Hacking Ético a Servicios Institucionales

Carlos Josseph Rodríguez Hernández

Trabajo de Grado para Optar al Título de Ingeniero de Sistemas

Director

John William Vásquez Capacho

Doctor en Ingeniería

Tutor

Pablo Josue Rojas Yepes

Master in science

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de ingeniería de Sistemas e Informática

Ingeniería de Sistemas

Bucaramanga

2025

Agradecimientos

El presente trabajo no habría sido posible sin el apoyo y la colaboración de diversas personas e instituciones que contribuyeron de manera significativa durante todas las etapas de su desarrollo.

En primer lugar, expreso mi más sincero agradecimiento a la Universidad Industrial de Santander (UIS), por brindarme la formación académica, los recursos y el ambiente propicio para el aprendizaje, la investigación y el crecimiento profesional.

Agradezco especialmente a la Dirección de Tecnologías de la Información y las Comunicaciones (DTIC) por permitir y facilitar el acceso a los entornos necesarios para la realización de las pruebas, así como por su disposición y colaboración constante en el marco de este proyecto. También extiendo mi gratitud a todos los docentes, tutores y compañeros que, con sus orientaciones, conocimientos, sugerencias y palabras de aliento, enriquecieron el proceso de desarrollo de esta investigación.

Por último, me gustaría agradecer especialmente a Pablo, ya que sin él me hubiera rendido en el camino.

Finalmente, agradezco a todas aquellas personas que, de manera directa o indirecta, aportaron a la concreción de este trabajo, ya sea mediante apoyo técnico, moral o logístico. A todos ustedes, gracias por hacer posible la culminación de este esfuerzo académico.

Tabla de Contenidos

Lista de figuras

Lista de tablas

Resumen

Título: Hacking Ético a Servicios Institucionales*

Autor: Carlos Josseph Rodríguez Hernández**

Palabras Clave: Hacking ético, Ciberseguridad, Seguridad de la información.

Descripción: Este documento titulado "Hacking Ético a Servicios Institucionales" presenta un ejercicio estructurado de evaluación de seguridad realizado a cinco objetivos dentro de un entorno institucional. El propósito principal fue identificar debilidades técnicas que puedan comprometer la confidencialidad, integridad o disponibilidad de los servicios analizados.

Para el desarrollo del ejercicio se empleó la metodología FPTM (Focalized Penetration Testing Methodology), una adaptación más concreta y dirigida de la metodología PTES, que permite un enfoque practico y sistemático sobre los activos evaluados. Esta metodología guio cada una de las etapas del proceso, desde la recopilación de información hasta la entrega del reporte técnico.

El contenido del documento se estructura en seis capítulos. En el primer, Recopilación de Información, se identificaron detalles públicos sobre los sistemas, servicios y posibles vectores de entrada. Posteriormente, en el capítulo de Análisis de Vulnerabilidades, se examinaron los datos recolectados para identificar posibles fallas de seguridad en la superficie de ataque. A continuación, en la sección de Exploración de Vulnerabilidades, se llevaron a cabo pruebas controladas para confirmar la existencia y el impacto real de dichas fallas, el Consolidado de Vulnerabilidades agrupa y clasifica las vulnerabilidades detectadas por criticidad y tipo de riesgo. El capítulo de Reporte y Sugerencias de Mitigación presenta una visión general de los hallazgos, Finalmente, en el capítulo de Conclusiones, se reflexiona sobre los resultados del proceso, destacando las lecciones aprendidas y las recomendaciones clave para mitigar los riesgos identificados.

Este ejercicio no solo permitió evidenciar deficiencias técnicas, sino también generar conciencia sobre la importancia de implementar controles de seguridad adecuados. El conocimiento adquirido a través de este trabajo contribuye al fortalecimiento de la postura de seguridad institucional y refuerza la necesidad de realizar auditorías periódicas.

---

*Trabajo de grado

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: John William Vásquez Capacho. Dr. En Ingeniería.

Abstract

Title: Ethical Hacking of Institutional Services

Author: Carlos Josseph Rodríguez Hernández**

Key Words: Ethical Hacking, Cybersecurity, Information Security.

Description: This document, entitled *"Ethical Hacking of Institutional Services"*, presents a structured security assessment conducted on five targets within an institutional environment. The primary objective was to identify technical weaknesses that could compromise the confidentiality, integrity, or availability of the analyzed services.

The assessment was carried out using the FPTM (Focalized Penetration Testing Methodology), a more focused and practical adaptation of the PTES framework, which provides a systematic approach to the evaluated assets. This methodology guided each stage of the process, from gathering information to the delivery of the technical report.

The content of the document is organized into six chapters. The first chapter, *Information Gathering*, identifies publicly available details about systems, services, and potential entry points. The *Vulnerability Analysis* chapter examines collected data to identify possible security flaws across the surface of the attack. Next, the *Vulnerability Exploration* chapter includes controlled testing to confirm the existence and real impact of those flaws. The *Vulnerability Consolidation* chapter classifies and prioritizes the detected vulnerabilities according to severity and risk type.

The *Reporting and Mitigation Suggestions* chapter provides an overview of the findings, followed by the *Conclusions* chapter, which reflects on the outcomes of the process, highlighting key lessons learned and recommendations for mitigating identified risks.

This exercise not only revealed technical deficiencies but also raised awareness of the importance of implementing proper security controls. The knowledge gained through this work contributes to strengthening the institutional security posture and reinforces the need for periodic audits.

---

*Degree Work

** Faculty of Physicomechanical Engineering. School of Systems and Computer Engineering. Director: John William Vásquez Capacho, PhD in Engineering

Advertencia y descargo de responsabilidad

El presente documento tiene fines estrictamente académicos y de investigación, en el marco del desarrollo de un proyecto de grado orientado al análisis de seguridad informática en entornos institucionales. Toda la información, ejemplos, técnicas y herramientas mencionadas han sido aplicadas únicamente en sistemas autorizados, con el debido consentimiento de las partes involucradas y en un entorno controlado. En ninguna circunstancia se llevaron a cabo acciones que comprometieran la integridad, confidencialidad o disponibilidad de servicios o infraestructuras ajenas a las previamente autorizadas. Se reitera que no se causó daño alguno a sistemas sin el consentimiento expreso de los responsables de los mismos.

Asimismo, el conocimiento técnico expuesto a lo largo de este documento tiene como propósito principal fortalecer la comprensión sobre la seguridad de los sistemas informáticos, promoviendo buenas prácticas y la adopción de medidas preventivas. El autor no se hace responsable del uso indebido, malintencionado o ilegal que terceros puedan hacer de la información contenida en este trabajo

Se exhorta a los lectores a actuar con responsabilidad, ética y dentro del marco legal vigente en sus respectivas jurisdicciones, recordando que la seguridad informática debe orientarse siempre hacia la protección y mejora de los sistemas, no hacia su explotación indebida.

Por último, se le recuerda al lector que toda la información real obtenida en este trabajo está a disposición únicamente de la DTIC, la cual es la responsable de manejar la información de los objetivos evaluados.

Introducción

En la actual era digital, donde el comercio electrónico y la interacción en línea dominan el panorama global, las organizaciones —incluidas las instituciones educativas— han depositado cantidades ingentes de datos en la web. Sin embargo, este entorno de oportunidades también alberga una sombra preocupante: la escasa conciencia en torno a la seguridad de la información. Numerosas entidades, sin importar su tamaño o reputación, han subestimado la astucia de los ciberdelincuentes, dejando sus sistemas expuestos y repletos de vulnerabilidades. Esta omisión representa un riesgo tanto para las instituciones como para los usuarios que confían en sus plataformas.

La Universidad Industrial de Santander (UIS), con su creciente presencia digital, no es la excepción y se convierte en un objetivo potencial para actores maliciosos. Dentro de este contexto, el hacking ético se erige no como una opción, sino como una obligación estrategia para identificar y documentar fallas de seguridad antes de que sean explotadas. A diferencia de las auditorías puramente automáticas, las evaluaciones de hacking ético combinan análisis manual, pensamiento creativo y el uso de herramientas especializadas, lo que permite descubrir fallos lógicos y errores de configuración que las soluciones automatizadas suelen pasar por alto. Esta práctica no solo mitiga vulnerabilidades; también fomenta una cultura de seguridad proactiva y consciente.

Así surge la pregunta que guía este trabajo:

¿Cómo puede el hacking ético identificar y documentar las vulnerabilidades presentes en los sitios web de la Universidad Industrial de Santander, fortaleciendo su seguridad y la confianza de su comunidad universitaria?

Responder a esta interrogante es indispensable, pues las consecuencias de una brecha incluyen desde pérdidas financieras hasta daños reputacionales significativos. Al integrar de forma sistemática pruebas manuales y automatizadas, el proyecto aspira a consolidar un

enfoque integral que fortalezca la postura defensiva de la UIS y sirva como referente para otras instituciones educativas ante las amenazas cibernéticas contemporáneas.

Presentación del Proyecto

El presente trabajo de grado titulado" Hacking Ético a Servicios Institucionales" tiene como objetivo principal realizar pruebas de seguridad en los sistemas crítico de la Universidad Industrial de Santander, con el fin de identificar y remediar vulnerabilidades presentes en su infraestructura tecnológica.

La modalidad de este trabajo es Practica Empresarial, en la cual el autor ha aplicado conocimientos adquiridos durante su formación académica en ingeniería de sistemas,

especialmente en el campo de la ciberseguridad y el hacking ético, para garantizar la protección de los activos informáticos de la universidad.

Este proyecto se desarrolla bajo la dirección de John William Vásquez PhD y cuenta con la supervisión del magister Pablo Josue Rojas Yepes, quien ha brindado orientación en aspectos clave para el desarrollo de este.

La entidad interesada en este proyecto es la Universidad Industrial de Santander, la cual busca reforzar sus políticas de seguridad informática y proteger los datos sensibles almacenados en sus sistemas institucionales.

A través de este trabajo, se espera contribuir al fortalecimiento de la seguridad en los servicios institucionales y establecer un marco de buenas prácticas que pueda ser implementado en futuros proyectos de seguridad dentro del ´ámbito universitario.

1. Objetivos

1.1 Objetivo general

Realizar un estudio evaluativo de seguridad en las páginas web de servicios institucionales empleando pruebas de impacto, riesgo y usabilidad de las vulnerabilidades en

concordancia con los principios de hacking ético.

1.2 Objetivos específicos

- Identificar y analizar 5 servicios institucionales seleccionados para determinar áreas críticas de seguridad que requieran atención inmediata y prioritaria.
- Diseñar pruebas manuales de penetración que simulen posibles escenarios de ataques, empleando técnicas de explotación utilizadas por ciber-atacantes, con el fin de evaluar la resiliencia de los sistemas frente a amenazas potenciales.
- Implementar un enfoque integral de evaluación de seguridad que combine pruebas automatizadas y manuales, con el propósito de identificar y analizar de manera exhaustiva las vulnerabilidades presentes en los servicios institucionales seleccionados.
- Elaborar un informe detallado que documente las vulnerabilidades identificadas durante el proceso de evaluación, junto con recomendaciones y medidas de mitigación específicas, en caso de que se requiera una solución concreta para abordar las vulnerabilidades encontradas.

Para garantizar el cumplimiento de los objetivos dentro de nuestro documento se creó la siguiente tabla 1 con el fin de dar claridad sobre que capítulos corresponden a que objetivo:

Tabla 1:

Correspondencia entre secciones del documento y objetivos alcanzados

Sección	**Objetivo(s) que se cumplen**
Cap. 6 - 14	Objetivo General
Cap. 7	Objetivo Especifico 1
Cap. 8	Objetivo Especifico 2

Cap. 9 y 10	Objetivo Especifico 3
Cap. 11 y 12	Objetivo Especifico 4
Cap. 13 y 14	Conclusiones y Trabajo futuro

2. Marco de Referencia

El hacking ético se ha consolidado como una disciplina esencial dentro del ámbito de la seguridad informática, ofreciendo a las organizaciones una herramienta proactiva para mejorar sus defensas ante un panorama de amenazas cibernéticas en constante evolución.

2.1 Common Weakness Enumeration (CWE)

El *Common Weakness Enumeration* (CWE) es una taxonomía mantenida por MITRE cuyo propósito es identificar, clasificar y describir debilidades comunes en software y sistemas informáticos (CWE - *Common Weakness Enumeration*, s. f.). Esta clasificación estandarizada permite a investigadores, desarrolladores y profesionales de la seguridad comprender mejor las causas subyacentes de vulnerabilidades explotables por actores maliciosos.

A diferencia del *Common Vulnerabilities and Exposures* (CVE), que cataloga vulnerabilidades específicas observadas en sistemas concretos, el CWE se enfoca en patrones genéricos de diseño, implementación o arquitectura que pueden resultar en fallos de seguridad (Scarfone & Mell, 2007). Esta orientación lo convierte en una herramienta tanto técnica como pedagógica para fomentar el desarrollo de software seguro desde etapas tempranas del ciclo de vida. Cada entrada en la base de datos CWE está identificada por un número único y acompañada de información detallada que incluye descripciones, ejemplos de código vulnerable, técnicas de detección, y estrategias de mitigación. Por ejemplo, la entrada CWE-434: Unrestricted Upload of File with Dangerous Type hace referencia a la posibilidad de que un sistema acepte la carga de archivos sin validación adecuada, lo cual puede permitir

la ejecución de código arbitrario en el servidor.

La incorporación del CWE como referencia en auditorias y procesos de análisis de seguridad permite estandarizar los hallazgos, mejorar la comunicación entre equipos técnicos y proporcionar trazabilidad en la documentación de riesgos. En este documento, cada vulnerabilidad identificada ha sido asociada con su correspondiente identificador CWE, a fin de situar los hallazgos dentro de un marco ampliamente aceptado por la comunidad de ciberseguridad.

2.2 Common Vulnerability Scoring System (CVSS)

El Common Vulnerability Scoring System (CVSS) es un estándar abierto para la evaluación y cuantificación de la severidad de vulnerabilidades de seguridad en sistemas informáticos (*CVSS v4.0 Specification Document*, s. f.). Desarrollado inicialmente por el *Forum of Incident Response and Security Teams* (FIRST), su objetivo es proporcionar una métrica objetiva y reproducible que facilite la priorización de vulnerabilidades en función de su impacto técnico y del contexto operativo en el que se encuentran.

CVSS permite asignar a cada vulnerabilidad una puntuación numérica que varía de 0.0 (sin impacto) a 10.0 (crítico), basada en tres métricas principales: la base (Base), que representa la gravedad inherente de la vulnerabilidad; el vector temporal (Temporal), que considera factores como la disponibilidad de *exploits* o soluciones; y el vector ambiental (*Environmental*), que toma en cuenta el contexto especifico del entorno donde reside el sistema afectado.

El uso de CVSS se ha convertido en una práctica común en la gestión de vulnerabilidades, ya que proporciona una forma estandarizada de comunicar riesgos técnicos a distintas audiencias, incluyendo equipos técnicos, gestores de TI y responsables de cumplimiento. En este informe, la clasificación de las vulnerabilidades detectadas se ha realizado con base en las métricas definidas por CVSS versión 3.1, considerando tanto su

severidad técnica como su relevancia para el entorno evaluado.

2.3 Herramientas de Hacking Ético

El arsenal de un hacker ´ético incluye una variedad de herramientas diseñadas para la exploración, análisis, y explotación de vulnerabilidades. Herramientas como Nmap, para el escaneo de puertos, y Metasploit, para la explotación de vulnerabilidades, son esenciales en el proceso de evaluación de seguridad ((*NMAP Network Scanning*, s. f.) (*Metasploit Unleashed | Metasploit Unleashed - Free Online Ethical Hacking Course*, s. f.).

2.4 Marco Legal y Consideraciones Éticas

La práctica del hacking ético se rige por consideraciones legales y éticas estrictas, asegurando que todas las actividades se realicen con el consentimiento explicito y no violen las leyes aplicables (Zimmerman, 2019).

La práctica del hacking ético opera dentro de un marco legal y ético estricto, asegurando que las pruebas de penetración se realicen de manera responsable y con el consentimiento explícito de las partes afectadas (Kim, 2014). Este marco no solo protege a las instituciones y a los investigadores, sino que también asegura la integridad y la confidencialidad de los datos durante el proceso de evaluación.

2.5 Protocolos de Seguridad

Los protocolos de seguridad, tales como SSL/TLS, juegan un papel fundamental en la protección de la transmisión de datos a través de Internet. La implementación adecuada de estos protocolos asegura la confidencialidad e integridad de la información transmitida entre dos partes, previniendo ataques como el *eavesdropping* o *el man-in-the-middle* (Rescorla, 2018). Es crucial que los desarrolladores y administradores de sistemas comprendan y apliquen correctamente estos protocolos para fortalecer la seguridad de sus aplicaciones web y servicios en línea.

El impacto del hacking ´ético en la mejora de la seguridad informática es ampliamente reconocido. A través de la identificación proactiva de vulnerabilidades y la implementación de medidas correctivas, las organizaciones pueden significativamente reducir su riesgo de sufrir brechas de seguridad (OWASP Web Security Testing Guide | OWASP Foundation, s. f.).

2.6 Análisis y Gestión de Riesgos

El análisis y la gestión de riesgos informáticos son esenciales para identificar, evaluar y mitigar potenciales vulnerabilidades dentro de los sistemas y redes. La adopción de marcos como el NIST *Special Publication* 800-30 proporciona una guía detallada para realizar análisis de riesgos efectivos, permitiendo a las organizaciones priorizar las amenazas y aplicar recursos de manera eficiente para abordar los riesgos más críticos (Stoneburner, Goguen y Feringa, 2002). Este proceso es dinámico y requiere una revisión continua para adaptarse al cambiante panorama de amenazas cibernéticas.

2.7 Psicología de la Ingeniería Social

La ingeniería social explota las vulnerabilidades humanas para obtener acceso no autorizado a sistemas o información confidencial. Comprender la psicología detrás de la ingeniera social es crucial para desarrollar estrategias efectivas de concienciación y formación que puedan mitigar este tipo de ataques. Las técnicas de ingeniera social, como el *phishing*, se basan en la manipulación psicológica, haciendo que la educación en seguridad cibernética y la conciencia sean tan importantes como las medidas de seguridad técnicas (Hadnagy, 2018). Este enfoque multidisciplinario hacia la ciberseguridad resalta la importancia de considerar el factor humano al diseñar e implementar medidas de seguridad.

2.8 Vulnerabilidades

Los ataques cibernéticos explotan vulnerabilidades específicas dentro de sistemas y redes, comprometiendo la confidencialidad, integridad, y disponibilidad de la información. La comprensión de estas vulnerabilidades y los métodos de ataque asociados es fundamental para el desarrollo de estrategias de defensa efectivas (*OWASP Top Ten 2017 | 2017 Top 10 | OWASP Foundation*, s. f.) (CVE, 2022).

2.8.1 Clickjacking

Clickjacking es un tipo de ataque en el cual un atacante engaña a un usuario para que haga clic en un elemento de una página web sin su conocimiento. Esto se logra incrustando un sitio web dentro de un *iframe* y manipulando la interfaz para ocultar el contenido real bajo una capa falsa (Clickjacking Defense - OWASP Cheat Sheet Series, s. f.).

