Análisis Integral de Seguridad Web

Internet Technology and Security

José Palacios Beortegui

Bachellor of Science / Applied Computing

23/04/2026

Índice

1. Introducción

El desarrollo de servicios web ha transformado la gestión de información sensible y, con ello, ha incrementado de forma exponencial la superficie de ataque disponible para actores maliciosos. Como señala OWASP (Open Web Application Security Project), las vulnerabilidades en

aplicaciones web siguen siendo el vector de ataque más frecuente en incidentes de seguridad a escala global (OWASP Foundation, 2021). El caso Equifax de 2017, en el que la explotación de una vulnerabilidad de inyección permitió la exfiltración de datos personales de 147 millones de personal ilustra con claridad las consecuencias reales de los fallos en la seguridad web (Fruhlinger, 2020).

Uno de los aspectos más importantes, es el de la falsa confianza donde los sistemas más comprometidos no lo son por ataques de alta sofisticación, sino porque sus propios diseñadores asumieron que los datos de entrada serian siempre legítimos, que los usuarios internos eran confiables, o que la oscuridad equivalía a seguridad. En este ensayo se abordan cuatro dimensiones críticas de la seguridad web directamente relacionadas con esta premisa: los permisos incorrectos en servidores, las inyecciones SQL, el robo de credenciales y el cifrado del canal mediante SSL/TLS y VPN.

2. Riesgos de una Mala Configuración de Permisos en Servidores Web

2.1.La tríada de permisos y el principio de menor privilegio

La seguridad del servidor web depende fundamentalmente de la triada de permisos POSIX: Lectura (Read), Escritura (Write), y Ejecución (Execute). Cada uno tiene implicaciones de riesgo radicalmente distintas.

El permiso de Ejecución habilita el motor dinámico; su concesión incorrecta puede permitir la ejecución de binarios arbitrarios. En notación octal, 644(archivos estáticos) y 755 (directorios) son valores de referencia.

Todo ellos se rigen por el Principio de Menor Privilegio (Principle of least Privilege, PoLP), formalizado por Saltzer y Schroeder (1975): cada proceso debe operar con el mínimo conjunto de permisos estrictamente necesario para cumplir su función. Atacarlo tiene consecuencias en cascada: un servidor Apache ejecutado como root que se vea comprometido por una vulnerabilidad en un plugin otorgará al atacante control total sobre la máquina. El CIS Benchmark fo Apache HTTP Server (Center for Internet Security, 2023) recomienda ejecutar el proceso del servidor bajo un usuario dedicado sin Shell interactiva y sin pertenencia a grupos privilegiados.

2.2 Aislamiento y contención: del chroot jail a los contenedores

La evolución del aislamiento de servidores sigue una matriz de madurez progresiva. En su forma más básica, el chroot jail restringe la vista del sistema de archivos del proceso a un subdirectorio, impidiendo que navegue hacia /etc, /bin o /usr. Sin embargo, el chroot tiene límites: un proceso con privilegios suficientes puede escapar de él, y no aísla la red ni los procesos.

Los contenedores Docker representan la solución moderna a este problema. Combinan namespaces de Linux (para aislar red, PID, montajes y UTS) con cgroups (para limitar CPU y memoria) y opcionalmente, perfiles seccomp para restringir las llamadas al sistema disponibles. El resultado es lo que puede denominarse una defensa en profundidad o arquitectura por capas: incluso si el atacante compromete el proceso del servidor, el contenedor actúa como segunda barrera, y el orquestador (Kubernetes, por ejemplo) como tercera. Esta arquitectura es la recomendada por el NIST SP 800-123 (Karen Scarfone, 2008) para entornos de producción críticos.

3. Inyección SQL: Anatomía, Tipología y Prevención

3.1 La crisis de identidad de los datos: cómo funciona la inyección

Su mecanismo esencial es una crisis de identidad de los datos: el SGBD no puede distinguir entre un dato y una instrucción cuando el desarrollador concatena la entrada del usuario directamente en el texto SQL.

