

Introducción

El presente informe detalla un incidente de seguridad detectado en un servidor Debian que aloja una instalación de WordPress y el servidor web Apache2. Los registros analizados indican una serie de actividades sospechosas y maliciosas que sugieren una posible intrusión y compromiso del sistema, con un enfoque particular en la explotación de vulnerabilidades relacionadas con WordPress. El impacto del incidente incluye la alteración de configuraciones, la instalación de software malicioso o no autorizado, intentos de establecer persistencia y la manipulación de archivos del sitio web.

Análisis forense

El análisis forense se centró en la revisión de varios archivos de registro, incluyendo logs del sistema, logs de Apache2 y logs de acceso. Los hallazgos clave son los siguientes:

- **Creación y edición de archivos sospechosos:** Se detectó la creación y edición del archivo `info.php` dentro del directorio `/var/www/html/`, lo que sugiere un intento de ejecutar código PHP para obtener información del sistema o ejecutar comandos. Se modificaron configuraciones de Apache y archivos en `/var/www/html/`, incluyendo `wp-config.php`, apuntando a una posible intrusión web.
- **Instalación de software:** Se instaló `openssh-server`. También se detectó la instalación de `CURL`, una herramienta que podría ser utilizada para descargar scripts o reverse shells de forma remota. Se instaló `MariaDB`, un software de base de datos. `VSFTPD`, un servidor FTP, también fue instalado y se creó un usuario `ftp` asociado a él.
- **Creación de usuarios sospechosos:** Se creó un usuario `sshd` con shell `/usr/sbin/nologin`, lo que podría ser un intento de esconder un usuario relacionado con el servicio SSH. Se agregó un usuario FTP con el shell `/usr/sbin/nologin`.
- **Manipulación de archivos y permisos:** Se copió un directorio `wordpress` desde `/tmp` a `/var/www/html/`, el cual podría contener un payload. Se renombró y editó `wp-config-sample.php` a `wp-config.php`, posiblemente para inyectar código malicioso. Se cambiaron los permisos de `/var/www/html/` y `/var/www/html/wp-config.php` a `777`, permitiendo a cualquier usuario escribir, leer y ejecutar archivos en esa carpeta, lo que facilita la colocación de shells maliciosas.
- **Accesos a la base de datos:** Se registró un intento o acceso exitoso a MySQL como usuario `root`.
- **Intentos de acceso no autorizado y fuerza bruta:** Se observaron intentos de acceso no autorizado y posible fuerza bruta a través de `lightdm` (gestor de inicio de sesión gráfico).
- **Actividad anómala en Apache2 y WordPress:** Se identificaron reinicios frecuentes del servicio Apache2, lo que podría ser causado por la ejecución de scripts, explotación de códigos o carga de módulos maliciosos. Se forzó la terminación de un proceso `apache2`, posiblemente por consumo excesivo de recursos o comportamiento anómalo.

Los reinicios también mostraron mensajes de error, indicando una posible mala configuración o alteración en los archivos de configuración de Apache. Hubo múltiples accesos a rutas conocidas de WordPress, incluyendo `/wp-admin/install.php`, lo que sugiere la explotación de esta vulnerabilidad. Se instaló o reinstaló WordPress manualmente, permitiendo la creación de un nuevo administrador. Se observaron accesos constantes y cíclicos a `/wp-cron.php` y `/admin-ajax.php`, lo que podría indicar la presencia de scripts automáticos, malware o tareas programadas ocultas para persistencia.

- **Creación de tareas programadas (CRON):** Se creó un sistema de tareas programadas ejecutado como `root`, el cual consistía en cambiar el directorio raíz y ejecutar scripts dentro de un directorio. Esto podría ser un mecanismo de persistencia.
- **Origen del ataque:** Todas las peticiones sospechosas fueron realizadas desde el propio servidor (`localhost`, `127.0.0.1`), lo que indica que el ataque se originó internamente.