2.8.2 Encabezados de Respuesta Innecesarios

La inclusión de encabezados innecesarios en las respuestas HTTP puede revelar información sensible sobre el servidor, como detalles del software o la versión utilizada, lo que puede ser aprovechado por un atacante para realizar otros tipos de ataques (*OWASP Secure Headers Project | OWASP Foundation*, s. f.).

2.8.3 Faltan los Encabezados HTTP

La ausencia de ciertos encabezados HTTP de seguridad, como X-Frame-Options o Strict-Transport-Security, deja a la aplicación web vulnerable a ataques como *Clickjacking*, SSL *stripping*, o manipulación de contenido (*OWASP Secure Headers Project | OWASP Foundation*, s. f.).

2.8.4 Exposición de Recursos Innecesarios

La exposición de archivos o directorios que no son necesarios para el funcionamiento

público de una aplicación web puede permitir a los atacantes acceder a información confidencial o vulnerar el sistema mediante la manipulación de esos recursos (OWASP Top Ten | OWASP Foundation, s. f.).

2.8.5 Cookie con Atributo SameSite Ausente

El atributo SameSite en las cookies previene que sean enviadas en solicitudes entre sitios, lo que mitiga ataques como Cross-Site Request Forgery (CSRF). Su ausencia deja la aplicación web vulnerable a este tipo de ataques (Bingler et al., 2025).

2.8.6 Cookie sin Atributo Secure Ausente

Las cookies que carecen del atributo Secure pueden ser transmitidas en conexiones no cifradas, lo que expone los datos a posibles ataques de tipo *man-in-the-middle* (O. Foundation, 2020).

2.8.7 Expiración de Sesión Inadecuada

Si las sesiones no caducan apropiadamente, los usuarios pueden estar expuestos a ataques como secuestro de sesión. Es fundamental configurar un tiempo de expiración razonable para reducir este riesgo (O. Foundation, 2020).

2.8.8 Inyección de Cabeceras de Host

Este ataque ocurre cuando una aplicación web permite que un atacante inyecte un valor malicioso en el encabezado Host, lo que podría resultar en una redirección indebida o en ataques de envenenamiento de caché (O. Foundation, 2020).

2.8.9 Fuerza Bruta

El ataque de fuerza bruta implica intentar adivinar credenciales repetidamente hasta que se obtenga acceso a una cuenta. Sin medidas de mitigación, como limitaciones en los intentos de inicio de sesión, las aplicaciones web son vulnerables a este tipo de ataque (O.

Foundation, 2021).

2.8.10 Consola de Administración Expuesta

Permitir el acceso público a la consola de administración de un sistema es una práctica peligrosa, ya que los atacantes podrían obtener control sobre el sistema si logran ingresar con credenciales validas (O. Foundation, 2021).

2.8.11 Control de Acceso Inadecuado

El control de acceso deficiente permite a los usuarios no autorizados acceder a funciones o datos sensibles. La falta de políticas de acceso bien definidas expone la aplicación a riesgos significativos (O. Foundation, 2020).

2.8.12 Cifrado Débil

El uso de algoritmos de cifrado débiles o claves insuficientemente largas puede permitir a los atacantes descifrar la información sensible, exponiendo datos como contraseñ as o números de tarjetas de crédito (O. Foundation, 2021).

2.8.13 Cookies Inseguras

Las cookies que no tienen configuraciones de seguridad adecuadas, como HttpOnly, Secure, o SameSite, pueden ser vulnerables a ataques de secuestro de sesión y otros tipos de manipulación (O. Foundation, 2019).

2.8.14 Divulgación de Información

La divulgación de información sensible, como mensajes de error detallados o metadatos en respuestas HTTP, puede proporcionar pistas a los atacantes sobre cómo explotar una vulnerabilidad en el sistema (O. Foundation, 2019).

2.8.15 Encabezados de Seguridad

La correcta configuración de encabezados HTTP, como Content-Security-Policy o X-

Content-Type-Options, ayuda a proteger las aplicaciones web de una amplia gama de ataques, incluidos inyecciones de código y ataques de tipo cross-site scripting (XSS) (O. Foundation, 2021).

2.8.16 Error de Configuración

Los errores de configuración, como usar credenciales predeterminadas o exponer ser- vicios innecesarios, pueden dejar a la aplicación web vulnerable a un ataque. La revisión periódica y ajuste de las configuraciones es esencial para mantener la seguridad (O. Foundation, 2020).

2.8.17 Acceso No Autorizado a Recursos Sensibles

Cuando los mecanismos de autenticación o autorización no están implementados correctamente, los usuarios pueden obtener acceso a información sensible o realizar acciones no permitidas (O. Foundation, 2020).

2.8.18 Bloqueo en WiFi No Realizado

Si no se implementa un bloqueo adecuado en redes Wifi, los atacantes pueden interceptar y manipular el tráfico de red, comprometiendo la confidencialidad de la información transmitida (O. Foundation, 2019).

2.8.19 Carga de Archivos No Controlada

Permitir que los usuarios suban archivos sin un control adecuado puede ser peligroso, ya que podrían subir archivos maliciosos que comprometan el sistema. Se recomienda establecer filtros y restricciones estrictas para prevenir esto (O. Foundation, 2021).

2.8.20 Falta de Autenticación para Recurso Web

Cuando un recurso web sensible no requiere autenticación, cualquier usuario puede acceder a ´el, lo que pone en riesgo la seguridad de la aplicación. Es fundamental exigir autenticación para proteger dichos recursos (O. Foundation, 2020).

2.8.21 Nginx Desactualizado

El uso de versiones obsoletas de servidores web como Nginx puede exponer la aplicación a vulnerabilidades conocidas. Mantener el software actualizado es crucial para evitar estos riesgos (O. Foundation, 2020).

2.8.22 Configuración de Credenciales Débiles

El uso de credenciales débiles, como contraseñas simples o predeterminadas, facilita los ataques de fuerza bruta y pone en riesgo la seguridad de la aplicación (O. Foundation, 2020).

2.8.23 Listado de Directorios

La exposición de directorios sin protección permite a los atacantes listar y acceder a archivos sensibles, lo que puede conducir a la explotación del sistema (O. Foundation, 2019).

3. Metodologías

3.1 Metodologías comunes

El hacking ´ético involucra una serie de metodologías estandarizadas que guían a los profesionales de la seguridad en el proceso de identificación y mitigación de vulnerabilidades en sistemas informáticos. Estas metodologías son fundamentales para realizar pruebas de penetración de manera sistemática y eficaz.

3.1.1 Metodología OSSTMM

La Metodología Abierta de Pruebas de Seguridad de Sistemas de Información (OSSTMM, por sus siglas en inglés) es una de las metodologías más respetadas y utilizadas en el mundo del hacking ´ético. Proporciona un marco comprensivo que cubre la seguridad en áreas de operaciones de red, comunicaciones y físicas. OSSTMM aboga por un proceso de pruebas riguroso que se centra en medir y comparar los niveles de seguridad operativa (Herzog, 2010).

3.1.2 Metodología PTES

La Metodología de Pruebas de Penetración Estándar (PTES, por sus siglas en inglés) ofrece un marco detallado que guía a los profesionales de seguridad a través de todas las fases de una prueba de penetración. Desde el pre-engagement y la inteligencia hasta el reporte de hallazgos, PTES ayuda a asegurar que todos los aspectos de la prueba de penetración sean cubiertos de manera exhaustiva (Long, 2010).

3.1.3 Metodología OWASP

La Fundación OWASP (Open Web Application Security Project) es bien conocida por su enfoque en la seguridad de aplicaciones web. Proporciona múltiples recursos, incluyendo guías, herramientas y foros para discutir las mejores prácticas de seguridad de aplicaciones web. La metodología de OWASP destaca por su enfoque práctico y dirigido hacia la creación de aplicaciones seguras desde su desarrollo hasta su despliegue (O. Foundation, 2020).

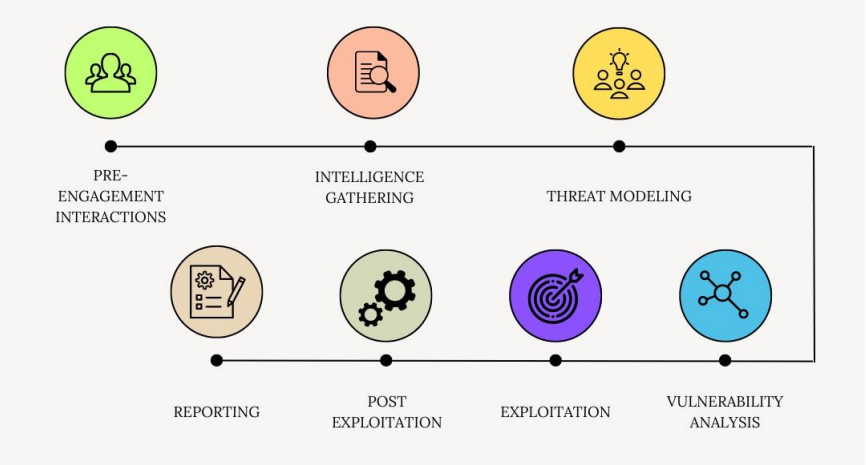
3.2 Metodología de Base de Referencia

La metodología PTES proporciona un marco de referencia que guía todas las fases de una prueba de penetración, desde la pre-engagement hasta el reporte de hallazgos. Este estándar asegura un enfoque estructurado y coherente, facilitando la identificación exhaustiva de

vulnerabilidades y la propuesta de medidas de mitigación efectivas (PTES Technical Guidelines - The Penetration Testing Execution Standard, s. f.). La metodología se desglosa en las siguientes fases principales:

Figura 1

Gráfica de la Metodología PTES.



Fuente: Elaboración propia.

3.2.1 Pre-engagement Interactions

En esta fase inicial, se alinearán los términos y condiciones del proyecto con las metas establecidas, asegurando que el alcance de la prueba abarque todas las áreas críticas de seguridad identificadas en el análisis objetivo y preparando el terreno para las pruebas manuales de penetración.

3.2.2 Intelligence Gathering

La recopilación de inteligencia estará´ enfocada en acumular información detallada sobre los servicios específicos que son críticos para la seguridad de la universidad. Este proceso es esencial para planificar eficientemente las pruebas de penetración manual y automatizada que se describen en los objetivos.

3.2.3 Threat Modeling

La fase de modelado de amenazas se centrará´ en identificar y priorizar las amenazas que podrían afectar los servicios críticos. Esta información servirá´ de base para desarrollar una estrategia de pruebas que documente exhaustivamente los fallos de seguridad.

3.2.4 Vulnerability Analysis

Durante el análisis de vulnerabilidades, utilizaremos herramientas automatizadas y manuales para examinar los servicios objetivo. Esta etapa es vital para detectar deficiencias y formular medidas de mitigación que se ajusten a las necesidades específicas identificadas.

3.2.5 Exploitation

La explotación controlada de vulnerabilidades confirmará su existencia y proporcionará´ una comprensión práctica de su impacto. Esto directamente apoya el objetivo de proponer recomendaciones basadas en vulnerabilidades reales y documentadas.

3.2.6 Post Exploitation

Una vez confirmadas las vulnerabilidades, se evalúa el impacto potencial de las mismas. Esta etapa permite entender el alcance real de un posible ataque y qué sistemas o datos podrían verse comprometidos.

3.2.7 Reporting

Finalmente, la elaboración de un informe detallado cerrara´ el ciclo del proyecto. Este documento no solo registrará los hallazgos y evidencia de las pruebas, sino que también incluirá´ propuestas concretas para medidas de mitigación alineadas con las vulnerabilidades y riesgos específicos encontrados, cumpliendo así con el último de nuestros objetivos específicos.

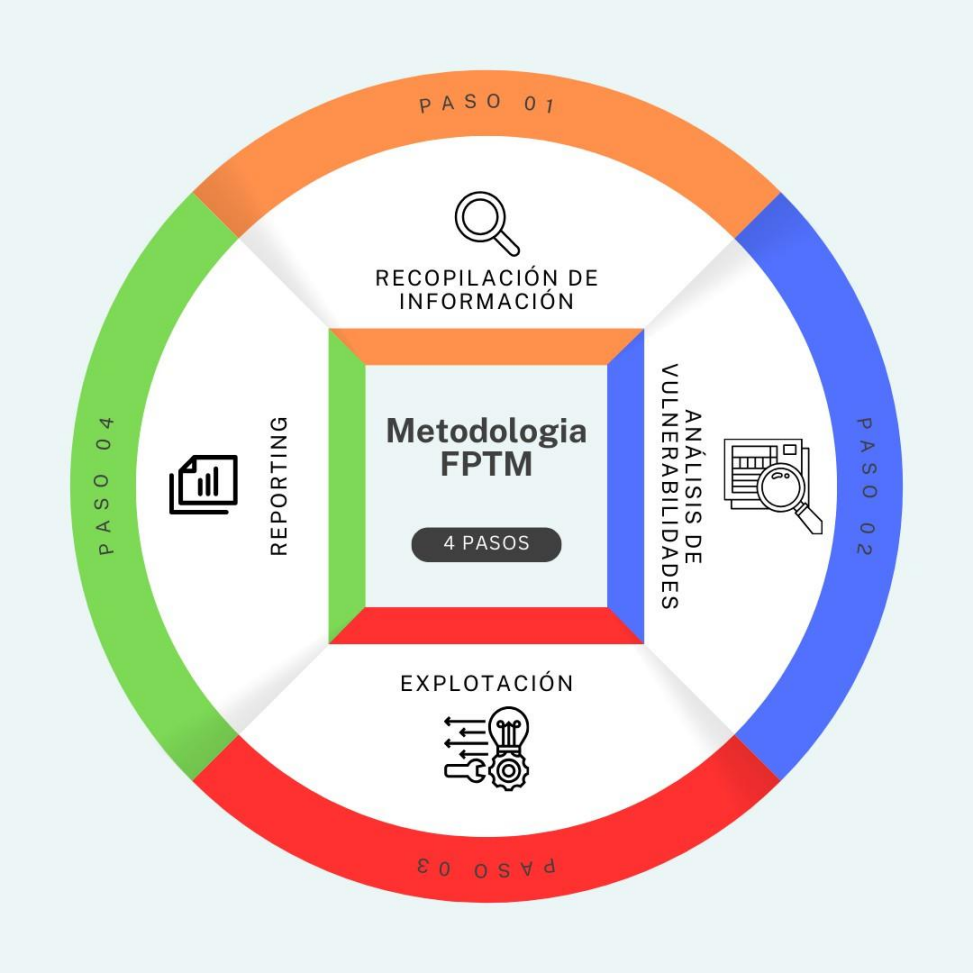
La metodología PTES (*Penetration Testing Execution Standard)* ofrece un marco ampliamente aceptado y probado que cubre de forma integral todo el ciclo de vida de una prueba de penetración, desde la planificación hasta la presentación de resultados. Tomarla como base aporta rigor, trazabilidad y alineación con las mejores prácticas internacionales, garantizando que cada fase —recolección de información, análisis de vulnerabilidades, explotación y reporte— siga procesos claramente definidos y repetibles. Sin embargo, las particularidades operativas y de riesgo de la Universidad Industrial de Santander exigen un enfoque más focalizado en los puntos donde su exposición es mayor. Por ello, partiendo de la estructura sólida de PTES, se diseñó la Metodología de Prueba de Penetración Focalizada (FPTM): conserva la disciplina y el orden de PTES, pero depura y prioriza actividades para maximizar el uso de recursos y concentrarse en las superficies críticas identificadas durante la fase de contextualización del proyecto. De esta manera, se logra un equilibrio entre la estandarización reconocida y la adaptabilidad necesaria para responder a los objetivos y restricciones propios del entorno universitario.

3.3 Metodología Propuesta para el Proyecto

El proyecto se ha basado en una metodología de prueba de penetración adaptada de la metodología PTES, denominada Metodología de Prueba de Penetración Focalizada (FPTM). Esta metodología se centra específicamente en las fases críticas del proceso de pruebas de penetración que son más relevantes para el contexto y los objetivos específicos del proyecto.

Figura 2

Gráfica de la Metodología FPTM



Fuente: Elaboración propia.

3.3.1 Recopilación de Información

La primera fase de la FPTM es la Recopilación de Información, que implica la identificación y recolección de datos sobre el objetivo. Esto incluye el mapeo de la red, la enumeración de sistemas, y la identificación de servicios activos y aplicaciones críticas. Esta etapa es fundamental para establecer un entendimiento profundo del entorno objetivo y para planificar las siguientes fases de la prueba.

3.3.2 Análisis de Vulnerabilidades

Una vez recopilada la información necesaria, se procede con el Análisis de Vulnerabilidades. Esta fase evalúa los sistemas para identificar posibles debilidades que puedan ser explotadas. Utiliza herramientas automatizadas y técnicas manuales para realizar un barrido exhaustivo, identificando vulnerabilidades conocidas y configuraciones erróneas que representan un riesgo para la organización.

3.3.3 Explotación

La fase de Explotación es donde se ponen a prueba las vulnerabilidades identificadas. El objetivo es determinar si las vulnerabilidades pueden ser explotadas para ganar acceso o elevar privilegios dentro del sistema. Esta etapa verifica la severidad real de las debilidades detectadas y ayuda a comprender el posible impacto de un ataque.

3.3.4 Reporte

La última fase es el Reporte, que implica la documentación de los hallazgos, la elaboración de un informe detallado y la presentación de recomendaciones para mitigar los riesgos identificados. El informe debe proporcionar una visión clara de las vulnerabilidades explotadas, la exposición potencial y las medidas correctivas sugeridas para mejorar la seguridad del sistema. Esta metodología adaptada, FPTM, permite un enfoque eficiente y objetivo en las pruebas de penetración, asegurando que los recursos se concentren en las ´áreas más críticas y relevantes para la organización.

Este enfoque metódico, basado en la metodología PTES, garantiza una evaluación de seguridad informática profunda y efectiva, permitiendo a la UIS fortalecer su postura de seguridad frente a amenazas cibernéticas, ahora continuando con el desarrollo de este libro, vamos a pasar a la sección de implementación de la FPTM.

4. Implementación de la FPTM

Una vez completado el proceso de identificación y análisis de los servicios institucionales seleccionados, se procedió´ con la implementación formal de las pruebas de penetración ética. Estas pruebas se llevaron a cabo en un entorno controlado, siguiendo lineamientos previamente establecidos y orientadas a evaluar el nivel real de exposición frente a amenazas externas.

La ejecución se estructuró conforme a los principios metodológicos definidos en el marco de evaluación adoptado, combinando herramientas automatizadas con técnicas manuales. Se inicio´ con la recolección de información pública (OSINT), seguida de un reconocimiento técnico de los servicios accesibles desde el exterior, con el objetivo de identificar vectores de ataque potenciales.

Durante esta fase, se emplearon herramientas ampliamente reconocidas en el ´ámbito de la ciberseguridad, tales como Nmap (Lyon, 2009), para la detección de servicios y mapeo de puertos; Nikto (Sullo, 2024) para la identificación de configuraciones inseguras y vulnerabilidades web conocidas; OpenVAS (Greenbone Networks, 2024) como escáner de vulnerabilidades integral; y Burp Suite (PortSwigger, 2024), utilizada en la etapa de análisis e interacción manual con aplicaciones web.