El ejemplo canónico es el de un formulario de autenticación cuya consulta se construye así: SELECT * FROM users WHERE user=´” + u + ”´ AND pass=`” + p + “¨. El atacante introduce como usuario el valor ‘OR ‘ 1 ‘=’ 1, convirtiendo la condición en una tautología que siempre evalúa a verdadero y concede acceso sin credenciales válidas (Pinto, 2011). El efecto es lo que puede denominarse ‘el colapso de la sintaxis’: la estructura lógica de la consulta se rompe porque el parser SQL no distingue entre los apostrofes del dato y los delimitadores de cadena de la consulta.

3.2 Tipos de inyección: in-band, inferential y out-of-band

Las inyecciones SQL no son un fenómeno monolítico. La clasificación técnica reconoce tres categorías principales. Las inyecciones in-band son las más directas: el resultado de la consulta manipulada se devuelve directamente en la respuesta HTTP. Dentro de ellas, la variante errorbased extrae información del SGBD a través de los mensajes de error (por ejemplo, forzando un error tipo con CONVERT()); la variante UNION-based inyecta una segunda consulta que devuelve datos adicionales en las mismas columnas.

Las inyecciones inferential u out-of-band operan de forma más sutil. En la variante boolean-based, el atacante formula preguntas de si/no observando cambios en el comportamiento de la aplicación. En la variante time-based, se emplean funciones como SLEEP(5) en MySQL o WAITFOR DELAY en SQL Server: si la respuesta tarda exactamente cinco segundos, la condición inyectada era verdadera. Herramientas como splmap automatizan todo este proceso, incluyendo el fingerprint del SGBD, la enumeración de bases de datos y la extracción de tablas, lo que hace imperativa la implementación de contramedidas desde el diseño.

3.3 Consultas parametrizadas y defensa en profundidad

La defensa fundamental contra la inyección SQL es el uso de consultas parametrizadas (prepared statements), en las que los datos del usuario se transmiten al SGBD como parámetros tipados independientes de la instrucción SQL. El SGBD recibe la platilla antes que los valores, imposibilitando que estos alteren la estructura sintáctica. Este mecanismo está disponible en prácticamente todos los lenguajes: PDO en PHP, PreparedStatement en Java, y en Python con psycopg2 usando la sintaxis de marcadores de posición (%s), que aplica el escapado automático.

Complementariamente, la defensa en profundidad exige: validación de entradas mediante listas blancas, principio de menor privilegio en las cuentas de base de datos (una cuenta de lectura no puede ejecutar DROP TABLE), y el despliegue de una Web Application Firewall (WAF) como ModSecurity con el ruleset de OWASP CRS. Ninguna de estas medidas aislada es suficiente; su combinación eleva el coste del ataque hasta hacerlo prohibitivo.

4. Técnicas de Obtención de Contraseñas y Métodos de Protección

4.1. El vector matemático: hashes y sus debilidades

Las contraseñas nunca deben almacenarse en texto plano ni con funciones de hash genéricas como MD5 o SHA-1. El hash es una función de vía de sentido único: a partir del resultado es computacionalmente inolvidable recuperar la entrada original, siempre que el algoritmo sea resistente a colisiones. MD5 y SHA-1 presentan colisiones conocidas y, al ser muy rápidos de calcular, son vulnerables a ataques de fuerza bruta generados por GPU.

La técnica mas habitual para romper hashes es el ataque de diccionario, que emplea listas de contraseñas filtradas en brechas anteriores. Bases de datos como RockYou (con 14 millones de contraseñas) o las compilaciones de HaveIBeenPwned (Hunt, 2023) permiten a herramientas como Hashcat o John the Ripper probar decenas de miles de millones de hashes por segundo en hardware moderno. Una variante mas sofisticada son las Rainbow Tables, tablas precalculadas de pares (contraseña, hash) que intercambian almacenamiento por velocidad de búsqueda.