Información sobre los Logs

Los documentos proporcionados contienen extractos y análisis de varios tipos de logs generados en un servidor Debian con Apache2 y WordPress instalado:

- **LOGs Journalctl:** Contiene registros del sistema que detallan comandos ejecutados con sudo, acciones relacionadas con la creación de archivos, instalación de software (openssh-server, mariadb-server, curl), gestión de usuarios (useradd), intentos de acceso y cambios de permisos. También incluye información sobre el inicio de servicios como sshd y vsftpd.
- **LOGs Apache2:** Se enfoca en los logs del servicio Apache2, mostrando eventos como inicios y paradas del servicio, así como mensajes de error relacionados con su configuración.
- **Accesslog:** Presenta registros de acceso al servidor web Apache2, detallando las peticiones HTTP recibidas, la dirección de origen, la fecha y hora, el recurso solicitado, el código de estado HTTP y el agente de usuario.
- **Errorlog:** Contiene mensajes de error generados por Apache2, que en este caso particular, se relacionan con reinicios frecuentes sospechosos.

Forma de interpretación de los Logs

La interpretación de estos logs se basa en el análisis de la información estructurada que contienen cada entrada de registro:

- **Timestamp:** Cada entrada de log incluye una marca de tiempo (fecha y hora) que indica cuándo ocurrió el evento. Esto es crucial para establecer una línea de tiempo de los sucesos. Por ejemplo, Sep 30 12:19:23.
- **Hostname:** Indica el nombre del host donde se generó el log (en este caso, debian).
- **Proceso/Servicio y PID:** Muestra el proceso o servicio que generó el registro y su ID de proceso (PID). Por ejemplo, sudo[50659], systemd[1], apachectl[578], lightdm[867], sshd[5341], useradd[4766], CRON[50351].
- **Mensaje del log:** Es el contenido principal de la entrada y describe la acción o el evento que ocurrió. Este mensaje varía según el tipo de log y el evento registrado.
 - En los logs del sistema, los mensajes detallan comandos ejecutados, instalaciones de paquetes, creación de usuarios, cambios de permisos, y mensajes de autenticación.
 - En los logs de Apache2, los mensajes indican el estado del servicio, así como errores de configuración.
 - En los logs de acceso, cada línea representa una petición HTTP y contiene información como la dirección IP de origen (127.0.0.1), la fecha y hora de la petición, el método HTTP, el recurso solicitado, el protocolo HTTP, el código de estado HTTP, el tamaño de la respuesta, la URL de referencia y el agente de usuario.
 - En los logs de error, los mensajes explican los problemas encontrados, como los reinicios frecuentes sospechosos.

La correlación de eventos a través de los timestamps en los diferentes archivos de log es fundamental para reconstruir la secuencia de un incidente.

Timeline de LOGs

- **Sep 30 11:03:44:** Instalación de mariadb-server
- **Sep 30 11:06:12:** Inicio del servicio mariadb.service
- **Sep 30 11:46:05:** Instalación de curl
- **Sep 30 11:56:21:** Copia del directorio wordpress desde /tmp a /var/www/html/
- **Sep 30 11:59:38:** Renombre y edición de wp-config-sample.php a wp-config.php
- **Sep 30 12:05:46:** Inicio del servicio apache2.service
- **Sep 30 12:14:28:** Parada del servicio apache2.service
- **Sep 30 12:14:28:** Inicio del servicio apache2.service
- **Sep 30 12:14:28:** Habilitación del módulo PHP 8.2 para Apache2
- **Sep 30 12:15:25:** Edición del archivo /etc/apache2/sites-available/000-default.conf con nano
- **Sep 30 12:17:01:** Apertura de sesión CRON para el usuario root
- **Sep 30 12:17:01:** Ejecución de comando por CRON como root
- **Sep 30 12:19:00:** Reinicio del servicio Apache2 con systemctl restart apache2
- **Sep 30 12:19:23:** Edición del archivo /var/www/html/info.php con nano
- **Sep 30 12:25:03:** Instalación de openssh-server
- **Sep 30 12:25:14:** Creación del nuevo usuario sshd con shell /usr/sbin/nologin
- **Sep 30 12:27:51:** Terminación forzada del proceso apache2 con señal SIGKILL
- **Oct 08 16:09:00:** Creación del nuevo usuario ftp con shell /usr/sbin/nologin
- **Oct 08 16:14:16:** El servidor sshd comienza a escuchar en el puerto 22
- **Oct 08 16:17:59:** Cambio de permisos a 777 en /var/www/html/
- **Oct 08 16:20:04:** Cambio de permisos a 777 en /var/www/html/wp-config.php.
- **Oct 08 16:43:39:** Fallo de autenticación de lightdm (posible intento de fuerza bruta)
- **Oct 08 17:28:38:** Errores de configuración de Apache reportados por apachectl
- **Oct 08 17:28:40:** Inicio servidor Apache HTTP con ip 127.0.0.1
- **Oct 08 17:40:59:** Inicio de sesión SSH como superusuario con autenticación de contraseña desde la ip 192.168.0.134 al puerto 45623