El uso combinado de estas herramientas proporciono´ una visión preliminar robusta sobre el estado de seguridad de los sistemas evaluados, permitiendo priorizar áreas críticas para análisis más detallados. Todas las acciones realizadas fueron debidamente documentadas, garantizando la trazabilidad del proceso y contribuyendo a la elaboración de un informe técnico estructurado y alineado con los principios éticos y legales que rigen este tipo de actividad.

5. Recopilación de Información

Durante esta fase se obtuvieron datos confidenciales relativos a la infraestructura y a los servicios institucionales evaluados. Por motivos de seguridad y para preservar la integridad de la organización, dichos detalles no se muestran de forma explícita en este documento. Sin embargo, la información recopilada fue esencial para orientar y fundamentar las actividades de análisis, explotación y mitigación desarrolladas en las fases posteriores del proyecto.

5.1 Objetivos de prueba

Para la realización de las pruebas de hacking ético, se nos han proporcionado cinco objetivos especifico, los cuales por cuestiones de confidencialidad se pondrán con un alias y una descripción. Estos entornos permitirán evaluar diferentes técnicas de análisis y explotación de seguridad.

Los objetivos de prueba son los siguientes:

- Página principal de la Universidad. - nombre clave: uisP
- Aplicación de gestión para el código desarrollado en proyectos de DTIC. - nombre clave: uisG
- Sistema de Mesa de ayuda de la UIS. - nombre clave: uisM
- Aplicación para Gestión de proyectos de desarrollo. - nombre clave: uisT
- Aplicación de Carnet Virtual para estudiantes UIS. - nombre clave: uisC

Cada uno de estos objetivos será´ analizado utilizando herramientas y metodología FPTM, con el fin de identificar vulnerabilidades y evaluar su impacto en la seguridad de los sistemas.

5.2 Recopilación de Información

Durante la fase de recopilación de información de la FPTM, se ejecutaron diversas actividades orientadas a obtener una comprensión detallada del entorno tecnológico objetivo,

sentando las bases para un análisis riguroso de vulnerabilidades.

El proceso comenzó con la identificación y enumeración de activos relevantes, tales como direcciones IP, nombres de dominio, servidores públicos y dispositivos de red asociados al sistema en evaluación. Para ello, se aplicaron técnicas de reconocimiento pasivo, incluyendo la consulta de registros WHOIS (IANA, 2024), el uso de herramientas OSINT como theHarvester (Martorella, 2024) y Shodan (Shodan, 2024), que facilitaron la obtención de información pública sin interacción directa con los sistemas objetivos.

Posteriormente, se recurrió al reconocimiento activo para mapear la superficie de exposición mediante el escaneo de puertos y detección de servicios en ejecución, utilizando herramientas como Nmap. Esta etapa permitió´ identificar configuraciones visibles desde el exterior, así como versiones específicas de software y sistemas operativos en uso, lo que contribuyó a construir un perfil técnico preciso sobre posibles vulnerabilidades asociadas.

Adicionalmente, se llevó´ a cabo una revisión de la arquitectura de red y de las políticas de seguridad mediante el análisis de documentación técnica interna y entrevistas estructuradas con personal del área de tecnologías de la información. Este enfoque mixto— combinando fuentes externas e internas— aseguró una visión integral del ecosistema evaluado.

La consolidación de toda la información recopilada permitió´ establecer una línea base sobre la cual enfocar el análisis de vulnerabilidades. (La información recopilada debido a que es de carácter confidencial se anexa al informe final) Así, se identificaron posibles vectores de ataque, clasificando los hallazgos preliminares en función de su nivel de exposición y criticidad, y estableciendo prioridades para la fase siguiente de Diseño de pruebas manuales.

6. Diseño de pruebas manuales

El uso de pruebas manuales de penetración resulta fundamental en un proyecto de hacking ético porque permite identificar vulnerabilidades que los escáneres automatizados suelen pasar por alto. Al reproducir de manera controlada el pensamiento creativo y la lógica de un atacante humano, se logra: (i) descubrir fallos lógicos y errores de configuración dependientes del contexto operacional; (ii) validar la explotación real de vulnerabilidades reportadas por herramientas automáticas, reduciendo falsos positivos; y (iii) evaluar la resiliencia del sistema ante tácticas de evasión personalizadas. De este modo, el diseño de pruebas manuales complementa las pruebas automatizadas y proporciona una visión más completa y realista del estado de seguridad de los servicios institucionales evaluados, habilitando recomendaciones precisas y priorizadas para su mitigación. por lo anteriormente dicho diseñarlas es una parte fundamental para continuar con nuestro ejercicio.

6.1. Listado de directorios

Objetivo: Verificar si el servidor publica el índice de contenido cuando falta el archivo por defecto (index).

Alcance: Servidor Web principal y subdirectorios (/uploads, /temp, /backup).

Prerrequisitos: Navegador y extensión de proxy (Burpsuit) configurados; acceso HTT-P/HTTPS sin autenticación.

Procedimiento:

1. Navegar directamente a la raíz y a rutas potenciales usando un diccionario con ffuf o gobuster.
2. Observar si la respuesta presenta un listado con hipervínculos.

Criterio de éxito: Se muestra un índice con al menos un archivo o subdirectorio listados.

6.2. Acceso no autorizado a recursos sensibles

Objetivo: Comprobar que páginas/archivos marcados como “privados” requieran sesión.

Alcance: URLs internas como /perfil/*, /admin/reportes, APIs REST.

Prerrequisitos: Navegador en modo incógnito, cookies limpias.

Procedimiento:

1. Desloguearse y limpiar sesión.
2. Solicitar cada URL sensible manualmente o con curl.

Criterio de éxito: El recurso carga (HTTP 200) o devuelve datos sin redirección al login.

6.3. Carga de archivos no controlada

Objetivo: Verificar controles de tipo y contenido en formularios de subida.

Alcance: módulos de CV, repositorio de tareas, formularios de soporte.

Prerrequisitos: Cuenta de usuario estándar; plantillas de archivos “polyglot” (shell.php.jpg, test.<svg onload=alert(1)>).

Procedimiento:

1. Subir archivos con extensiones dobles o contenido activo.
2. Interceptar la petición y alterar el nombre a una extensión prohibida.
3. Anotar la ubicación final y, si procede, solicitar la URL resultante.

Criterio de éxito: El archivo sube y queda accesible o se ejecuta código/JavaScript.

6.4. Falta de autenticación para recurso web

Objetivo: Asegurar que todas las funciones críticas requieren login.

Alcance: Paneles de profesor, endpoints de calificaciones, páginas de configuración.

Procedimiento:

1. Buscar rutas administrativas/REST con herramientas como Feroxbuster.

Criterio de éxito: 200/JSON válido sin cabecera 401/403 ni redirección.

6.5. Divulgación de información

Objetivo: Detectar fuga de versiones, rutas y datos internos mediante errores o cabeceras.

Alcance: Cualquier endpoint con manejo de errores; cabeceras HTTP globales.

Procedimiento:

1. Introducir parámetros inválidos para forzar excepciones.
2. Revisar respuesta y cabeceras Server, X-Powered-By.
3. Enumerar archivos ocultos (.git, .env) con dirbusting.

Criterio de éxito: Se revela software, rutas o información sensible.

6.6. Consola de administración expuesta

Objetivo: Determinar si interfaces de gestión (Tomcat, phpMyAdmin, CMS) están accesibles públicamente.

Procedimiento:

1. Ejecutar nmap --script http-enum sobre el host.
2. Probar manualmente rutas comunes: /manager/html, /phpmyadmin/, /admin.

Criterio de éxito: Página de login de consola visible o acceso sin credenciales.

6.7. *Clickjacking*

Objetivo: Verificar protección X-Frame-Options / CSP frame-ancestors.

Procedimiento:

1. Crear un HTML local con un *iframe* apuntando al dominio objetivo.
2. Abrirlo en navegador y comprobar si la página se renderiza.

Criterio de éxito: Página visible dentro del *iframe*.

6.8. Bloqueo en Wifi no realizado

Objetivo: Comprobar que no se puedan acceder a recursos sensibles desde el wifi.

Procedimiento:

1. con nmap buscar todos los recursos alcanzables desde el wifi.

Criterio de éxito: obtener acceso a recursos o páginas con información sensible.

6.9. Inyección de cabeceras Host

Objetivo: Validar que la aplicación no confía ciegamente en la cabecera Host.

Procedimiento:

1. Repetir una petición legítima cambiando Host: evil.com (Por ejemplo).
2. Observar si en la respuesta se reflejan enlaces con el host modificado.

Criterio de éxito: Reflexión o uso de sitios maliciosos para inyectar en la cabecera Host.

6.10. Fuerza bruta

Objetivo: Evaluar protección contra múltiples intentos de autenticación.

Procedimiento:

1. Configurar Burp Intruder con combinación usuario fijo + 50 contraseñas comunes.
2. Lanzar 100–200 intentos y medir si surge bloqueo/CAPTCHA.

Criterio de éxito: Más de X intentos sin defensa visible.

6.11. Exposición de recursos innecesarios

Objetivo: Identificar archivos/directorios obsoletos en producción.

Procedimiento:

1. Escaneo con Feroxbuster usando diccionario common.txt.

Criterio de éxito: Acceso a backups, entornos /old, /test o dumps SQL.

6.12. Control de acceso inadecuado

Objetivo: Validar separación de permisos usuario - administrador.

Procedimiento:

1. Iniciar sesión como usuario estándar.
2. Cambiar parámetros (id, role) en URL o cuerpo JSON.

3. Interceptar respuesta; comprobar si la acción se ejecuta.

Criterio de éxito: Acceso o modificación de datos/funciones ajenas.

6.13. Expiración de sesión inadecuada

Objetivo: Verificar tiempo de vida y revocación de sesión.

Procedimiento:

1. Autenticarse y capturar cookie.
2. Tras periodo inactivo (según política, p. ej. 30 min), reenviar la misma cookie.

Criterio de éxito: Sesión sigue activa más allá del umbral definido.

El diseño de pruebas manuales constituye el puente metodológico entre la planificación y el análisis de vulnerabilidades: establece, antes de cualquier ejecución, los objetivos, alcances y criterios de éxito que guían cada intento de explotación. Al llevar a cabo estas pruebas según lo previsto, los hallazgos obtenidos se documentan y contrastan con los criterios definidos, permitiendo interpretar de forma estructurada la exposición real de los servicios evaluados. Así, el análisis de vulnerabilidades se sustenta en evidencias coherentes con el plan de pruebas, garantizando que las conclusiones y recomendaciones reflejen con precisión los riesgos identificados.

7. Análisis de vulnerabilidades

En esta fase se realizó un análisis exhaustivo de las posibles vulnerabilidades presentes en el sistema evaluado. Se utilizaron herramientas y técnicas especializadas para identificar debilidades en la configuración, el software y la infraestructura.

7.1 Identificación de Vulnerabilidades

Se emplearon herramientas de escaneo automatizado y manual para detectar vulnerabilidades conocidas y potenciales, los pasos que se siguieron fueron:

- Escáneres de vulnerabilidades (por ejemplo, Nessus, OpenVAS).

- Análisis de configuración de sistemas.
- Revisión de versiones de software y Vulnerabilidades recientes.

7.2 Escaneo de Vulnerabilidades con OpenVAS

Para la evaluación de seguridad, se utilizó OpenVAS (Open Vulnerability Assessment System), una herramienta de código abierto diseñada para realizar análisis automatizados de vulnerabilidades en sistemas y redes, se aclara que para este ejercicio se va a explicar únicamente la configuración básica para hacer el escaneo obviando las diferentes funcionalidades de la herramienta, en caso tal de requerir más información de las funcionalidades de la misma se puede acceder a la documentación oficial de Greenbone (*GreenBone Cloud Service*, s. f.).

7.3 Configuración del Escaneo

Para demostrar el uso de OpenVAS, se realizó un escaneo sobre los Objetivos de prueba anteriormente establecidos. Los pasos seguidos fueron los siguientes:

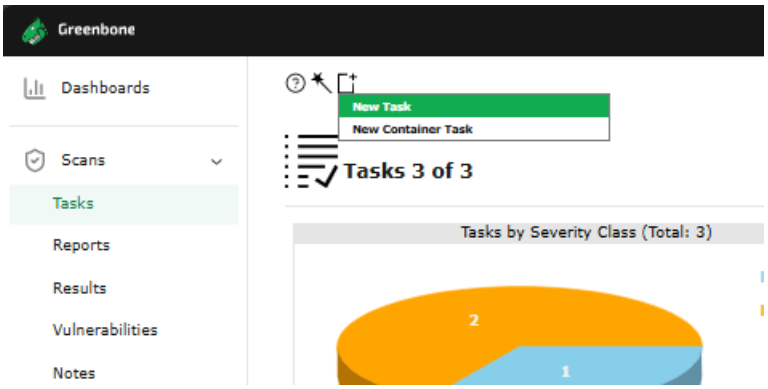
- Selección de Objetivos: Se definieron direcciones IP pertenecientes a los sistemas a evaluar.
- Configuración del Escaneo: Se utilizo´ el perfil de escaneo Full and Fast Como indica la documentación oficial de Greenbone el cual permite una evaluación integral de vulnerabilidades conocidas.
- Ejecución del Escaneo: Se inició el análisis y se monitoreó su progreso mediante la interfaz web de OpenVAS.

7.3.1 Creación de un Escaneo

Para iniciar un nuevo escaneo, se debe navegar a la sección correspondiente dentro de la interfaz. Esto se realiza a través del menú de navegación.

Figura 3

Ruta de acceso para configurar un escaneo.



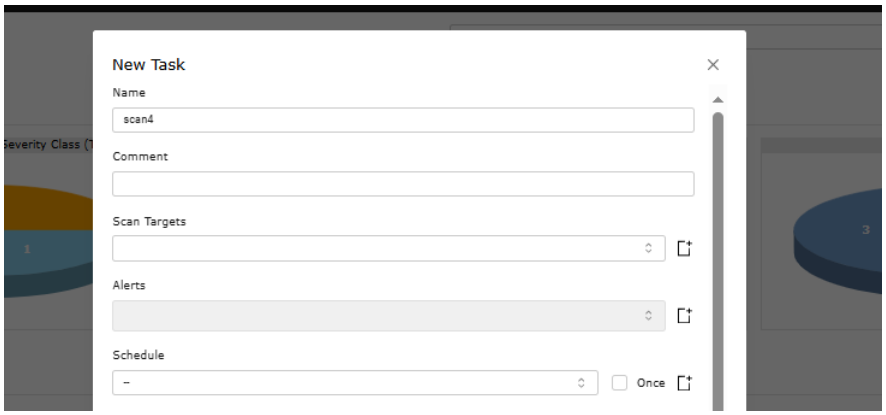
Como se muestra en la Figura 3, vamos a ir “New Task”, esto con el fin de crear un nuevo escaneo. También nos da la opción de “New container Task” para crear varios escaneos en simultaneo.

7.3.2 Configuración de un Nuevo Escaneo

Antes de ejecutar un escaneo, es necesario definir su configuración. Esto incluye parámetros como la profundidad del análisis y la política de escaneo utilizada.

Figura 4

Configuración inicial del escaneo en OpenVAS.

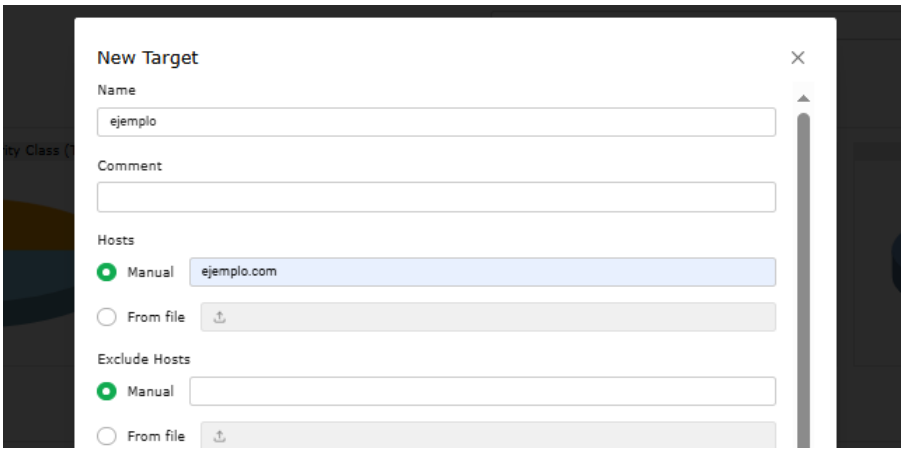


La Figura 4 muestra el panel donde debemos ponerle un nombre al escaneo, podemos ponerle comentarios, escogemos el objetivo en “scan target”, como se puede ver en la figura 5 y adicionalmente a esto podemos dejarlo programado para otra fecha o iniciarlo inmediatamente.

El siguiente paso en la configuración de OpenVAS es definir el objetivo del escaneo, es decir, la dirección IP o el rango de direcciones a analizar.

Figura 5

Configuración del objetivo del escaneo (target).



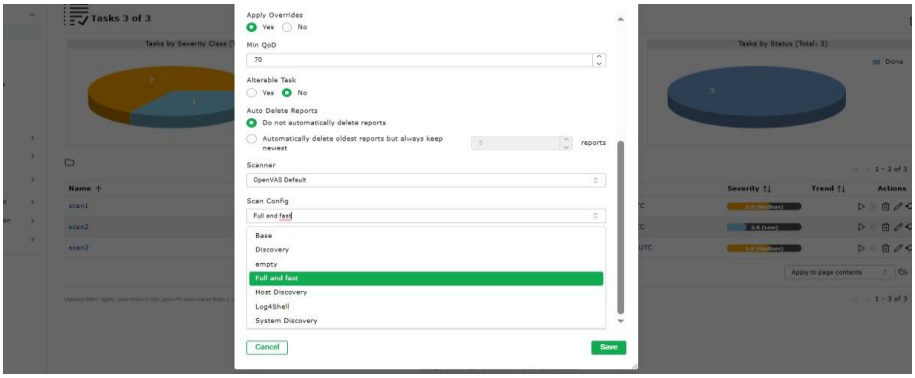
En la Figura 5, se observa la configuración del objetivo. Aquí se especifica la dirección IP del sistema que será analizado, junto con parámetros como la profundidad del escaneo y la autenticación, si es requerida.

7.3.3 Selección del Tipo de Escaneo

Una vez definido el objetivo, se debe seleccionar el tipo de escaneo a realizar. OpenVAS ofrece diferentes perfiles de escaneo según el nivel de análisis deseado.