4.2. Vectores de ataque: phishing, MitM y credential stuffing

Mas allá del ataque offline a bases de datos de hashes, los atacantes disponen de vectores directos. El phishing crea paginas web falsas que imitan interfaces legitimas para inducir al usuario a introducir sus credenciales; el pharming mas sofisticado, envenena la caché DNS para redirigir el tráfico legitimo sin intervención del usuario. El ataque de intermediario (Man-in-the-Middle, MitM) intercepta el trafico no cifrado, siendo las redes Wi-Fi públicas el entorno mas propicio para su ejecución.

El credential stuffing aprovecha la reutilización de contraseñas: pares usuario-contraseña obtenidos en una brecha se prueban automatizadamente en otros servicios. Este vector fue responsable de compromisos masivos en plataformas como Netflix y Spotify, siendo su principal característica que no explota vulnerabilidades técnicas sino el comportamiento reutilizador del usuario.

4.3. Contramedidas: algoritmos de hash robustos y autenticación multifactor

El almacenamiento correcto de contraseñas requiere algoritmos diseñados específicamente para este propósito: bcrypt, Argon2 o PBKDF2. Estos algoritmos incorporan dos mecanismos de defensa: la sal (Salt) es un valor aleatorio único por usuario que se concatena con la contraseña antes del hash, invalidando las Rainbow Tables; el factor de coste (workfactor) controla el número de iteraciones del algoritmo, haciendo cada verificación costosa en tiempo aceptable para el usuario legítimo, prohibitivo para el atacante offline, NIST SP 800-63B (Grassi, 2020) recomienda Argon2id con un mínimo de 19 MiB de memoria y 2 iteraciones.

La autenticación multifactor (MFA) añade una capa que invalida las credenciales robadas sin el segundo factor. La limitación de intentos de inicio de sesión (ratelimiting), el CAPTCHA y el

bloqueo por IP tras N fallos mitigan los ataques online. La política de contraseñas debe alinearse con las recomendaciones del NIST: promover frases de paso largas, prohibir contraseñas que aparezcan en listas de brechas conocidas y no imponer cambios periódicos obligatorios, que paradójicamente incentivan la elección de contraseñas débiles y predecibles.

5. Seguridad del Canal: SSL/TLS, Criptografía Asimétrica y VPN

5.1. De la postal al candado: la necesidad de cifrar el canal

HTTP transmite la información en texto plano, cualquier intermediario puede leerla, por lo que los fallos criptográficos constituyen la segunda categoría más critica del OWASP Top Ten (2021): la ausencia de cifrado o el uso de algoritmos obsoletos expone datos sensibles en tránsito.

El protocolo HTTPS definido en el RFC 2818 (Rescorla, HTTP Over TLS., 2000) y basado en TLS (Transport Layer Security), cifra el canal entre cliente y servidor garantizando tres propiedades fundamentales: confidencialidad (el contenido es ilegible para terceros), integridad (cualquier modificación en transito es detectable mediante códigos de autentificación de mensaje, MAC) y autenticación (el servidor demuestra su identidad mediante un certificado digital firmado por una Autoridad de Certificación). Las versiones modernas TLS 1.2 y 1.3 (RFC 5246 y RFC 8446 respectivamente) eliminan vulnerabilidades presentes en SSL y TLS 1.0/1.1, que deben considerarse obsoletos.

5.2. La criptografía de clave publica: mecanismo y funcionamiento

El mecanismo central TLS es la criptografía asimétrica, basada en problemas computacionalmente difíciles como la factorización de enteros grandes (RSA) o el problema de logaritmo discreto en curvas elípticas (ECDSA/ECDH). Cada entidad dispone de un par de claves matemáticamente vinculadas: la clave pública, distribuible libremente, y la clave privada, que debe protegerse con el mismo rigor que una contraseña maestra.