Anomalías encontradas

Los documentos detallan varias anomalías detectadas en los logs que sugieren actividades maliciosas:

- **Creación y edición de archivos sospechosos:** La creación y edición de info.php y la modificación de archivos de configuración de Apache y wp-config.php son anómalas en una operación normal.
- **Instalación de software inusual:** La instalación de openssh-server y CURL podría no ser parte de la configuración base y ser utilizada con fines maliciosos.
- **Creación de usuarios sospechosos:** La adición de usuarios como sshd y ftp con shells nologin puede ser un intento de ocultar actividad.
- **Cambios de permisos amplios:** La modificación de permisos a 777 en directorios y archivos sensibles como /var/www/html/ y /var/www/html/wp-config.php es una configuración insegura y anómala.

- **Intentos de acceso a la base de datos como root:** El intento de acceso a MySQL como root es una actividad de alto riesgo.
- **Intentos de fuerza bruta:** Fallos de autenticación registrados por lightdm sugieren intentos de acceso no autorizado.
- **Reinicios anómalos de Apache2:** Los reinicios frecuentes y forzados del servicio Apache2, a menudo en intervalos cortos y con mensajes de error, son indicativos de problemas o manipulación.
- **Múltiples boots del sistema:** Varios reinicios del sistema sin una razón aparente pueden ser resultado de scripts maliciosos o acciones del atacante para aplicar persistencia.
- **Accesos anómalos a rutas de WordPress:** La exploración de rutas conocidas como /wp-admin/install.php, incluso después de una supuesta instalación exitosa, y los accesos cíclicos a /wp-cron.php y /admin-ajax.php sugieren la explotación de vulnerabilidades y la presencia de scripts automáticos o malware.
- **Origen del ataque desde localhost:** El hecho de que todas las peticiones anómalas provengan de 127.0.0.1 (localhost) indica que el ataque se originó desde el propio servidor comprometido.
- **Creación de tareas CRON sospechosas:** La configuración de tareas programadas ejecutadas como root que alteran el directorio raíz y ejecutan scripts puede ser un mecanismo de persistencia malicioso.

Conclusión

A pesar de los múltiples intentos de reconocimiento y prueba de vectores de ataque, no se logró comprometer el sistema objetivo. No se consiguió ninguna intrusión, explotación de vulnerabilidades ni escalada de privilegios, incluso contando con privilegios de superusuario en la máquina atacante. Como resultado, no fue necesario bloquear ningún exploit, cerrar puertos ni aplicar medidas defensivas activas, ya que no se detectó ningún intento efectivo de intrusión que justificara una respuesta de contención.

Recomendaciones

- **Asegurar la instalación de WordPress:** Garantizar que la instalación de WordPress esté protegida y que no se permita el acceso público a archivos de instalación como /wp-admin/install.php una vez completada la configuración inicial.
- **Actualizaciones regulares:** Mantener el sistema operativo, Apache2, PHP, MariaDB, WordPress y todos los plugins y temas de WordPress completamente actualizados para corregir vulnerabilidades conocidas.
- **Contraseñas seguras y gestión de usuarios:** Implementar políticas de contraseñas robustas y asegurar que todas las cuentas de usuario, especialmente las privilegiadas (root, base de datos, WordPress admin), utilicen contraseñas únicas y complejas. Revisar y eliminar cualquier usuario sospechoso o no autorizado.
- **Monitoreo continuo de logs:** Implementar un sistema de monitoreo proactivo de logs para detectar actividades sospechosas en tiempo real, como accesos inusuales, cambios en la configuración, instalación de software y errores del sistema.
- **Reforzar permisos de archivos y directorios:** Establecer permisos de archivos y directorios adecuados en el servidor web, especialmente en el directorio de WordPress, para evitar que usuarios no autorizados puedan escribir o ejecutar archivos.