Figura 6

Selección del tipo de escaneo en OpenVAS



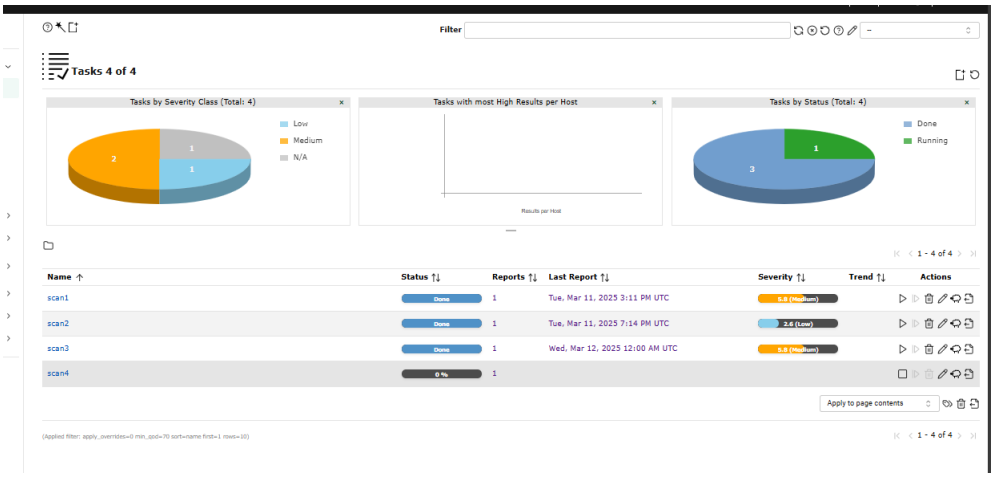
La Figura 6 muestra las diferentes opciones de escaneo disponibles, incluyendo escaneos rápidos, completos y personalizados, dependiendo del nivel de profundidad requerido, para nuestro ejercicio escogemos “full and fast”.

7.3.4 Ejecución del Escaneo

Tras completar la configuración, al hacer clic el botón “save”, el escaneo se ejecuta y OpenVAS comienza a analizar los sistemas en busca de vulnerabilidades.

Figura 7

Escaneo en ejecución en OpenVAS.



En la Figura 7, se puede observar el proceso de escaneo en ejecución. OpenVAS analiza los puertos abiertos, servicios en ejecución y posibles vulnerabilidades en la máquina objetivo.

7.4 Análisis de Riesgos Asociados

Para cada vulnerabilidad, se evaluaron los riesgos asociados considerando el contexto específico del sistema y la probabilidad de explotación. Se generaron recomendaciones preliminares para la remediación o mitigación de los riesgos identificados.

Esta fase proporcionó una visión detallada de las áreas más críticas que requieren atención en el siguiente paso de explotación, permitiendo enfocar los esfuerzos en aquellas vulnerabilidades que podrían tener un mayor impacto en la seguridad del sistema.

8. Explotación de Vulnerabilidades

En esta sección se presentan las evidencias obtenidas durante la explotación de vulnerabilidades en los objetivos de prueba. Con el fin de mantener un enfoque dinámico y evitar redundancias, se ha decidido priorizar aquellas vulnerabilidades clasificadas como medias, altas y críticas, de acuerdo con estándares como CVSS (Common Vulnerability Scoring System) 4.2. Esto se debe a que, si bien se han identificado múltiples vulnerabilidades de severidad baja y media en los distintos entornos evaluados, muchas de ellas son recurrentes y su análisis no aportaría un valor significativo a la investigación.

Las vulnerabilidades presentadas en este capítulo han sido seleccionadas por su potencial impacto en la seguridad de los sistemas, ya que podrían comprometer la confidencialidad, integridad y disponibilidad de la información. Además, se documentan los métodos utilizados para su explotación.

8.1 Herramientas Utilizadas

Durante la fase de explotación, se utilizaron herramientas especializadas que permitieron validar la existencia de vulnerabilidades previamente identificadas y simular escenarios reales de ataque de forma controlada.

Para la ejecución de pruebas de penetración avanzadas, se empleó el Metasploit Framework (Rapid7), una plataforma ampliamente reconocida que proporciona una colección de exploits, payloads y módulos auxiliares para evaluar la seguridad de los sistemas. Esta herramienta permitió probar de forma segura vulnerabilidades conocidas y obtener acceso controlado a servicios comprometidos.

Con el fin de detectar y explotar vulnerabilidades del tipo inyección SQL, se utilizó SQLmap (Belluci, 2025), una herramienta automatizada que facilita la identificación de puntos vulnerables en bases de datos y permite la extracción de información crítica en condiciones controladas.

Para realizar ataques de enumeración de directorios y descubrir rutas expuestas en aplicaciones web, se recurrió a Feroxbuster (epi052), una herramienta de fuerza bruta optimizada para análisis de estructura y contenidos en servidores HTTP.

Finalmente, se empleó Burp Suite Community Edition para llevar a cabo auditorias manuales de peticiones HTTP y análisis de APIs, permitiendo una inspección granular de los flujos de comunicación y detección de problemas como inyecciones, control de acceso deficiente o exposición de datos sensibles.

La combinación de estas herramientas permitió simular distintos vectores de ataque con precisión, fortaleciendo la validación técnica de los hallazgos obtenidos durante las fases anteriores del proceso de evaluación.

8.2. Ejemplos de Explotación

A continuación, se presentan unos bocetos diseñados para ejemplificar las vulnerabilidades encontradas en el proyecto, esto con el fin de mantener la confidencialidad del mismo y evitar que dicha información sea usa de manera maliciosa.

8.3. Listado de Directorios

Durante las pruebas de seguridad, se identificó´ una vulnerabilidad de listado de directorios en el servidor web del objetivo. Esta vulnerabilidad permite a un atacante visualizar y acceder a archivos y carpetas que no deberían ser públicamente accesibles, lo que podría exponer información sensible como credenciales, configuraciones y código fuente.

Utilizando la herramienta feroxbuster, se realizó un escaneo que confirmó la exposición de múltiples directorios accesibles sin autenticación esto se puede ver en la figura 8.

El escaneo se realizó con el comando feroxbuster -u https://patinetaspro.com/ -w /diccionario.txt, empleando una lista personalizada de directorios.

Figura 8

Ejemplo de Listado de Directorios

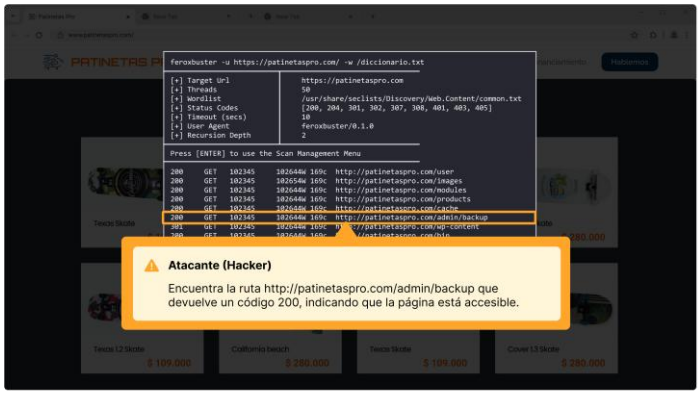
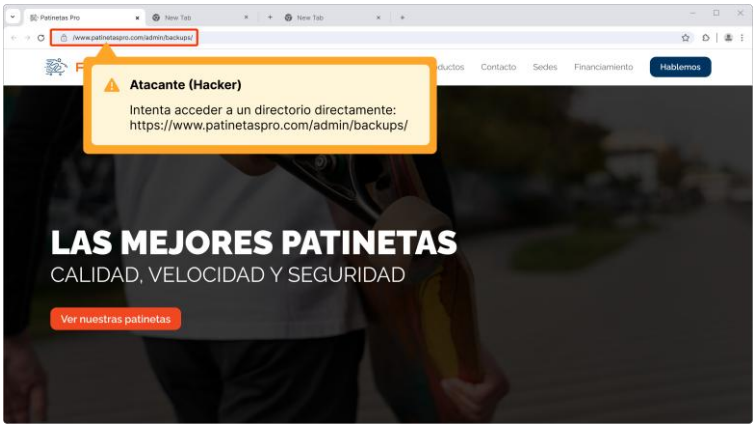


Figura 9

Ejemplo de Listado de Directorios



El resultado del escaneo reveló la presencia de directorios sensibles en el servidor, entre los cuales se identificó uno que contenía archivos de configuración expuestos. Estos archivos podrían incluir credenciales, claves de acceso u otra información critica que, en manos de un atacante, comprometería la seguridad del sistema. La figura 9 ilustra este hallazgo, destacando la accesibilidad de dichos archivos y el potencial riesgo asociado a su exposición.

Listing 1: Salida de feroxbuster mostrando directorios accesibles

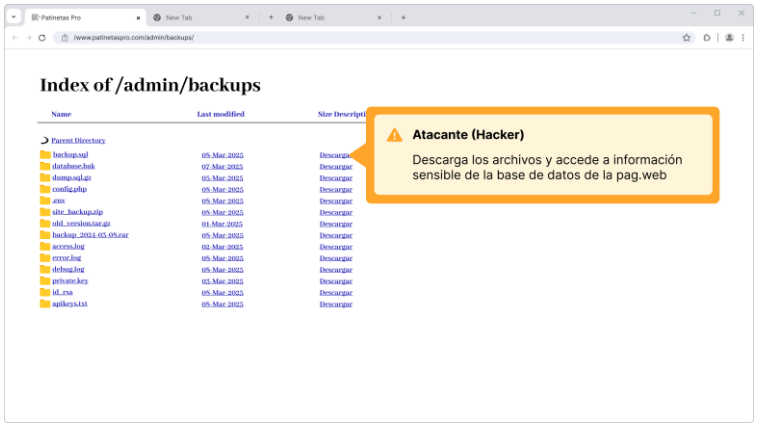
+ https : // p a t i n e ta s p r o . com/ admin/ backups /

Una vez identificado este directorio, un atacante puede acceder a él directamente a

través de la URL, explorando su contenido en busca de información sensible. Este tipo de exposición representa un riesgo significativo, ya que podría permitir la filtración de credenciales, configuraciones internas o incluso accesos administrativos no autorizados. En la figura 10 se ilustra cómo un atacante podría explotar esta vulnerabilidad para obtener datos críticos del sistema.

Figura 10

Ejemplo de Listado de Directorios



El acceso a estos directorios podría permitir a un atacante extraer información sensible del sitio web, lo que representa un grave riesgo de seguridad. En este caso particular, la exposición de archivos de configuración o bases de datos podría revelar credenciales de usuarios, incluidas contraseñas almacenadas de manera insegura. Esta vulnerabilidad podría facilitar accesos no autorizados, suplantación de identidad o incluso la toma de control del sistema. La figura 11 ilustra cómo un atacante puede aprovechar esta debilidad para obtener información crítica.

Figura 11

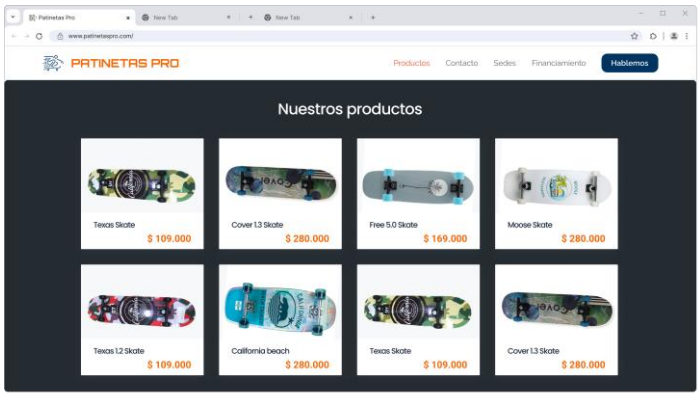
Ejemplo de Listado de Directorios

8.4. Acceso no autorizado a Recursos Sensibles

Cuando un atacante logra acceder, cambiar o borrar información que debería estar protegida, estamos ante un caso de acceso no autorizado a recursos Críticos. Estos incidentes pueden surgir por configuraciones mal establecidas, ausencia de sistemas de autenticación o autorización apropiados, o deficiencias en los mecanismos de control de acceso. Es esencial corregir estos problemas para garantizar la seguridad de la información.

Figura 12

Ejemplo de Acceso no autorizado a Recursos Sensibles

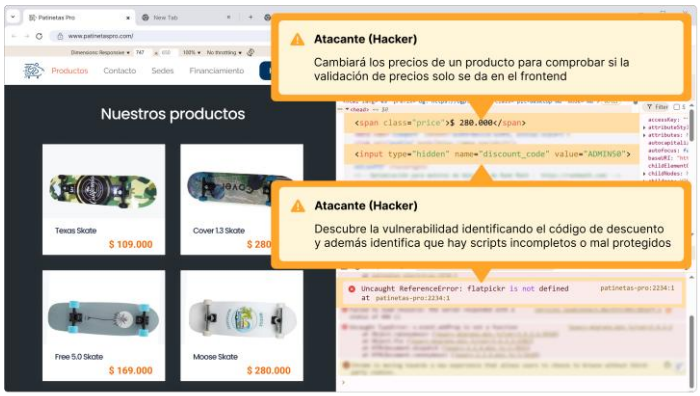


En la figura 12 se observa que un usuario no autenticado ha accedido a la página principal del sitio. Dado que se trata de un usuario malicioso, es probable que inicie una serie de pruebas manuales con el objetivo de identificar posibles vulnerabilidades y obtener acceso no autorizado. Estas pruebas pueden incluir la manipulación de parámetros en la URL, la

exploración de directorios ocultos o el intento de acceso a paneles de administración restringidos.

Figura 13

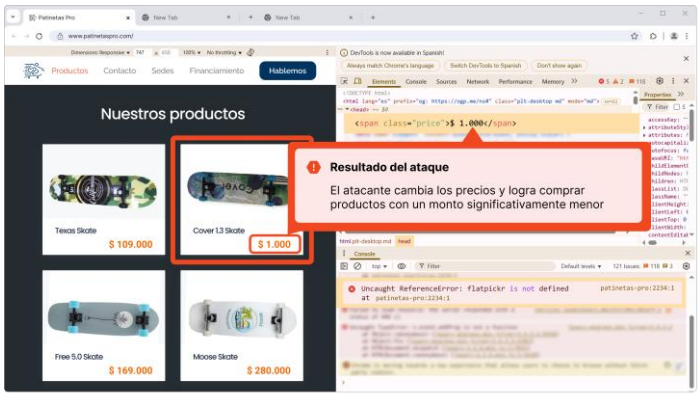
Ejemplo de Acceso no autorizado a Recursos Sensibles



En la figura 13 se evidencia cómo el atacante, utilizando la herramienta 'Inspeccionar' presente en los navegadores, intenta modificar elementos del frontend de la aplicación. Su objetivo es evaluar si existen mecanismos de seguridad implementados en el lado del cliente que restrinjan el acceso a ciertas funciones o contenidos. Este tipo de prueba es común en ataques donde se busca evadir validaciones superficiales de autorización, permitiendo potencialmente el acceso a secciones restringidas sin credenciales válidas.

Figura 14

Ejemplo de Acceso no autorizado a Recursos Sensibles

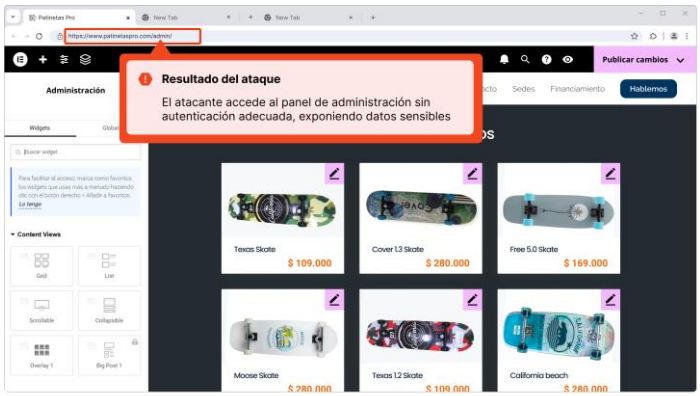


En la figura 14 se muestra cómo un usuario sin los permisos adecuados puede

modificar o interactuar con elementos del sistema, evidenciando una deficiencia en los controles de acceso. Esta vulnerabilidad podría ser aprovechada por un atacante para manipular configuraciones, acceder a funciones restringidas o incluso modificar datos críticos. En un escenario más grave, esta falla podría derivar en una escalación de privilegios, permitiendo que un usuario no autorizado obtenga mayores permisos dentro del sistema y comprometa su seguridad de manera significativa.

Figura 15

Ejemplo de Acceso no autorizado a Recursos Sensibles



En la figura 15 se muestra cómo un usuario sin los permisos adecuados logra acceder al panel de administración del sitio. Esta vulnerabilidad amplía significativamente la superficie de ataque, ya que desde esta interfaz el atacante podría intentar realizar cambios en la configuración del sistema, crear o modificar cuentas de usuario, extraer información sensible o incluso desplegar otras técnicas de explotación para obtener un control aún mayor sobre la plataforma. La falta de restricciones efectivas en este acceso representa un riesgo crítico para la seguridad del sitio.

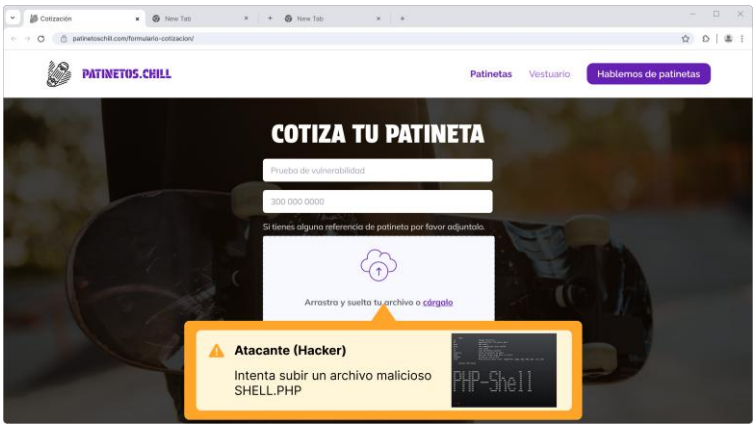
8.5. Carga de Archivos no Controlada

La carga de archivos no controlada se refiere a un fallo en aplicaciones web que dejan a los usuarios subir archivos sin las debidas comprobaciones o filtros. Si una aplicación no revisa

correctamente los archivos que se suben, un atacante podría introducir y activar archivos dañinos.

Figura 16

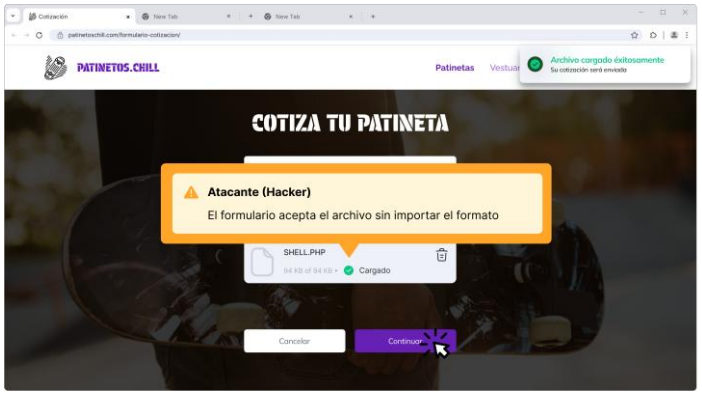
Ejemplo de Carga de Archivos no controlada



En la figura 16 se observa que la página web cuenta con un panel de carga de archivos. Un usuario malicioso está intentando aprovechar esta funcionalidad para subir un archivo malicioso denominado ‘shell.php’. Este tipo de ataque es común en aplicaciones web con controles de seguridad insuficientes, permitiendo que el atacante cargue y ejecute código arbitrario en el servidor. Si la carga es exitosa y el archivo se ejecuta, podría otorgarle al atacante acceso remoto al sistema, facilitando la manipulación de archivos, la extracción de información sensible o incluso la escalación de privilegios para obtener un control total del servidor.

