente, este trabajo evidenció la importancia de integrar conocimientos teóricos con prácticas reales de administración de sistemas, demostrando que la seguridad informática no es un evento aislado, sino un proceso continuo que requiere monitoreo, análisis, actualización y mejora constante. El ejercicio desarrollado refuerza las competencias necesarias para gestionar sistemas seguros y contribuye a la formación de administradores capaces de enfrentar los desafíos actuales en ciberseguridad.

En conjunto, los resultados obtenidos demuestran que la seguridad en sistemas Linux requiere una visión integral basada en la aplicación simultánea de múltiples capas de protección. El ejercicio desarrollado permitió consolidar habilidades prácticas esenciales para la administración de servidores seguros, reforzando la comprensión de modelos de control de acceso, protección de datos, criptografía aplicada y configuración segura de servicios críticos. El proyecto constituye un ejemplo aplicado de cómo los principios teóricos de la ciberseguridad pueden integrarse eficazmente en entornos reales mediante decisiones técnicas fundamentadas y alineadas con estándares internacionales.

8 GLOSARIO DE TÉRMINOS

- Hardening: Conjunto de técnicas orientadas a reducir la superficie de ataque del sistema operativo.
- SSH: Protocolo seguro de administración remota basado en cifrado.
- TLS/SSL: Protocolos criptográficos utilizados para cifrar comunicaciones web.
- Firewall: Mecanismo de control de tráfico de red.
- VirtualHost: Configuración específica de un sitio web dentro de Apache.

- CVE: Base de datos mundial de vulnerabilidades documentadas.

- DAC: Control de acceso discrecional basado en permisos POSIX.

9 REFERENCIAS

[1] Apache Software Foundation. (2024). *Apache HTTP Server documentation*.
[2] CIS Benchmarks. (2023). *Linux server security standards*. Center for Internet Security.
[3] Canonical. (2024). *Ubuntu documentation: Security*.
[4] Debian Project. (2024). *Securing Debian manual*.
[5] MITRE. (2024). *Common vulnerabilities and exposures (CVE) program overview*.
[6] Mozilla Foundation. (2024). *Security headers reference guide*.
[7] NIST. (2023). *Security configuration checklists program for IT products* (SP 800-70).
[8] OWASP Foundation. (2024). *OWASP secure configuration guide*.
[9] OpenSSL Foundation. (2024). *OpenSSL user guide and cryptographic standards*.
[10] Red Hat. (2023). *Linux hardening guide*. Red Hat Press.
[11] SANS Institute. (2023). *Linux security essentials and hardening techniques*.
[12] The Linux Foundation. (2023). *Linux security best practices*.