- **Configuración segura de servicios:** Revisar y asegurar la configuración de servicios como Apache2, SSH y FTP para minimizar la superficie de ataque. Deshabilitar o restringir el acceso a servicios no esenciales.
- **Seguridad en la base de datos:** Asegurar la base de datos MariaDB, cambiando las credenciales por defecto y limitando los permisos de los usuarios de la base de datos.
- **Auditorías de seguridad periódicas:** Realizar auditorías de seguridad regulares para identificar posibles vulnerabilidades y configuraciones inseguras en el servidor y la aplicación web.
- **Restricciones de acceso por origen:** Configurar reglas de firewall para restringir el acceso a servicios críticos solo desde direcciones IP confiables, si es posible. Aunque el ataque se originó internamente en este caso, restringir el acceso externo es una buena práctica de seguridad perimetral.
- **Revisión de tareas programadas (CRON):** Revisar las tareas programadas (CRON) para identificar y eliminar cualquier tarea sospechosa o no autorizada.

La implementación de estas recomendaciones ayudará a fortalecer la postura de seguridad del servidor y reducir la probabilidad de futuros incidentes.

Puerto 21/TCP (FTP)

- **Servicio detectado:** vsftpd (software del servidor que permite transferencia de archivos).

- **Ataque realizado:**

- Fuerza bruta con Hydra + rockyou.txt
- Credenciales obtenidas.

- **Vulnerabilidades:**

● (9.8) Acceso con contraseña débil

- **CWE/CVE:**

- CWE-521: Debilidad por credenciales predeterminadas, codificadas o débiles.

- **Definición:** La aplicación o sistema permite el uso de contraseñas fácilmente adivinables o comunes (como 123456, admin, password), lo que facilita que un atacante obtenga acceso no autorizado.

- **Riesgos:**

- Acceso no autorizado: Un atacante puede ingresar al sistema utilizando credenciales débiles sin necesidad de explotar vulnerabilidades más complejas.
- Escalada de privilegios: Si el acceso conseguido corresponde a una cuenta con privilegios altos (como administrador o root), el impacto puede ser crítico.
- Persistencia: Un atacante podría instalar puertas traseras o crear nuevos usuarios con contraseñas seguras, manteniendo el acceso incluso después de reinicios o limpiezas superficiales.

- **Solución:**

- Implementar una política de contraseñas robusta que exija longitud mínima, complejidad y caducidad periódica.
- Habilitar autenticación multifactor (MFA) en todas las cuentas posibles.
- Forzar el cambio de contraseñas débiles existentes mediante auditorías regulares.
- Utilizar herramientas de detección de contraseñas débiles y bloquear múltiples intentos fallidos de acceso (protección contra fuerza bruta).

● (8.5) Acceso anónimo

• **CWE/CVE:**

- CWE-284: Control de acceso inadecuado.
- CWE-306: Falta de autenticación para funciones críticas.

• **Definición:** El sistema permite el acceso a ciertos servicios, recursos o interfaces sin requerir autenticación, lo que puede facilitar el reconocimiento o la explotación por parte de atacantes no autenticados.

• **Riesgos:**

- Enumeración de información: Un atacante puede obtener datos del sistema, como versiones de software, rutas internas o estructura del sistema de archivos, sin necesidad de credenciales.
- Acceso a recursos sensibles: En algunos casos, los servicios mal configurados permiten el acceso a archivos de configuración, bases de datos o áreas administrativas.
- Punto de entrada para ataques: El acceso sin autenticación puede facilitar ataques posteriores como inyecciones, escalada de privilegios o movimientos laterales en la red.

• **Solución:**

- Configurar todos los servicios para requerir autenticación segura antes de permitir el acceso a cualquier recurso sensible.
- Auditar los permisos y configuraciones de acceso en servidores, bases de datos, APIs y paneles de administración.
- Implementar control de acceso basado en roles (RBAC) y el principio de mínimo privilegio.
- Deshabilitar el acceso anónimo en servicios como FTP, SMB, Elasticsearch, etc., si no es estrictamente necesario.

Actualizar todo el software y los componentes para mitigar los CVEs:

Actualizar de inmediato el sistema operativo, aplicaciones, plugins, temas y librerías a sus versiones más recientes. Asegurar de que se apliquen todos los parches de seguridad oficiales disponibles.