Figura 17

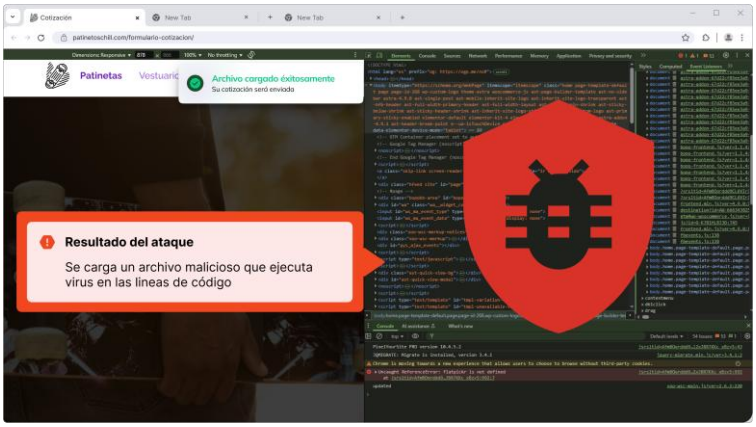
Ejemplo de Carga de Archivos no controlada



En la figura 17 se evidencia que el formulario de carga de archivos no está validando correctamente la extensión del archivo, lo que permite al atacante subir un malware. La ausencia de filtros adecuados en este proceso representa una vulnerabilidad crítica, ya que archivos maliciosos, como ‘shell.php’, podrían ser ejecutados en el servidor, otorgando acceso no autorizado al sistema. Este tipo de falla puede ser explotada para la ejecución remota de comandos, la modificación de archivos críticos o incluso la toma de control total del servidor por parte del atacante.

Figura 18

Ejemplo de Carga de Archivos no controlada



En la figura 18 se observa que el archivo malicioso se ha subido exitosamente al servidor, debido a la falta de validación de extensiones en el proceso de carga. Esta vulnerabilidad permite que un atacante aloje y ejecute código arbitrario en el sistema, lo que puede derivar en la obtención de acceso no autorizado, la manipulación de archivos, la extracción de información sensible o incluso el compromiso total del servidor. La ejecución de ‘shell.php’ representa un grave riesgo de seguridad, ya que podría ser utilizada para ejecutar comandos remotos y escalar privilegios dentro del entorno comprometido.

8.6. Falta Autenticación para recurso web

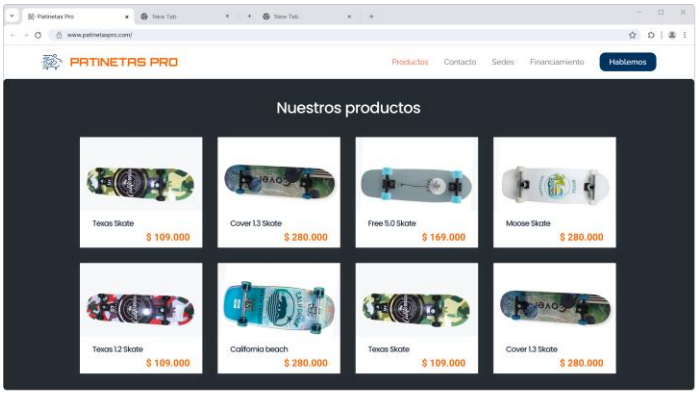
La autorización se refiere a decidir si un usuario, con una identidad específica, tiene permiso para acceder a un recurso concreto, basándose en sus privilegios y las reglas de

acceso establecidas para ese recurso.

Si no se establecen o se aplican correctamente estas reglas de acceso, los usuarios podrían llegar a acceder a información o realizar acciones que no deberían. Esto puede desencadenar diversos problemas, desde la revelación no deseada de datos hasta la interrupción de servicios o la ejecución de comandos no autorizados. Un ejemplo de esto es el acceso no restringido al contenido de las APIs. Es fundamental implementar y verificar adecuadamente los controles de acceso para evitar tales vulnerabilidades.

Figura 19

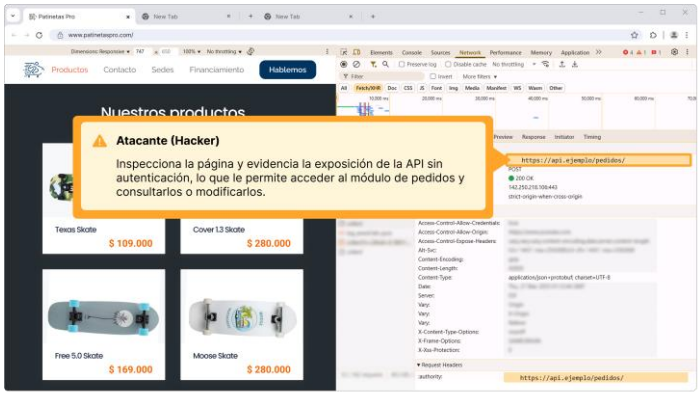
Ejemplo de Falta Autenticación para recurso web



En la figura 19 se observa que un usuario no autenticado ha accedido a la página principal del sitio. Dado que se trata de un usuario malicioso, es probable que inicie un proceso de reconocimiento y exploración en busca de vulnerabilidades en la API del sistema. Esto podría incluir la manipulación de solicitudes, la prueba de endpoints expuestos o la explotación de configuraciones incorrectas en los controles de acceso. Si la API ca- rece de mecanismos adecuados de autenticación y validación, el atacante podría obtener información sensible, modificar datos o incluso comprometer el funcionamiento del sitio.

Figura 20

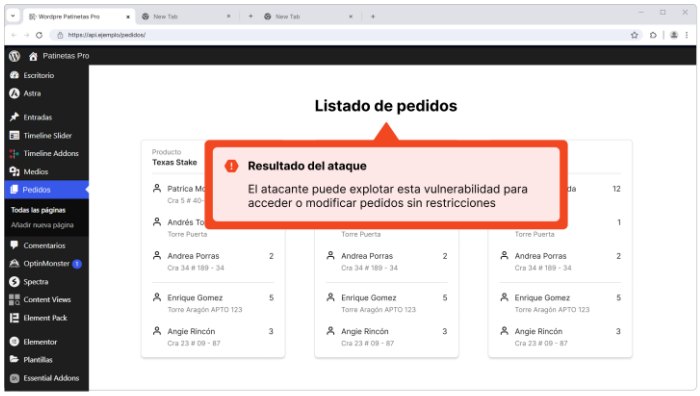
Ejemplo de Falta Autenticación para recurso web



En la figura 20 se observa que el usuario malicioso ha logrado identificar la API dentro de las respuestas del servidor utilizando la herramienta 'Inspeccionar' del navegador. Este hallazgo le permite al atacante dirigir sus esfuerzos hacia la explotación de posibles vulnerabilidades en la API, como la falta de autenticación, endpoints expuestos o configuraciones incorrectas. A partir de este punto, el atacante podría intentar extraer información sensible, manipular datos o incluso obtener acceso no autorizado al sistema, ampliando significativamente el alcance del ataque.

Figura 21

Ejemplo de Falta Autenticación para recurso web



En la figura 21 se observa que el atacante ha logrado ejecutar peticiones a la API del sitio sin necesidad de autenticación. Como resultado, puede acceder a la información de los

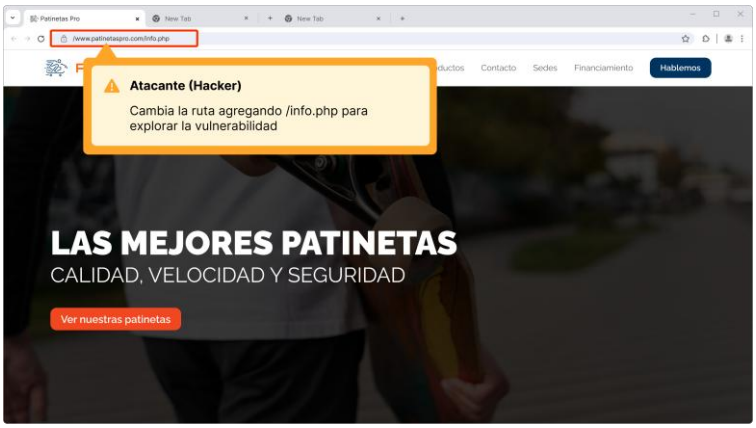
pedidos y modificarlos sin ninguna restricción, lo que representa un grave riesgo de seguridad. Esta vulnerabilidad podría permitir la manipulación de datos críticos, la alteración de transacciones o incluso la eliminación de registros, afectando la integridad y confiabilidad del sistema. La falta de mecanismos de autenticación y control de acceso en la API expone la plataforma a ataques más sofisticados y potencialmente devastadores.

8.7. divulgación de información

La vulnerabilidad de “divulgación de información” se refiere a cualquier defecto, debilidad o error en un sistema de software o hardware que podría permitir a un atacante acceder a información confidencial o sensible que no está destinada a ser pública. Esto puede incluir datos personales, credenciales de acceso, detalles internos del sistema, y más. La información obtenida puede ser utilizada para realizar ataques más complejos, como el robo de identidad, fraudes o intrusiones adicionales en el sistema.

Figura 22

Ejemplo de divulgación de información

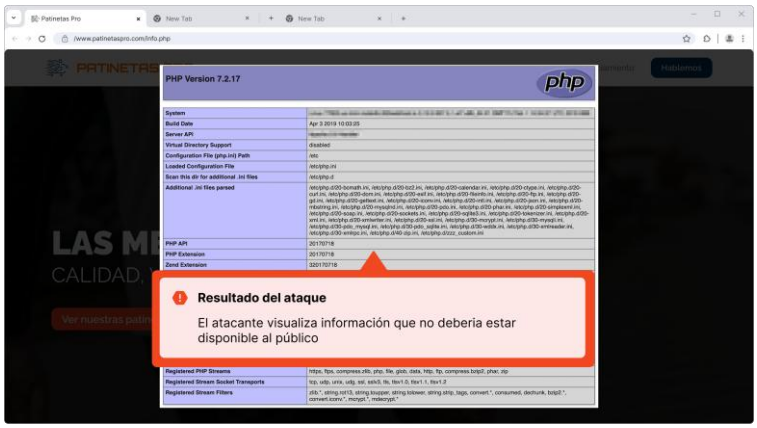


En la figura 22 se observa cómo un usuario malicioso realiza una búsqueda de la ruta ‘info.php’, la cual es ampliamente conocida por contener información sensible sobre la configuración del servidor. Este archivo suele exponer detalles como la versión de PHP, módulos instalados, variables del entorno y otras configuraciones que pueden ser

aprovechadas por un atacante para identificar vulnerabilidades específicas en el sistema. La exposición de esta información representa un riesgo significativo, ya que facilita el reconocimiento del entorno y permite el desarrollo de ataques dirigidos.

Figura 23

Ejemplo de divulgación de información



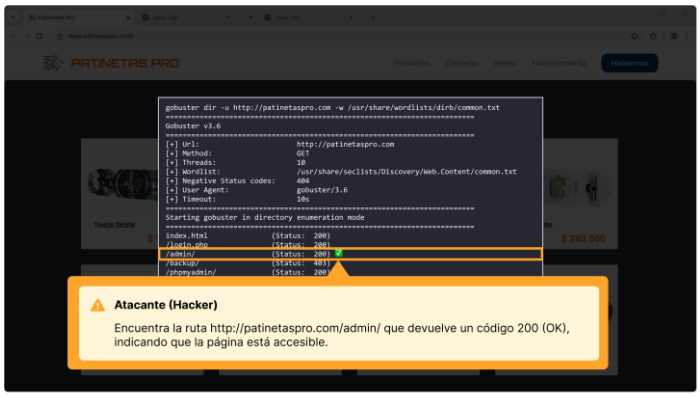
En la figura 23 se observa que el atacante ha logrado acceder a la ruta ‘info.php’, ya que esta se encuentra disponible en el servidor sin restricciones de acceso. Como resultado, el atacante ha obtenido información detallada sobre la configuración del servidor, incluyendo la versión de PHP, módulos habilitados, rutas del sistema y otras variables que pueden ser utilizadas para planificar ataques más específicos. La exposición de estos datos representa un riesgo significativo, ya que permite al atacante identificar vulnerabilidades explotables y adaptar sus técnicas de ataque en función del entorno detectado.

8.8. Consola de Administración Expuesta

Se identificó una consola de administración accesible desde internet sin medidas de protección. Esto representa un riesgo significativo, ya que un malintencionado podría acceder y realizar diversos ataques. Es crucial restringir el acceso a esta consola para garantizar la seguridad del sistema.

Figura 24

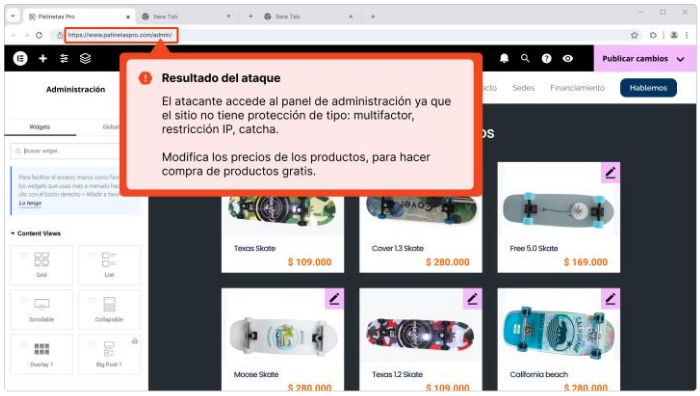
Ejemplo de Consola de Administración Expuesta



En la figura 24 se observa la consola del atacante ejecutando un escaneo de directorios mediante la herramienta Gobuster [47]. Como resultado, se ha identificado que la ruta '/admin/' devuelve una respuesta HTTP 200, lo que indica que está accesible desde el navegador. Esto representa un posible riesgo de seguridad, ya que, si el directorio de administración no cuenta con mecanismos adecuados de autenticación o restricciones de acceso, un atacante podría intentar explotarlo para obtener control sobre el sistema, acceder a información sensible o realizar modificaciones no autorizadas.

Figura 25

Ejemplo de Consola de Administración Expuesta



En la figura 25 se observa que la consola de administración del sitio está expuesta y accesible sin restricciones aparentes. Esta situación representa un riesgo significativo, ya que,

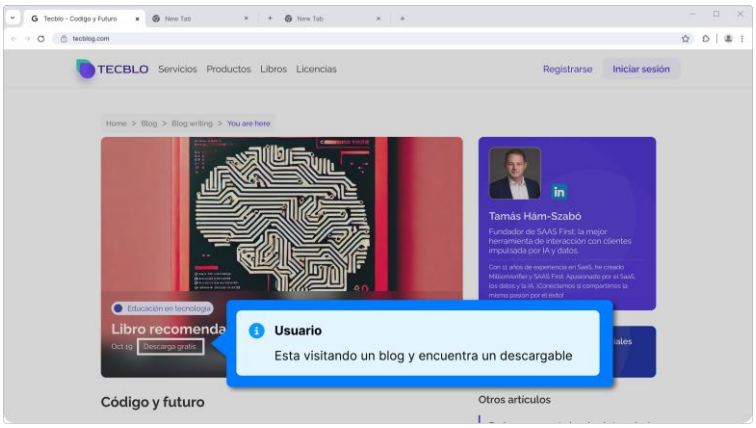
al no contar con medidas de seguridad adecuadas, podría ser vulnerable a ataques de fuerza bruta. Un atacante podría intentar adivinar credenciales de acceso mediante múltiples intentos automatizados, lo que eventualmente le permitiría obtener control sobre la administración del sistema. La falta de mecanismos como autenticación multifactor, restricciones por IP o bloqueo de cuentas tras varios intentos fallidos aumenta la probabilidad de un acceso no autorizado, comprometiendo la seguridad del sitio.

8.9. ClickJacking

El servidor web no tiene ciertas protecciones activadas que evitan que otros sitios muestren su contenido en un marco o ventana. Esto puede permitir a los malintencionados engañar a los usuarios para que hagan clic en cosas que no deberían, llevándolos a realizar acciones no deseadas.

Figura 26

Ejemplo de ClickJacking

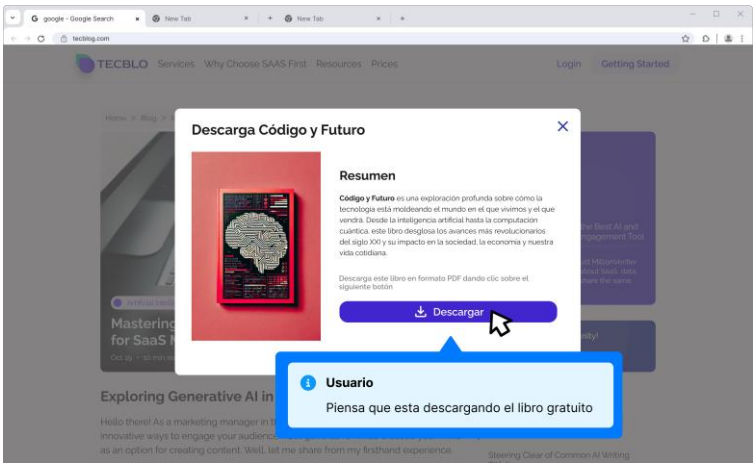


En la figura 26 se observa que un usuario está navegando por el sitio y encuentra un enlace para descargar un archivo. A simple vista, la acción parece legítima; sin embargo, si el sitio no cuenta con las protecciones adecuadas, esta funcionalidad podría ser explotada mediante un ataque de Clickjacking. En este tipo de ataque, un actor malicioso podría superponer elementos invisibles sobre el sitio real, engañando al usuario para que realice

acciones no deseadas, como descargar archivos maliciosos o ejecutar comandos sin su conocimiento.