El hansshake TLS 1.3 opera del siguiente modo: el cliente envía sus capacidades criptográficas; el servidor responde con su certificado digital (que contiene su clave publica y esta firmado por una CA de confianza), y ambos ejecutan un intercambio ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) para derivar una clave de sesión efímera. Esta efemeralidad es clave: significa que incluso si la clave privada del servidor se compromete en el futuro, no será posible descifrar conversaciones pasadas (propiedad denominada Perfect Forward Secrecy). El trafico posterior se cifra con AES-GCM o ChaCha20-Poly1305, algoritmos simétricos de alta eficiencia. La combinación criptográfica asimétrica para el intercambio de claves y simetría para el cifrado del flujo de datos ofrece el equilibrio optimo entre seguridad y rendimiento (Rescorla 2018).

5.3.VPN: extendiendo la seguridad a nivel de red

Las VPN (Virtual Private Networks) extienden el concepto de canal cifrado a nivel de red, creando un túnel encapsula y cifra todo el trafico IP independientemente de la aplicación que lo genere. Protocolos como OpenVPN (basado en TLS) o WireGuard (basado en criptografía de curva elíptica Curve25519) neutralizan eficazmente los ataques de MitM en redes no confiables al garantizar que incluso si un atacante captura los paquetes, solo obtendrá texto cifrado sin posibilidad de descifrado sin la clave privada correspondiente. En entornos corporativos, las VPN son

credenciales para el acceso remoto seguro, especialmente cuando los empleados operan desde redes públicas.

6. Conclusión

La seguridad web no es un estado binario, sino un proceso continuo de identificación, mitigación y monitorización de riesgos. El análisis presentado demuestra que los vectores de ataque mas prevalentes comparten un denominador común: la falsa confianza. Los servidores mal configurados confían en que solo accederán usuarios legítimos; las consultas SQL concatenadas confían en que la entrada del usuario nunca será maliciosa; los sistemas que almacenan contraseñas con MD5 confían en que la red es segura.

Frente a esta falsa confianza, la respuesta profesional es la defensa en profundidad: la triada de permisos mínimos, las consultas parametrizadas, el almacenamiento con Argon2, la autenticación multifactor y el cifrado de canal con TLS 1.3 no son medidas alternativas sino capas complementarias. El coste de implementar estas medidas correctamente desde el diseño es marginal en comparación con el coste de una brecha de datos, tanto económico como reputacional. Como profesionales de la computación, nuestra responsabilidad no es solo hacer que los sistemas funcionen, sino garantizar que funcionen de forma segura para todos los que los utilizan.

7. Bibliografía

Center for Internet Security. (2023). *Apache HTTP Server*. Obtenido de Center for Internet Security: https://www.cisecurity.org/benchmark/apache_http_server

Fruhlinger, J. (2020). Equifax data breach FAQ: What happened, who was affected, what was the impact? *CSO Online*. Obtenido de https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html

Grassi, P. G. (2020). *Digital Identity guides- Authentication and lifecycle Management*. National Institute of standarts and technology.

Hunt, T. (2023). *Have I been Pwned* . Obtenido de https://haveibeenpwned.com/

Karen Scarfone, W. J. (2008). *Guide to General Server | Recommendations of the National Institute of standarts and technology* . Washinthon: NIST SPECIAL PUBLICATION.

OWASP Foundation. (2021). *OWASP*. Obtenido de Top 10 2021: https://owasp.org/www-project-top-ten/

Pinto, S. y. (2011). *The Web Application Hacker´s Handbook: Finding and Exploiting Security Flaws* . Wiley.

Rescorla, E. (2000). *HTTP Over TLS*. Internet Engineering Task Force . Obtenido de https://www.rfc-editor.org/rfc/rfc2818#page-6

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. Internet Engineering Task Force. Obtenido de https://www.rfc-editor.org/rfc/rfc8446

Schroeder, S. y. (1975). *The protection of information in computer systems. Proceedings of the IEEE*. Obtenido de https://doi.org/10.1109/PROC.1975.9939

Temoshok, D. (Agosto de 2025). *NIST* .