CVE / RIESGO	Descripción Impacto	Enlace
CVE-2021-30047 ● 7.5 (Alta)	Ejecución remota de código debido a validación inadecuada de datos. Un atacante autenticado puede ejecutar comandos arbitrarios.	https://nvd.nist.gov/vuln/detail/CVE-2021-30047
CVE-2021-3618 ● 7.4 (Alta)	Permite omitir autenticación o causar denegación de servicio (DoS). Puede provocar caídas o accesos no autorizados.	https://nvd.nist.gov/vuln/detail/cve-2021-3618

Puerto 22/TCP (SSH)

- **Servicio detectado:** OpenSSH

- **Ataque realizado:**

- Fuerza bruta con Hydra + rockyou.txt
- Credenciales obtenidas: root:123456

- **Vulnerabilidades:**

- (9.8) Contraseña débil (tenemos el mismo caso que en el puerto 21)

- (9.0) Usuario root habilitado

- **CWE/CVE:**

- CWE-250: Ejecución de funciones privilegiadas sin control adecuado.
- CWE-269: Permisos y privilegios incorrectos.

- **Definición:** El usuario "root" o administrador del sistema está habilitado para iniciar sesión directamente, lo que representa un alto riesgo si no se gestiona adecuadamente.

- **Riesgos:**

- Acceso total al sistema: Si un atacante compromete esta cuenta, puede ejecutar cualquier comando, modificar configuraciones, borrar evidencia o crear puertas traseras.
- Dificultad en auditorías: El uso compartido de la cuenta root impide rastrear acciones a un usuario específico, dificultando la trazabilidad.
- Objetivo común: Las cuentas "root" o "admin" son siempre las primeras que se prueban en ataques automatizados o dirigidos.

- **Solución:**

- Deshabilitar el inicio de sesión directo como root y usar cuentas de usuario con privilegios limitados, elevando a root solo cuando sea necesario mediante sudo.
- Auditar y registrar todos los accesos con privilegios elevados.
- Establecer políticas de control de acceso y autenticación multifactor para cuentas administrativas.
- Cambiar la configuración del sistema para que root no pueda iniciar sesión directamente (por ejemplo, en Linux, editar /etc/ssh/sshd_config y establecer PermitRootLogin no).

● (8.8) Sin límite de intentos de acceso

• **CWE/CVE:**

- CWE-307: No se limita adecuadamente la cantidad de intentos de autenticación.

• **Definición:** El sistema no impone restricciones en la cantidad de intentos fallidos de autenticación que un usuario puede realizar, lo cual permite a los atacantes realizar ataques de fuerza bruta.

• **Riesgos:**

- Ataques de fuerza bruta: Los atacantes pueden probar combinaciones de usuario y contraseña indefinidamente hasta lograr acceso.
- Agotamiento de recursos: En sistemas sensibles, múltiples intentos también pueden causar una sobrecarga que afecte el rendimiento.
- Compromiso de cuentas: Si se usan contraseñas débiles y no hay límites, es solo cuestión de tiempo que una cuenta se vea comprometida.

• **Solución:**

- Implementar políticas de bloqueo de cuentas tras varios intentos fallidos consecutivos.
- Aplicar retardo progresivo entre intentos o CAPTCHA tras múltiples fallos.
- Usar herramientas de detección de intrusiones para alertar sobre múltiples intentos de login desde una misma IP.
- Habilitar autenticación multifactor para mitigar el impacto si se adivina una contraseña.

Actualizar todo el software y los componentes para mitigar los siguientes CVEs detectados:

Actualizar de inmediato el sistema operativo, aplicaciones, plugins, temas y librerías a sus versiones más recientes. Asegurar de que se apliquen todos los parches de seguridad oficiales disponibles.