Figura 27

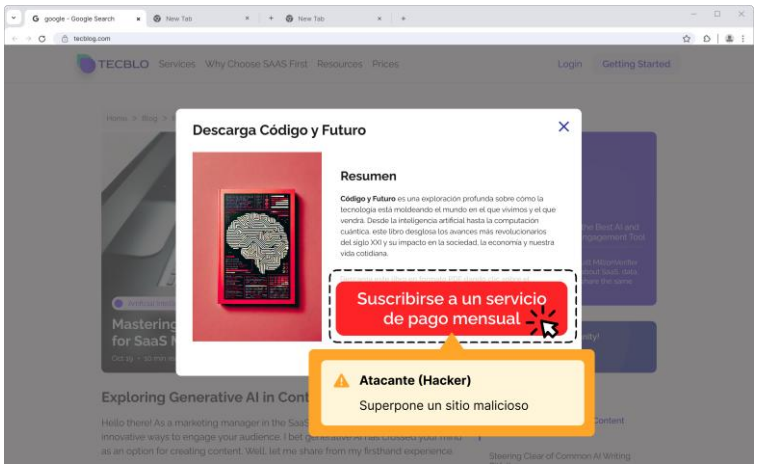
Ejemplo de ClickJacking



En la figura 27 se observa que el usuario, convencido de que el archivo corresponde a un libro gratuito legítimo, procede a descargarlo sin sospechar que podría tratarse de un engaño. Si el sitio ha sido comprometido mediante un ataque de Clickjacking, el usuario podría estar interactuando con elementos superpuestos diseñados para inducirlo a descargar software malicioso o proporcionar información sensible. Este tipo de ataque se aprovecha de la confianza del usuario en la legitimidad del sitio, facilitando la distribución de malware, el robo de credenciales o la ejecución de acciones perjudiciales sin el conocimiento del afectado.

Figura 28

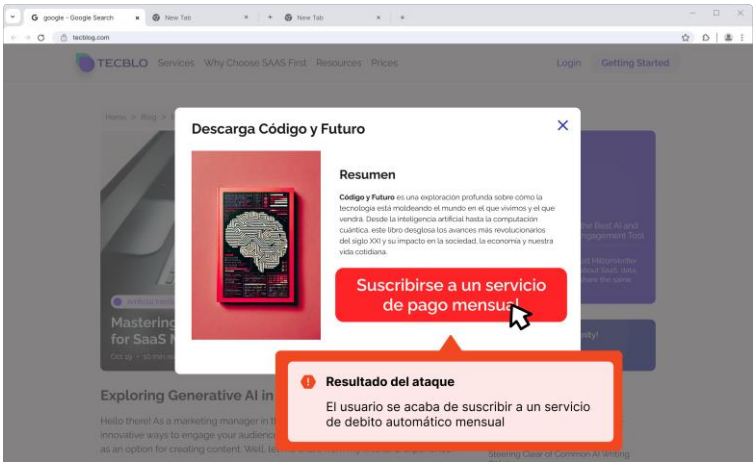
Ejemplo de ClickJacking



En la figura 28 se observa cómo un sitio malicioso ha sido superpuesto sobre el sitio original, imitando su apariencia para engañar al usuario. Debido a la similitud con la página legítima, el usuario cae en la trampa del atacante sin notar la diferencia. Este tipo de ataque, conocido como Clickjacking, permite que el usuario realice acciones sin darse cuenta, como ingresar credenciales en un formulario falso, descargar archivos maliciosos o habilitar permisos que comprometan su seguridad.

Figura 29

Ejemplo de ClickJacking



En la figura 29 se observa el resultado del ataque, donde el usuario, sin darse cuenta, se ha suscrito a un servicio de descuentos mensuales que no pertenece al sitio original. Esto permite que los atacantes obtengan acceso a su información financiera y puedan debitar

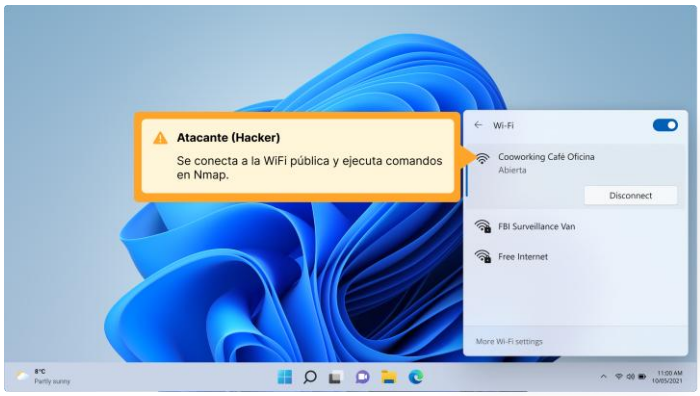
dinero de su cuenta bancaria de manera fraudulenta. Este tipo de ataque es especialmente peligroso porque explota la confianza del usuario en la legitimidad del sitio, logrando que realice acciones perjudiciales sin sospecharlo.

8.10. Bloqueo en Wifi no realizado

Esta vulnerabilidad trata cuando en nuestra infraestructura tenemos redes wifi abiertas las cuales no están debidamente configuradas permitiendo a un atacante acceder a recursos sensibles de la organización.

Figura 30

Ejemplo de Bloqueo en Wifi no realizado

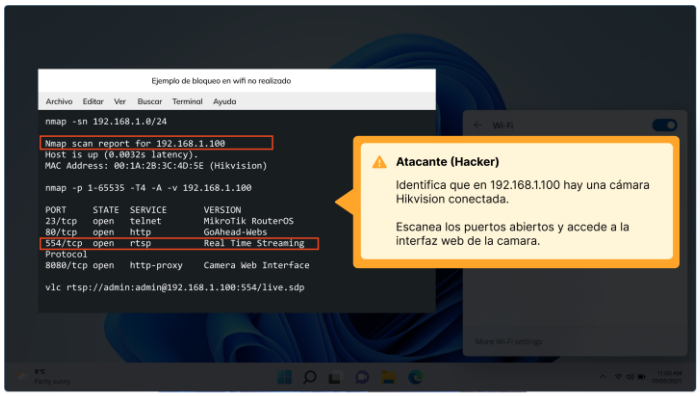


En la figura 30 se observa cómo un atacante se conecta a una red Wifi abierta, lo que le permite analizar el tráfico de la red y ejecutar herramientas de escaneo como Nmap. Nmap es una herramienta ampliamente utilizada para mapear redes y detectar dispositivos, servicios y puertos abiertos, lo que facilita la identificación de posibles vulnerabilidades.

Al no existir mecanismos de seguridad en la red, como el uso de cifrado WPA2/WPA3 o segmentación adecuada, un atacante puede obtener información sensible sobre los recursos disponibles y utilizarlos como punto de partida para ataques más avanzados, como la explotación de servicios mal configurados o la interceptación de tráfico de red.

Figura 31

Ejemplo de Bloqueo en Wifi no realizado



En la figura 31 se evidencia que el atacante ha logrado ejecutar con éxito un escaneo a todo el segmento de red, identificando dispositivos conectados y sus respectivos puertos abiertos. Como resultado, se ha detectado una cámara de seguridad accesible, lo que representa un riesgo significativo. La exposición de estos dispositivos con puertos sin restricciones podría permitir al atacante tomar el control de la cámara, interceptar su transmisión en vivo, modificar su configuración o incluso deshabilitarla.

Figura 32

Ejemplo de Bloqueo en Wifi no realizado

En la figura 32 se observa el resultado del ataque, donde el atacante ha logrado acceder con éxito a las cámaras de la organización. Esta brecha de seguridad permite al atacante visualizar en tiempo real las imágenes captadas por los dispositivos, lo que representa un grave riesgo para la privacidad y la seguridad de la institución. Además, el atacante podría manipular la configuración de las cámaras, deshabilitarlas, redirigir las

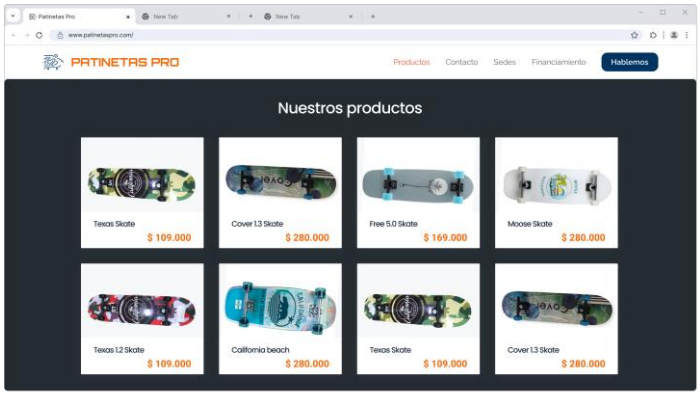
transmisiones o incluso usarlas como punto de entrada para otros ataques dentro de la red.

8.11. Inyección de Cabeceras de Host

La Inyección de Cabeceras de Host es una vulnerabilidad que permite a los malintencionados alterar o controlar el encabezado HTTP “Host” dirigido a una aplicación web. Si se explota, pueden crear enlaces dañinos, ejecutar códigos no deseados o llevar a los usuarios a sitios peligrosos. Es crucial protegerse contra esto para mantener la seguridad del sitio y de sus usuarios.

Figura 33

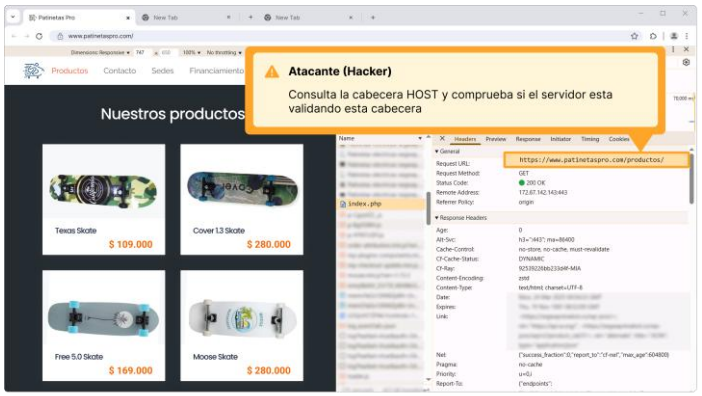
Ejemplo de Inyección de Cabeceras de Host



En la figura 33 se muestra la página web tal como la vería un usuario al cargarla en su navegador. A simple vista, la interfaz parece funcionar correctamente, mostrando los elementos habituales del sitio.

Figura 34

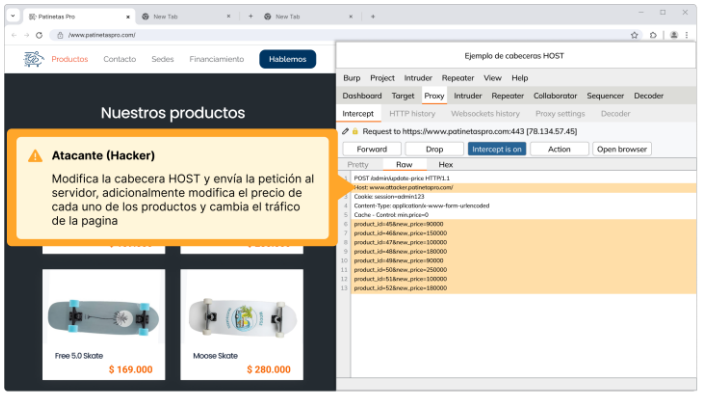
Ejemplo de Inyección de Cabeceras de Host



En la figura 34 se observa cómo el atacante utiliza la herramienta ‘Inspeccionar’ del navegador para analizar las respuestas del servidor al cargar la página. Mediante esta técnica, el atacante puede visualizar información clave contenida en los encabezados de respuesta, incluyendo el encabezado Host. Este encabezado es fundamental para la identificación del servidor y, si no se implementan medidas de seguridad adecuadas, podría exponer información sensible que facilite ataques como la manipulación de peticiones, el abuso de configuraciones incorrectas o incluso la explotación de vulnerabilidades en la infraestructura del sitio.

Figura 35

Ejemplo de Inyección de Cabeceras de Host

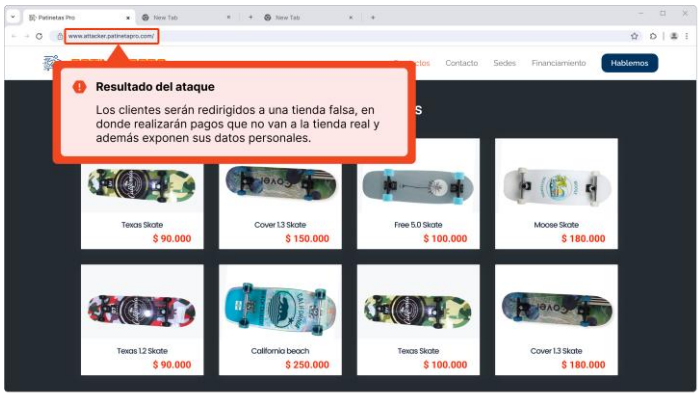


En la figura 35 se muestra cómo el atacante manipula el encabezado Host, inyectando

un valor diferente con el propósito de redirigir a los usuarios a un sitio malicioso. Este tipo de ataque, conocido como 'Host Header Injection', puede explotarse para realizar phishing, robar credenciales o distribuir malware. La falta de validaciones en el servidor para verificar los valores permitidos en el encabezado Host facilita este tipo de ataques, permitiendo que los usuarios sean engañados y expuestos a amenazas sin darse cuenta.

Figura 36

Ejemplo de Inyección de Cabeceras de Host



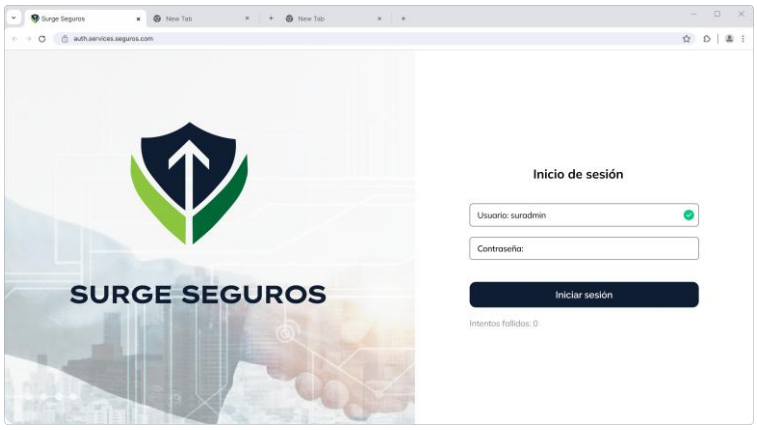
En la figura 36 se observa cómo un usuario carga la página sin darse cuenta de que en realidad se trata de una réplica maliciosa creada por el atacante. Gracias a la manipulación del encabezado Host, los usuarios son redirigidos automáticamente a este sitio fraudulento, creyendo que están accediendo al portal legitimo.

8.12. Fuerza Bruta

La aplicación permite múltiples intentos de inicio de sesión sin implementar medidas de seguridad, como captcha, para prevenir ataques. Esto deja la puerta abierta para que los atacantes intenten adivinar contraseñas mediante técnicas de fuerza bruta o "password spraying". Es vital incorporar mecanismos de protección para evitar estos riesgos.

Figura 37

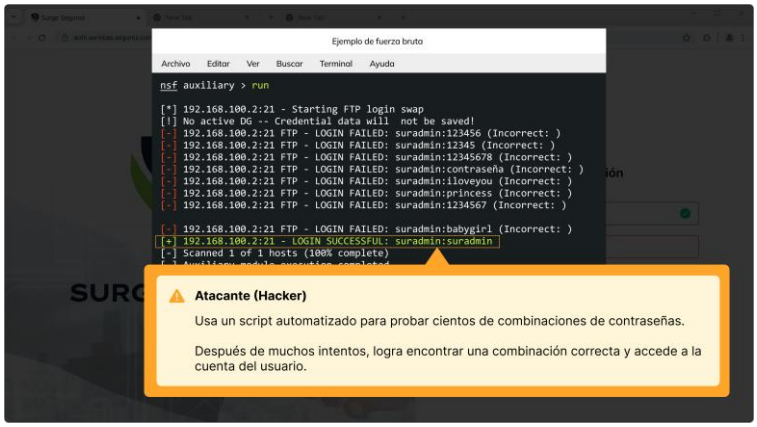
Ejemplo de Fuerza Bruta



En la figura 37 se observa una página con un formulario de autenticación que no cuenta con un mecanismo de protección como reCAPTCHA. La ausencia de esta medida de seguridad deja el sistema vulnerable a ataques de fuerza bruta, en los cuales un atacante puede automatizar intentos de inicio de sesión probando combinaciones de usuario y contraseña de manera masiva.

Figura 38

Ejemplo de Fuerza Bruta

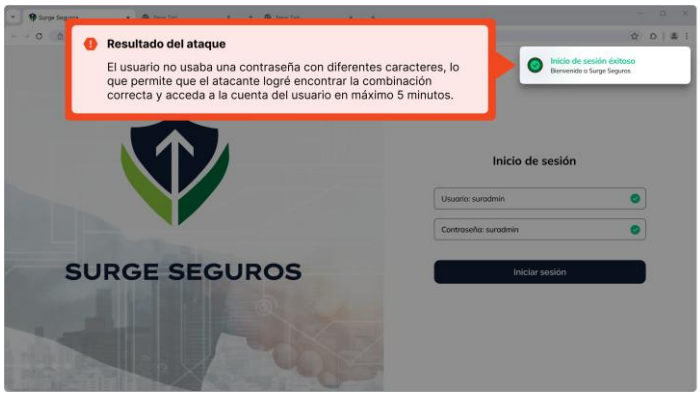


En la figura 38 se muestra cómo un atacante podría emplear Metasploit Framework junto con un diccionario de contraseñas para llevar a cabo un ataque de fuerza bruta contra el formulario de autenticación. Al no contar con mecanismos de protección como reCAPTCHA, límites en los intentos de inicio de sesión o bloqueo de cuentas tras múltiples intentos fallidos, el sistema se vuelve vulnerable a este tipo de ataques. Si el atacante logra adivinar una credencial débil, podrá acceder a la cuenta comprometida, lo que podría derivar en el

robo de información, la escalación de privilegios o la ejecución de acciones maliciosas dentro del sistema.

Figura 39

Ejemplo de Fuerza Bruta



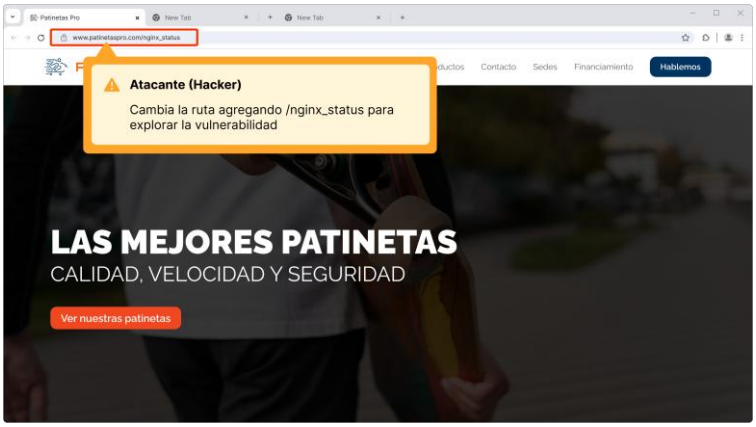
En la figura 39 se observa el resultado exitoso del ataque de fuerza bruta, donde el atacante logra autenticarse en el sistema utilizando una combinación de usuario y contraseña que estaba presente en el diccionario utilizado. Esto evidencia la vulnerabilidad del sistema ante ataques automatizados y la falta de controles de seguridad adecuados en el proceso de autenticación.

8.13. Exposición de recursos innecesarios

Se identificaron directorios o páginas web que, aunque no contienen información sensible, están accesibles al público sin autenticación. Estos recursos, aunque no presentan un riesgo inmediato, sugieren posibles problemas de gestión de configuraciones y pueden incentivar investigaciones adicionales por parte de actores malintencionados.

Figura 40

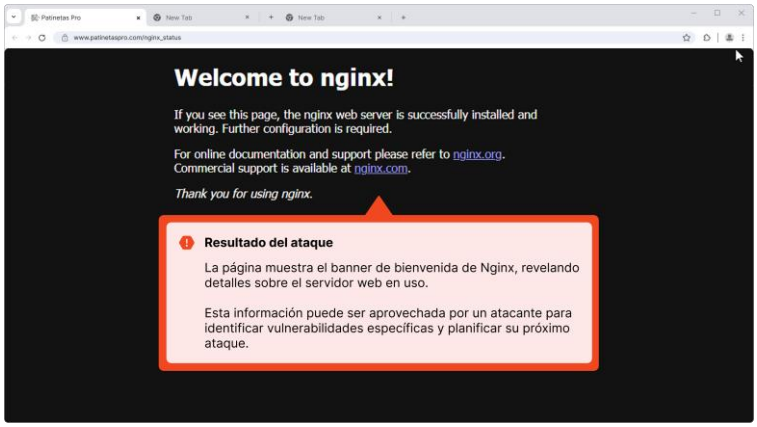
Ejemplo de Exposición de recursos innecesarios



En la figura 40 se observa cómo un atacante intenta acceder a la ruta /nginx status con el objetivo de obtener información sobre el servicio web que se encuentra en ejecución en el servidor. Esta ruta es utilizada por Nginx para proporcionar estadísticas en tiempo real sobre las conexiones activas, las solicitudes procesadas y otros detalles operativos. Si esta información es accesible sin restricciones, un atacante podría utilizarla para mapear el estado del servidor, identificar posibles puntos débiles y planificar ataques más avanzados, como la explotación de vulnerabilidades o la realización de ataques de denegación de servicio (DoS).

Figura 41

Ejemplo de Exposición de recursos innecesarios



En la figura 41 se observa que el atacante ha logrado acceder al banner de bienvenida de Nginx, el cual expone información sobre el servicio en ejecución en el servidor, incluyendo, en algunos casos, la versión específica utilizada. La divulgación de esta información representa un riesgo de seguridad, ya que permite a un atacante identificar si el

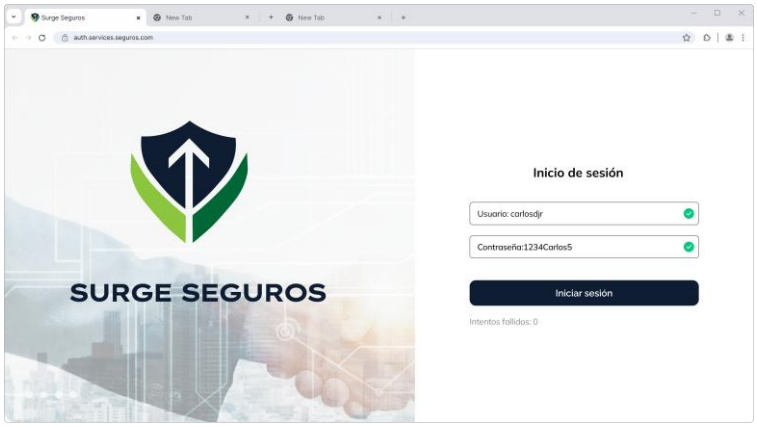
servidor está utilizando una versión obsoleta o vulnerable de Nginx, lo que facilitaría la explotación de fallos de seguridad conocidos.

8.14. Control de acceso inadecuado

El control de acceso es una medida de seguridad esencial en cualquier sistema web que garantiza que solo los usuarios autorizados puedan realizar ciertas acciones o acceder a ciertos recursos.

Figura 42

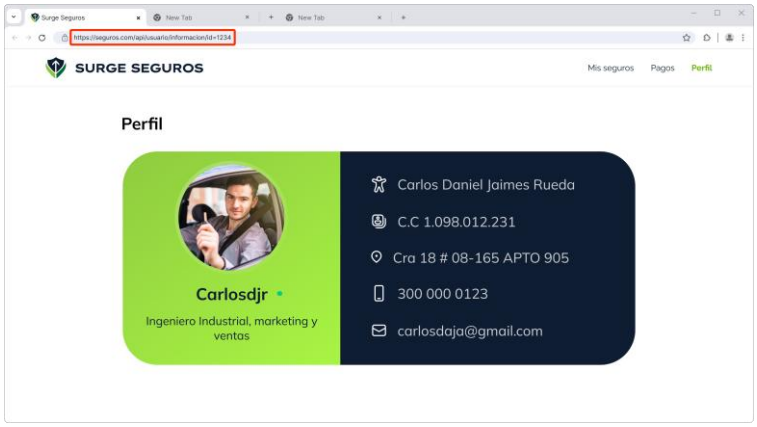
Ejemplo de Control de acceso inadecuado



En la figura 42 se observa el proceso de autenticación de un usuario en la página web. En este punto, el usuario ingresa sus credenciales, las cuales son enviadas al servidor para su validación.

Figura 43

Ejemplo de Control de acceso inadecuado

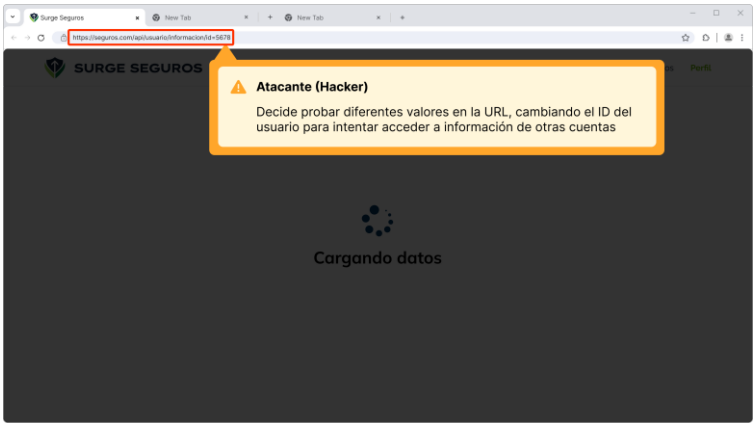


En la figura 43 se observa que la URL expone información sensible sobre el usuario

autenticado, específicamente el identificador (ID) que la API de la aplicación utiliza para gestionar las solicitudes. La exposición de este tipo de información en la URL representa un riesgo de seguridad, ya que podría permitir a un atacante manipular los parámetros para acceder a datos de otros usuarios.

Figura 44

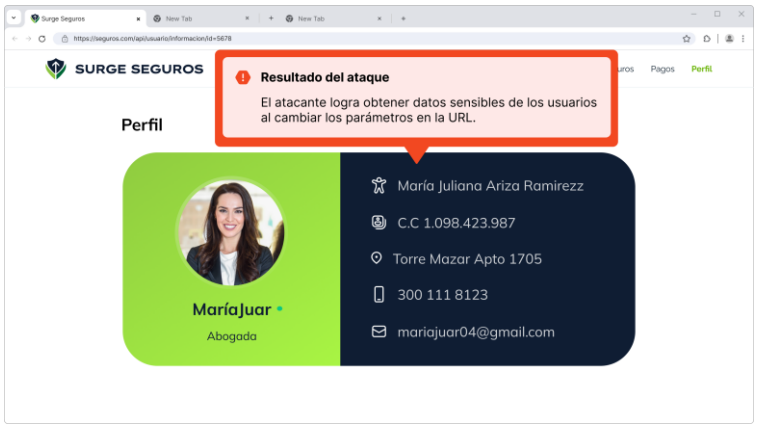
Ejemplo de Control de acceso inadecuado



En la figura 44 se observa cómo un usuario malintencionado podría manipular el parámetro **id** en la URL, probando diferentes valores con el objetivo de acceder a la información de otros usuarios. Esto permite a un atacante enumerar y acceder a registros a los que no debería tener permiso, comprometiendo la confidencialidad de los datos. Si la API no implementa controles de autorización adecuados, el atacante podría visualizar, modificar o incluso eliminar información de otros usuarios.

Figura 45

Ejemplo de Control de acceso inadecuado



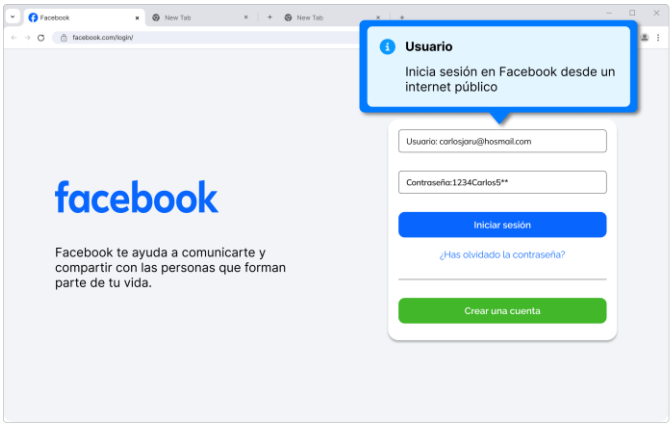
En la figura 45 se evidencia el impacto de la falta de controles de acceso adecuados en la aplicación. Como resultado de la explotación de la vulnerabilidad mencionada previamente, un atacante logró acceder a información de otros usuarios simplemente manipulando el parámetro id en la URL.

8.15. Expiración de Sesión Inadecuada

La 'Expiración de Sesión Inadecuada' es una vulnerabilidad de seguridad que se presenta cuando una aplicación web no termina o invalida adecuadamente las sesiones de usuario una vez que se ha cerrado la sesión o tras un periodo prolongado de inactividad. Esto puede permitir a un atacante aprovechar la sesión para obtener acceso no autorizado.

Figura 46

Ejemplo de Expiración de Sesión Inadecuada

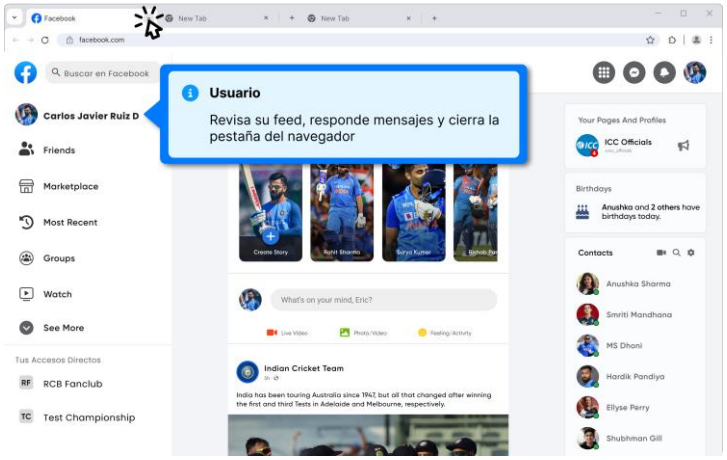


En la figura 46 se observa cómo un usuario se autentica en la página web de Facebook

desde un computador público. El uso de dispositivos compartidos para acceder a cuentas personales representa un riesgo de seguridad significativo, ya que si el usuario no cierra sesión correctamente, sus credenciales pueden quedar expuestas a otros usuarios del mismo equipo.

Figura 47

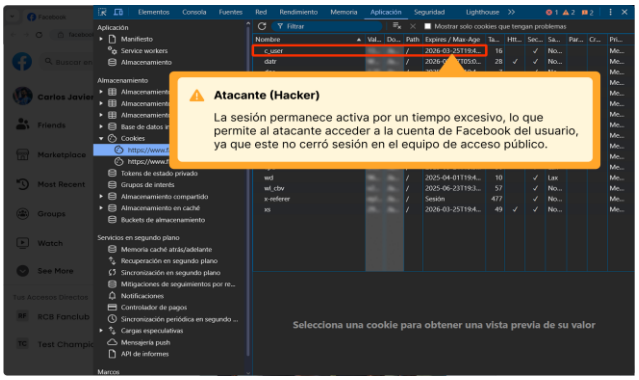
Ejemplo de Expiración de Sesión Inadecuada



En la figura 47 se observa cómo el usuario navega de manera habitual dentro de la plataforma y posteriormente cierra el navegador. Sin embargo, cerrar la ventana del navegador no siempre implica que la sesión se haya cerrado correctamente. Si la plataforma no cuenta con mecanismos de expiración de sesión o cierre automático al detectar inactividad, un atacante que acceda posteriormente al mismo computador podría restaurar la sesión utilizando cookies almacenadas o el historial de navegación.

Figura 48

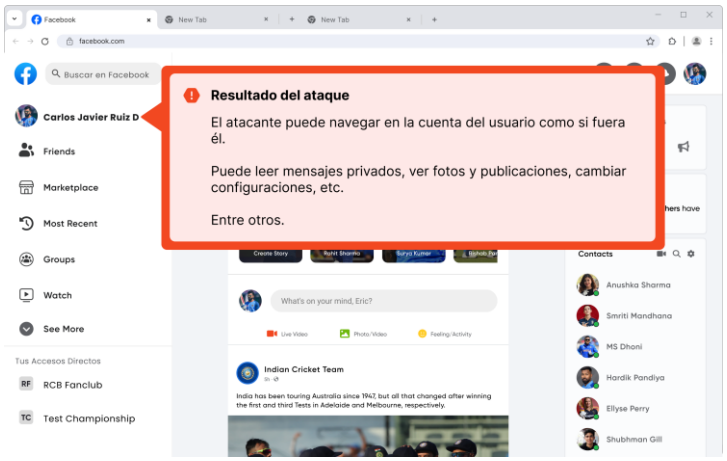
Ejemplo de Expiración de Sesión Inadecuada



En la figura 48 se evidencia un problema de configuración en la expiración de la sesión, ya que esta permanece abierta por un tiempo excesivo incluso después de que el usuario haya cerrado el navegador. Esta vulnerabilidad puede ser explotada por un atacante, quien, al acceder al mismo dispositivo, podría restaurar la sesión sin necesidad de ingresar credenciales. Esto representa un riesgo significativo, especialmente en dispositivos compartidos o públicos.

Figura 49

Ejemplo de Expiración de Sesión Inadecuada



En la figura 49 se observa el impacto de la falta de una correcta configuración en la expiración de sesión. Como resultado, el atacante logró acceder a la cuenta del usuario sin necesidad de autenticarse nuevamente, obteniendo así libre acceso a su información privada, configuraciones y posiblemente a datos sensibles como mensajes, historial de actividad y detalles financieros si la plataforma lo permite.

Una vez identificadas y explotadas las vulnerabilidades presentes en los objetivos evaluados, es fundamental documentar los hallazgos de manera estructurada y precisa (Cabe recalcar que la documentación de estos por su carácter confidencial solo será mostrada en el informe final el cual será presentado a la DTIC).

9. Consolidado de Vulnerabilidades

En esta sección se presenta un resumen de las vulnerabilidades identificadas en cada

uno de los objetivos evaluados. La siguiente tabla (2) muestra la cantidad de vulnerabilidades encontradas, categorizadas en base a su severidad: baja, media, alta y crítica, esta categorización se explica mejor en la sección 4.2.

Tabla 2

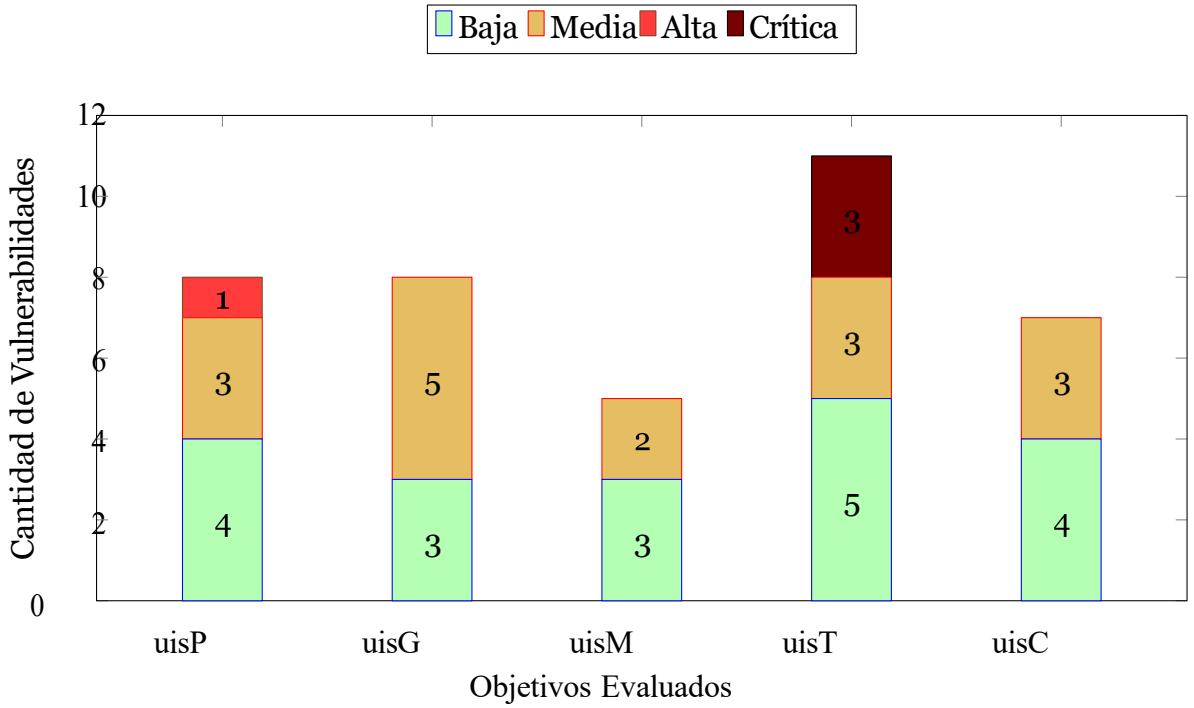
Consolidado de vulnerabilidades encontradas por severidad en cada objetivo.

Objetivo	Baja	Media	Alta	Crítica
uisP	4	3	1	0
uisG	3	5	0	0
uisM	3	2	0	0
uisT	5	3	0	3
uisC	4	3	0	0

Para visualizar estos datos, se ha generado la gráfica de barras apiladas 50, donde se representa la cantidad de vulnerabilidades en cada nivel de severidad para cada objetivo evaluado.

Figura 50

Distribución de vulnerabilidades por severidad en cada objetivo.



Adicionalmente a la gráfica anteriormente vista, tenemos la siguiente tabla donde Asociamos el código CWE de vulnerabilidad asociado a los hallazgos encontrados, este código y su función se explica mejor en la sección 4.1.

Tabla 3

Consolidado de códigos CWE asociados a las Vulnerabilidades.