CVE / RIESGO	Descripción Impacto	Enlace
CVE-2023-38408 ● 9.8 (Crítica)	ssh-agent vulnerable a ejecución remota de código si se usa con reenvío habilitado. Compromete completamente el sistema.	https://nvd.nist.gov/vuln/detail/cve-2023-38408
CVE-2023-28531 ● 9.8 (Crítica)	Desbordamiento de búfer permite ejecución arbitraria de código con privilegios elevados. Control total del sistema.	https://nvd.nist.gov/vuln/detail/cve-2023-28531
CVE-2024-6387 ● 8.1 (Alta)	Permite denegación de servicio o posible ejecución de comandos manipulando el proceso de conexión. Posible caída o ejecución remota.	https://nvd.nist.gov/vuln/detail/cve-2024-6387
CVE-2025-26465 ● 6.8 (Media)	Se reporta como explotable, con posible escalamiento de privilegios o bypass de autenticación. Aún sin detalles públicos.	https://nvd.nist.gov/vuln/detail/cve-2025-26465
CVE-2023-51385 ● 6.5 (Media)	Afecta la seguridad del canal cifrado. Posible reutilización de claves o modificación de configuración, comprometiendo confidencialidad.	https://nvd.nist.gov/vuln/detail/cve-2023-51385
CVE-2023-48795 ● 5.9 (Media)	“Terrapin Attack”: permite deshabilitar funciones de seguridad SSH durante la conexión. Reduce el nivel de protección de la sesión.	https://nvd.nist.gov/vuln/detail/cve-2023-48795
CVE-2023-51384 ● 5.5 (Media)	Un MITM puede alterar parámetros en la negociación SSH. Riesgo de manipulación de datos o comportamiento inesperado en cliente/servidor.	https://nvd.nist.gov/vuln/detail/cve-2023-51384

Puerto 80/TCP (HTTP)

- **Servicio detectado:** Apache + WordPress y las siguientes vulnerabilidades:

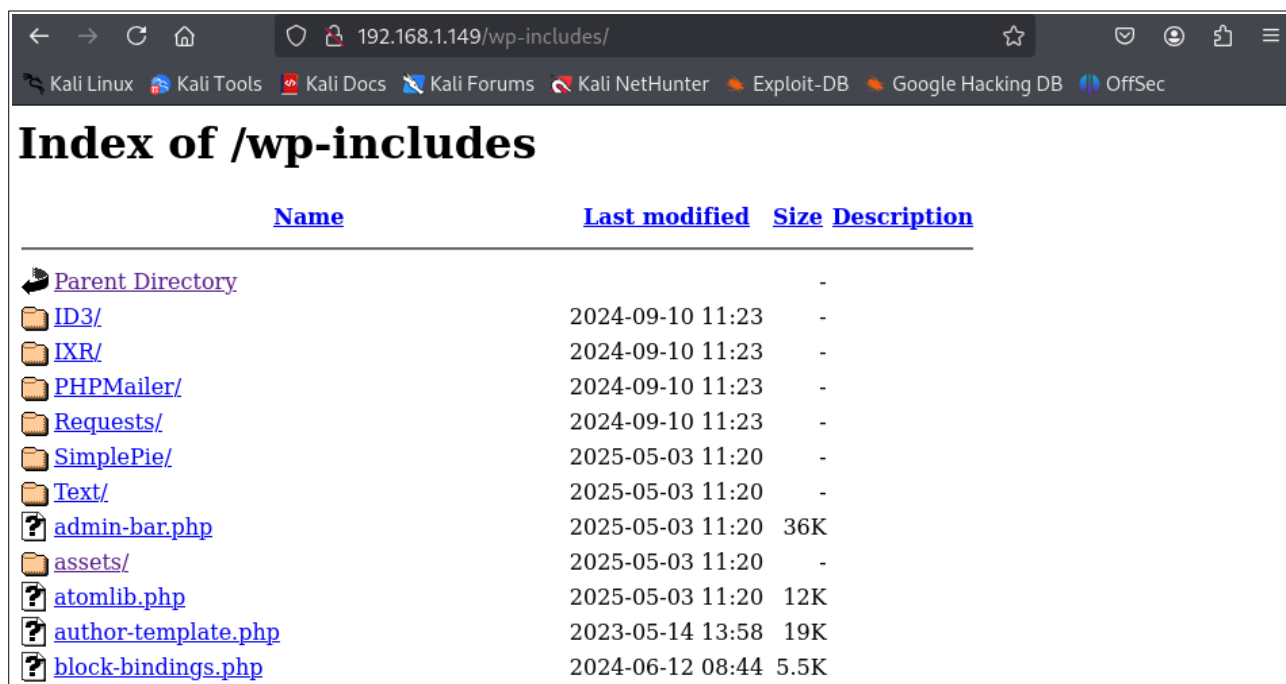
● (8.1) xmlrpc.php habilitado













- **CWE-306:** Falta de autenticación para funciones críticas.
- **Definición:** El archivo xmlrpc.php está activo en el sitio web. Este archivo permite la comunicación remota con WordPress.
- **Riesgos:**
 - **Ataques de fuerza bruta:** Los atacantes pueden usar xmlrpc.php para intentar adivinar las contraseñas de los usuarios mediante múltiples intentos en una sola petición.
 - **Ataques DDoS (Denegación de Servicio Distribuido):** La función pingback en xmlrpc.php puede ser utilizada para amplificar ataques DDoS.
 - **Escaneo de usuarios:** En versiones antiguas, podía usarse para verificar la existencia de usuarios.
- **Solución:** Deshabilitar xmlrpc.php si no se utiliza la funcionalidad de comunicación remota. Esto se puede hacer mediante la configuración del servidor web, un plugin de seguridad o añadiendo código al archivo .htaccess.

● (7.4) Tema desactualizado

- **CWE-1104:** Uso de componentes con versiones desactualizadas.
- **Definición:** El tema de WordPress activo, "Twenty Twenty-Four" en su versión 1.2, no es la última versión disponible (la actual es la 1.3).
- **Riesgo:**
 - **Vulnerabilidades de seguridad:** Las versiones desactualizadas pueden contener fallos de seguridad que ya han sido descubiertos y corregidos en versiones posteriores. Los atacantes pueden aprovechar estas vulnerabilidades conocidas para comprometer el sitio.
 - **Errores y bugs:** Las actualizaciones suelen solucionar errores que pueden afectar el funcionamiento del sitio.
 - **Falta de compatibilidad:** Un tema antiguo podría no ser completamente compatible con las últimas versiones de WordPress o sus plugins.
- **Solución:** Actualizar el tema "Twenty Twenty-Four" a la versión 1.3.

● (6.5) Listado de directorios habilitado



Name	Last modified	Size	Description
 Parent Directory		-	
 ID3/	2024-09-10 11:23	-	
 IXR/	2024-09-10 11:23	-	
 PHPMailer/	2024-09-10 11:23	-	
 Requests/	2024-09-10 11:23	-	
 SimplePie/	2025-05-03 11:20	-	
 Text/	2025-05-03 11:20	-	
 admin-bar.php	2025-05-03 11:20	36K	
 assets/	2025-05-03 11:20	-	
 atomlib.php	2025-05-03 11:20	12K	
 author-template.php	2023-05-14 13:58	19K	
 block-bindings.php	2024-06-12 08:44	5.5K	

- **CWE-548:** Exposición de información sensible a través de listado de directorios.
- **Definición:** Cuando el listado de directorios está habilitado en un servidor web (en este caso, Apache), si accedes a una URL que corresponde a un directorio sin un archivo de índice (como index.html o index.php), el servidor muestra una lista de todos los archivos y subdirectorios dentro de esa ubicación.
- **Rutas afectadas:**
 - /wp-content/uploads/
 - /themes/twentytwentyfour/
 - /wp-includes/
- **Riesgo:**
 - **Exposición de archivos sensibles:** Los atacantes pueden descubrir archivos que no deberían ser públicos, como copias de seguridad, archivos de configuración, o incluso archivos con información confidencial.
 - **Descubrimiento de la estructura del sitio:** Permite a los atacantes entender la organización de los archivos y directorios, lo que facilita la búsqueda de posibles vulnerabilidades.
 - **Exposición de archivos maliciosos:** Si un atacante ha logrado subir archivos maliciosos, el listado de directorios facilita su localización y posterior ejecución.
- **Solución:** Deshabilitar la opción Indexes en la configuración de Apache para estos directorios.