Vulnerabilidad	**CWE**	**Referencia**
Listado de Directorios	CWE-548: Exposure of Information Th- rough Directory Listing	https://cwe.mitre.org/data/definitions/548.html
Acceso no autorizado a Recursos Sensibles	CWE-200: Exposure of Sensitive Infor mation to an Unauthorized Actor	https://cwe.mitre.org/data/definitions/200.html
Carga de Archivos no Controlada	CWE-434: Unrestricted Upload of File with Dangerous Type	https://cwe.mitre.org/data/definitions/434.html
Falta de Autenticación para Re-curso Web	CWE-306: Mis-Sing Authentication for Critical Function	https://cwe.mitre.org/data/definitions/306.html
Divulgación de Información	CWE-200: Exposure of Sensitive Infor-mation to an Unauthorized Actor	https://cwe.mitre.org/data/definitions/200.html

Consola de Administración Expuesta	CWE-419: Unprotected Primary Channel	https://cwe.mitre.org/data/definitions/419.html
ClickJacking	CWE-1021: Improper Restriction of Rende- red UI Layers or Frames	https://cwe.mitre.org/data/definitions/1021.html
Bloqueo en WiFi no realizado	CWE-284: Improper Access Control	https://cwe.mitre.org/data/definitions/284.html
Inyección de Cabeceras de Host	CWE-644: Improper Neutralization of HTTP Headers for Scripting/Injection	https://cwe.mitre.org/data/definitions/644.html
Fuerza Bruta	CWE-307: Improper Restriction of Excessive Authentication Attempts	https://cwe.mitre.org/data/definitions/307.html
Exposición de Recursos Innecesarios	CWE-668: Exposure of Re- source to Wrong Sphere	https://cwe.mitre.org/data/definitions/668.html
Control de Acceso Inadecuado	CWE-284: Improper Access Control	https://cwe.mitre.org/data/definitions/284.html
Expiración de Sesión Inadecuada	CWE-613: Insufficient Session Expiration	https://cwe.mitre.org/data/definitions/613.html

La sección siguiente presenta el reporte correspondiente, en el cual se detallan las pruebas realizadas y sus implicaciones. Dado que este informe puede contener información

confidencial, solo se incluirán en este documento los enunciados de las vulnerabilidades sin una caracterización especifica y las recomendaciones de solución, asegurando así la protección de los datos sensibles y el cumplimiento de las normativas de seguridad establecidas.

10. Reporte y Sugerencias de Mitigación

En consonancia con el objetivo general de este proyecto—realizar un estudio evaluativo de seguridad en las Páginas web de servicios institucionales empleando pruebas de impacto, riesgo y usabilidad de las vulnerabilidades en concordancia con los principios de hacking ético— se elaboró un reporte técnico que consolida de forma exhaustiva los hallazgos obtenidos durante las fases de reconocimiento, explotación y análisis de usabilidad. El documento se estructura para cumplir tres propósitos fundamentales:

1. Documentar las vulnerabilidades detectadas. Incluye la descripción detallada de cada falla, su categoría (CWE), el vector de ataque empleado y la evidencia de explotación.
2. Calcular impacto y riesgo. Para cada hallazgo se aporta una valoración de criticidad que combina la probabilidad de explotación con las consecuencias potenciales para la confidencialidad, integridad y disponibilidad de los activos.
3. Orientar la remediación. Se proponen contramedidas priorizadas que contemplan buenas prácticas, controles defensivos y lineamientos de implementación seguros.

10.1 Distribución y confidencialidad

- El documento completo ha sido entregado únicamente a los responsables designados por la Universidad Industrial de Santander (UIS) para su recepción y custodia.
- Debido a la sensibilidad de la información recopilada, este manuscrito público excluye detalles técnicos (direcciones exactas, proof-of-concept, capturas de sesión, etc.) que puedan facilitar actividades hostiles.
- Todo acceso adicional al reporte integral debe autorizarse bajo acuerdos de

confidencialidad y siguiendo el principio de privilegio mínimo.

10.2 Contenido resumido en el documento de Reporte

Con el fin de facilitar la toma de decisiones estratégicas sin comprometer la seguridad, las secciones siguientes presentan:

a) Síntesis de riesgos: agrupación de los hallazgos por nivel de criticidad (Critico, alto, medio, bajo).
b) Recomendaciones de mitigación: recomendaciones generales o particulares que puedan mitigar los hallazgos encontrados.

10.3 Advertencia de aplicación

La ejecución de las medidas correctivas debe ser realizada exclusivamente por personal autorizado y con conocimiento profundo de los sistemas implicados. Se recomienda seguir procedimientos formales de gestión de cambios—incluyendo ambientes de prueba, ventanas de mantenimiento y planes de reversión—para evitar efectos adversos sobre la operación normal de los servicios institucionales.

En suma, el Reporte técnico constituye la pieza central del estudio, articulando la evidencia empírica obtenida mediante pruebas de impacto, riesgo y usabilidad con directrices concretas que permiten a la UIS fortalecer su postura de seguridad de manera informada y priorizada.

10.4 Recomendaciones Generales de Solución

Para mitigar las Vulnerabilidades identificadas durante la evaluación de seguridad, se presentan a continuación una serie de medidas recomendadas. La implementación de estas soluciones contribuirá a reducir la exposición de los sistemas y fortalecer la seguridad de la infraestructura tecnológica.

- Gestión de Accesos: Implementar controles de acceso estrictos mediante

autenticación multifactor (MFA) y el principio de mínimo privilegio para los usuarios.

- Protección de Datos Sensibles: Asegurar que toda la información sensible esté cifrada tanto en tránsito como en reposo, utilizando estándares criptográficos robustos.
- Monitoreo y Auditoría: Configurar registros de auditoría detallados y establecer un sistema de monitoreo continuo para la detección temprana de actividades sospechosas.
- Seguridad en Aplicaciones Web: Aplicar medidas como validación de entrada, sanitización de datos y protección contra ataques de inyección (SQLi, XSS, CSRF).
- Actualización y Parches de Seguridad: Mantener todos los sistemas, aplicaciones y dependencias actualizados con los últimos parches de seguridad.
- Concienciación y Capacitación: Capacitar periódicamente al personal en buenas prácticas de seguridad informática y concienciar sobre los riesgos asociados a ataques de ingeniería social.

La implementación de estas recomendaciones debe realizarse de manera planificada y en conformidad con las políticas de seguridad establecidas por la organización. Además, es recomendable realizar pruebas posteriores a su aplicación para validar su efectividad y garantizar la integridad de los sistemas evaluados.

10.5 Recomendaciones Particulares de Solución

10.5.1. Listado de Directorios

Descripción: Como se menciona en la sección 8.3, esta vulnerabilidad puede exponer información sensible. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Configurar correctamente permisos en el servidor web.
- Deshabilitar la opción de listado de directorios en la configuración del servidor.

- Implementar autenticación y autorización para recursos sensibles.

10.5.2. Acceso no autorizado a Recursos Sensibles

Descripción: Como se menciona en la sección 8.4, esta vulnerabilidad permite a usuarios no autorizados acceder a información o funcionalidades restringidas. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Aplicar control de acceso basado en roles (RBAC).
- Implementar autenticación multifactor.
- Realizar auditorías de accesos y permisos.

10.5.3. Carga de Archivos no Controlada

Descripción: La falta de validación en la carga de archivos puede permitir a un atacante subir archivos maliciosos. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Restringir tipos de archivos permitidos.
- Analizar archivos subidos con antivirus.
- Almacenar archivos en ubicaciones separadas del código fuente.

10.5.4. Falta de Autenticación para Recurso Web

Descripción: La ausencia de mecanismos de autenticación en recursos web puede exponer información sensible y permitir accesos no autorizados. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Implementar autenticación y autorización en todas las rutas críticas.
- Usar tokens de sesión seguros.

10.5.5. Divulgación de Información

Descripción: Como se menciona en la sección 8.7, la exposición de mensajes de error detallados y metadatos en el código puede revelar información sensible a un atacante. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Configurar mensajes de error genéricos
- Eliminar comentarios y metadatos de código en producción.

10.5.6. Consola de Administración Expuesta

Descripción: Como se menciona en la sección 8.8, las consolas de administración accesibles públicamente pueden ser objetivo de ataques, comprometiendo la seguridad del sistema. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Restringir el acceso por IP.
- Habilitar autenticación multifactor.

10.5.7. ClickJacking

Descripción: Como se menciona en la sección 8.9, un atacante puede incrustar una página web en un iframe con el objetivo de engañar a los usuarios y realizar acciones maliciosas sin su conocimiento. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Usar el encabezado HTTP X-Frame-Options con la directiva DENY o SAMEORIGIN.
- Implementar Content Security Policy (CSP).

10.5.8. Bloqueo en Wifi no realizado

Descripción: Como se menciona en la sección 8.10, la falta de restricciones en redes

Wifi-abiertas permite a un atacante conectarse y realizar ataques como el escaneo de la red en busca de dispositivos vulnerables. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Implementar WPA3 o, en su defecto, WPA2 con contraseña robusta.
- Deshabilitar redes abiertas o con WEP.

10.5.9. Inyección de Cabeceras de Host

Descripción: Como se menciona en la sección 8.11, un atacante puede manipular el encabezado Host en las solicitudes HTTP para redirigir a los usuarios a sitios maliciosos o realizar ataques de envenenamiento de caché. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Validar y sanitizar valores de host en el servidor.
- Usar encabezados de seguridad como Strict-Transport-Security.

10.5.10. Fuerza Bruta

Descripción: Como se menciona en la sección 8.12, un atacante puede utilizar herramientas automatizadas para probar múltiples combinaciones de usuario y contraseña en un formulario de autenticación hasta encontrar credenciales validas. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Implementar bloqueo de cuenta tras múltiples intentos fallidos.
- Usar autenticación multifactor.

10.5.11. Exposición de Recursos Innecesarios

Descripción: Como se menciona en la sección 8.13, un atacante puede obtener información sensible del servidor, como versiones de software o configuraciones expuestas, mediante banners de bienvenida o rutas accesibles públicamente. Para mitigar este riesgo, se

recomienda:

Recomendaciones:

- Deshabilitar servicios y puertos innecesarios.
- Implementar listas de control de acceso (ACL).

10.5.12. Control de Acceso Inadecuado

Descripción: Como se menciona en la sección 8.14, un atacante puede manipular parámetros en la URL o en solicitudes a la API para acceder a información de otros usuarios debido a una falta de controles de acceso adecuados. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Implementar control de acceso basado en roles (RBAC).
- Revisar regularmente los permisos de usuario.

10.5.13. Expiración de Sesión Inadecuada

Descripción: Como se menciona en la sección 8.15, si una sesión no expira correctamente tras un tiempo de inactividad o después de cerrar el navegador, un atacante podría reutilizar la sesión para acceder sin autenticación a la cuenta de la víctima. Para mitigar este riesgo, se recomienda:

Recomendaciones:

- Configurar tiempo de expiración de sesión adecuado.
- Invalidar tokens de sesión tras inactividad prolongada.

11. Conclusiones

Se pudieron llegar a las siguientes conclusiones en base al cumplimiento de nuestros objetivos específicos propuestos.

1. Claridad de las áreas críticas y priorización efectiva: El análisis de los cinco servicios

institucionales reveló puntos de exposición recurrentes, se distinguió con precisión qué fallos exigen atención inmediata, optimizando la asignación de recursos y esfuerzos de remediación.

2. Valor añadido de las pruebas manuales de penetración: Las técnicas de explotación manual descubrieron vulnerabilidades que las herramientas automáticas pasaban por alto. Esto confirma que la simulación fiel del comportamiento de atacantes reales sigue siendo imprescindible para medir la resiliencia operativa de los sistemas.
3. Eficacia del enfoque híbrido (automatizado + manual): La combinación de escaneos automatizados con pruebas manuales incrementó la cobertura y redujo falsos positivos. El flujo híbrido permitió´ detectar vulnerabilidades de configuración masiva en minutos, mientras que el análisis manual profundizó en vectores complejos, garantizando una evaluación exhaustiva y alineada con las buenas prácticas de seguridad ofensiva.
4. Importancia de la documentación estructurada: El informe final, organizado por nivel de riesgo, impactó y usabilidad facilitó la comprensión ejecutiva y técnica de los hallazgos. Al incluir recomendaciones aceleró la toma de decisiones.
5. Mejora tangible de la postura de seguridad: La retroalimentación inmediata y las acciones correctivas tempranas derivadas del proyecto permitieron reducir significativamente la superficie de ataque, sentando las bases para un ciclo continuo de mejora y verificación.
6. Necesidad de mantenimiento y revisiones periódicas: El panorama de amenazas evoluciona rápidamente. Para sostener los beneficios obtenidos, se recomienda institucionalizar pruebas híbridas recurrentes. y reforzar la capacitación del personal en nuevas tácticas de ataque.

Finalmente, este ejercicio reafirma la validez del hacking ético como una estrategia

legítima, ética y efectiva para mejorar la seguridad de los sistemas, siempre que se realice con responsabilidad, autorización y dentro del marco legal correspondiente.

12. Trabajo futuro

Si bien el presente trabajo permitió identificar y analizar múltiples vulnerabilidades en servicios institucionales mediante metodologías de hacking ético, se abren diversas líneas de acción para trabajos futuros que complementen y amplíen los hallazgos obtenidos.

En primer lugar, sería conveniente extender la evaluación a un conjunto más amplio de servicios institucionales, incluyendo aquellos que no fueron cubiertos en esta fase inicial, con el fin de construir una visión más global del estado de seguridad de la infraestructura tecnológica evaluada.

Asimismo, se propone incorporar metodologías de evaluación continua que permitan realizar análisis de seguridad de forma periódica y automatizada, empleando herramientas de monitoreo en tiempo real, integración de escaneo en entornos CI/CD y mecanismos de respuesta ante incidentes.

Otra línea de trabajo futuro consiste en fortalecer las pruebas de ingeniería social, tales como ataques de phishing o vishing, enmarcadas dentro de simulaciones controladas y con el debido consentimiento institucional, con el fin de evaluar el factor humano en la seguridad de la organización.

Además, se sugiere el desarrollo de políticas y programas de capacitación orientados a mejorar la cultura de ciberseguridad entre los usuarios y administradores de los servicios, basados en las vulnerabilidades detectadas y los patrones de ataque observados.

Por último, se recomienda la implementación de un laboratorio institucional controlado para la realización de pruebas de penetración, donde se puedan replicar ambientes reales sin comprometer la integridad de los sistemas en producción, favoreciendo así la

formación continua y la experimentación segura.

Estas propuestas representan un punto de partida para el fortalecimiento progresivo de la seguridad institucional, apoyándose en la experiencia adquirida durante el desarrollo de este trabajo.

Referencias bibliográficas

Bingler, S., West, M., & Wilander, J. (2025, 27 de agosto). *Cookies: HTTP state management mechanism*. https://httpwg.org/http-extensions/draft-ietf-httpbis-rfc6265bis.html

Clickjacking Defense - OWASP Cheat Sheet Series. (s. f.). *OWASP Foundation*. https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

CVE - Common Vulnerabilities and Exposures. (2022). *MITRE Corporation*. https://cve.mitre.org/

CVSS v4.0 Specification Document. (s. f.). *FIRST — Forum of Incident Response and Security Teams*. https://www.first.org/cvss/specification-document

Greenbone Networks GmbH. (2024). *OpenVAS - Open Vulnerability Assessment System*. https://www.greenbone.net/en/

Greenbone Networks GmbH. (s. f.). *GreenBone Cloud Service*. https://docs.greenbone.net/GCS-Manual/gcs/en/

Hadnagy, C. (2018). *Social engineering: The science of human hacking*. Wiley.

IANA. (2024). *WHOIS protocol*. https://www.iana.org/whois

Kim, P. (2014). *The hacker playbook: Practical guide to penetration testing*. CreateSpace Independent Publishing Platform.

Lyon, G. (2009). *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Nmap Project. https://nmap.org/book/

Metasploit Unleashed. (s. f.). *OffSec*. https://www.offsec.com/metasploit-unleashed/

Nmap network scanning. (s. f.). https://nmap.org/book/toc.html

OWASP Foundation. (2017). *OWASP Top Ten 2017*. https://owasp.org/www-project-top-ten/2017/Top_10

OWASP Foundation. (2019). *Information disclosure in web applications*.

https://owasp.org/www-community/Information_Leakage

OWASP Foundation. (2019). *WiFi security in web applications*. https://owasp.org/www-community/Wireless_Security

OWASP Foundation. (2020a). *Access control risks and best practices*. https://owasp.org/www-community/Access_Control

OWASP Foundation. (2020b). *Encryption weaknesses in web applications*. https://owasp.org/www-community/vulnerabilities/Weak_Cryptography

OWASP Foundation. (2020c). *Host header injection*. https://owasp.org/www-community/attacks/Host_header_injection

OWASP Foundation. (2020d). *Misconfiguration in web applications*. https://owasp.org/www-community/Improper_Configuration

OWASP Foundation. (2020e). *Password security in web applications*. https://owasp.org/www-community/controls/Password_Strength

OWASP Foundation. (2020f). *Securing cookies in web applications*. https://owasp.org/www-community/controls/SecureCookieAttribute

OWASP Foundation. (2020g). *Software update security in web applications*. https://owasp.org/www-project-automated-software-updates/

OWASP Foundation. (2021a). *Brute force attack prevention*. https://owasp.org/www-community/attacks/Brute_Force

OWASP Foundation. (2021b). *File upload security in web applications*. https://owasp.org/www-community/controls/File_Upload_Security

OWASP Foundation. (2021c). *Securing administrative consoles in web applications*. https://owasp.org/www-project-secure-admin-console/

OWASP Foundation. (2021d). *Session management security*. https://owasp.org/www-community/Session_Management

OWASP Foundation. (s. f.-a). *OWASP Web Security Testing Guide*. https://owasp.org/www-project-web-security-testing-guide/

OWASP Foundation. (s. f.-b). *WSTG - Session management testing: Cookies attributes*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes

OWASP Foundation. (s. f.-c). *WSTG - Session management testing: README*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/README

PortSwigger Ltd. (2024). *Burp Suite: The leading web vulnerability scanner*. https://portswigger.net/burp

PTES. (s. f.). *PTES technical guidelines*. http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

Rapid7, LLC. (2024). *Metasploit framework*. https://www.metasploit.com

Rescorla, E. (2018). *SSL and TLS: Designing and building secure systems*. Addison-Wesley Professional.

Scarfone, K. A., & Mell, P. M. (2007). *Guide to intrusion detection and prevention systems (IDPS)* (NIST SP 800-94). National Institute of Standards and Technology. https://doi.org/10.6028/nist.sp.800-94

Shodan. (2024). *Shodan: The search engine for the Internet of Things*. https://www.shodan.io

Stoneburner, G., Goguen, A., & Feringa, A. (2002). *Risk management guide for information technology systems* (NIST SP 800-30). National Institute of Standards and Technology.

Sullo, C. (2024). *Nikto2 web server scanner*. https://cirt.net/Nikto2

Zimmermann, P. (2019). Legal and ethical implications of hacking. *Journal of Cybersecurity*

Ethics.

epi052. (2024). *Feroxbuster: A fast, simple, recursive content discovery tool*. https://github.com/epi052/feroxbuster

Martorella, C. (2024). *theHarvester: E-mail, subdomain and people names harvester*. https://github.com/laramies/theHarvester

Reeves, O. (2024). *Gobuster: Directory/file, DNS and VHost busting tool written in Go*. https://github.com/OJ/gobuster

CWE - Common Weakness Enumeration. (s. f.). *MITRE*. https://cwe.mitre.org/