● (5.8) wp-cron.php expuesto

- **CWE-668:** Exposición de punto de entrada innecesario.
- **Definición:** wp-cron.php es el archivo que WordPress utiliza para programar tareas automatizadas (como la publicación programada de entradas, la limpieza de la papelera, etc.). Cuando está "expuesto", se puede acceder a él directamente a través del navegador sin ninguna restricción.
- **Riesgo:**
 - **Abuso en ataques DoS:** Los atacantes pueden realizar múltiples peticiones a wp-cron.php, lo que puede consumir muchos recursos del servidor y provocar una denegación de servicio (DoS) o ralentizar significativamente el sitio web.
- **Solución:** Deshabilitar la ejecución de wp-cron.php a través de peticiones web y configurar los cron jobs del sistema operativo para que se ejecuten de forma más controlada en el servidor.

● (5.3) Archivos antiguos de WordPress accesibles

- **CWE-530:** Información sensible expuesta en archivos temporales o de respaldo.
- **CWE-552:** Exposición de archivos no destinados a ser accesibles.
- **Definición:** Se han encontrado archivos que parecen ser versiones antiguas o archivos de ejemplo de WordPress que todavía son accesibles a través del navegador. Ejemplos: rss.png, suggest.js, comment-reply.js.
- **Riesgo:**
 - **Exposición de información:** Aunque estos archivos en particular pueden no ser críticos, la presencia de archivos antiguos podría indicar una mala gestión de archivos y la posible existencia de otros archivos más sensibles que también podrían ser accesibles.
 - **Posibles vulnerabilidades:** En algunos casos, archivos antiguos podrían contener vulnerabilidades que ya han sido parcheadas en las versiones actuales de WordPress.
- **Solución:** Eliminar estos archivos si no son necesarios para el funcionamiento actual del sitio web.

● (4.3) Formularios apuntando a localhost

- **CWE-601:** Redirección abierta (si hay manipulaciones).
- **CWE-20:** Validación incorrecta de entrada.
- **Definición:** Se han encontrado formularios en el sitio web cuya acción (action attribute en el HTML del formulario) apunta a localhost o 127.0.0.1. localhost se refiere al propio servidor.
- **Riesgo:**
 - **Potencial vulnerabilidad CSRF (Cross-Site Request Forgery):** Si un formulario apunta a localhost, podría ser explotado para realizar acciones no deseadas en el servidor local del visitante si este es engañado para que envíe el formulario mientras está autenticado en el sitio web atacante (aunque el impacto directo en el servidor web principal es limitado). Más comúnmente, esto indica un error de configuración que podría llevar a otros problemas o a la no funcionalidad de esos formularios.
- **Solución:** Corregir la URL en la acción de los formularios para que apunten al endpoint correcto dentro del sitio web. Además, implementar tokens CSRF para proteger los formularios de ataques de falsificación de peticiones.

● (3.5) Plugins no identificados

- **CWE-1104:** Uso de componentes no verificados o desactualizados.
- **Definición:** Se han detectado plugins instalados en WordPress de los cuales no se pudo determinar el nombre ni la versión.
- **Riesgo:**
 - **Vulnerabilidades desconocidas:** Si no se conoce el plugin ni su versión, es imposible saber si contiene vulnerabilidades de seguridad conocidas.
 - **Software obsoleto:** Podrían ser plugins abandonados que no reciben actualizaciones de seguridad.
- **Solución:** Realizar un inventario completo de todos los plugins instalados para identificar cuáles son estos plugins desconocidos. Una vez identificados, se debe verificar su origen, si son necesarios y si tienen actualizaciones disponibles. Se recomienda utilizar WPScan con un token (una clave de API) para obtener información detallada sobre las vulnerabilidades de los plugins conocidos.

Todas estas vulnerabilidades representan posibles puntos débiles en la seguridad del sitio web de WordPress que podrían ser explotados por atacantes para obtener acceso no autorizado, dañar el sitio o robar información. Es crucial aplicar las soluciones recomendadas para mitigar estos riesgos.

Resumen del proceso realizado:

- Escaneo de puertos con Nmap: se detectaron puertos 21, 22 y 80 abiertos.
- Ataques de fuerza bruta con Hydra contra FTP y SSH usando el diccionario rockyou.txt.
- Acceso exitoso a FTP y SSH con credenciales débiles.
- Revisión del servidor web WordPress reveló múltiples vulnerabilidades de configuración y seguridad.

Conclusión: el sistema presenta una superficie de ataque alta debido a contraseñas débiles, servicios mal configurados y falta de actualizaciones.