INVESTIGACIÓN

OWASP *Odyssey:* una guía completa para prácticas de software seguras

OWASP Odyssey: A Complete Guide to Secure Software Practices

Cristhian Santana Ferrer*

Recibido: 06/2024 | Aceptado: 08/2024 | Publicado: 12/2024

Resumen

La seguridad en el desarrollo de aplicaciones web es un proceso integral que debe iniciarse desde las primeras etapas del ciclo de desarrollo. Los *frameworks* modernos proporcionan herramientas de seguridad importantes, pero no son suficientes por sí solos para asegurar las aplicaciones. Es esencial que se acompañe de prácticas adecuadas y de un diseño de software sólido. Las vulnerabilidades pueden aparecer no solo a nivel de *framework*, sino también debido a políticas de acceso inadecuadas, validación de datos deficiente y mecanismos de autenticación insuficientes. La adopción de las guías de la *Open Web Application Security Project* y los estándares del OWASP —*Application Security Verification Standard*— es crucial para mitigar estas vulnerabilidades, así como la familiarización con los *Common Vulnerabilities and Exposures* y *Common Vulnerabilities and Exposures* asociados a las tecnologías empleadas.

Palabras clave: ciberseguridad, OWASP, desarrollo seguro de software, inteligencia de vulnerabilidades, OSINT.

Abstract

Security in web application development is a comprehensive process that needs to be started early in the development cycle. Modern frameworks provide important security tools, but they are

1* Universidad de Ciencias Informáticas/ UCI, Facultad de Ciberseguridad.

not enough to secure applications. They must be accompanied by good development practices and sound software design. Vulnerabilities can occur not only at the framework level, but also due to inadequate access policies, poor data validation, and insufficient authentication mechanisms. Adoption of the Open Web Application Security Project guidelines and the OWASP Application Security Verification Standard standards is critical to mitigating these vulnerabilities, as is familiarity with Common Vulnerabilities and Exposures and Common Vulnerabilities and Exposures associated with the technologies used.

Keywords: *Cybersecurity, OWASP, secure software development, vulnerability intelligence, OSINT.*

Introducción

Se suele pensar que la seguridad es un proceso de la etapa de desarrollo en las aplicaciones web. Mientras que los desarrolladores suelen confiar mucho en los mecanismos que ofrecen los *frameworks*, estos no brindan protección frente a las malas prácticas durante el desurante un mal manejo de software, ausencia de mecanismos de autenticación, entre otros. Lo anterior propicia aplicaciones vulnerables en producción, quedando expuestas a ciberatacantes que pueden explotar debilidades como ataques XSS, o con bases de datos no aseguradas que pueden conducir a fugas de información.

Surge así la necesidad de llevar a cabo la presente investigación con la finalidad de exponer vulnerabilidades que se presentan en el desarrollo de las aplicaciones web y las maneras de mitigarlas siguiendo las especificaciones propuestas en las guías de la *Open Web Security Project* (OWASP) y los requerimientos de seguridad mínimos con el que deberían cumplir la mayoría de las aplicaciones siguiendo las recomendaciones de la fundación OWASP en su proyecto *Application Security Verification Standard* (ASVS). La mitigación de vulnerabilidades como inyecciones SQL, *Broken authentication*, *Broken Access Control* y filtración de información sensible serán algunas de las tratadas en la presente investigación.

De forma tal que, como parte de las tareas para atenuar las vulnerabilidades en el proceso de desarrollo de aplicaciones

web, las organizaciones deben adoptar acciones de inteligencia de vulnerabilidades de forma proactiva para prevenir futuras amenazas, frente a *exploits* maliciosos, o nuevas vulnerabilidades identificadas en los *frameworks* utilizados por estas. Para ello, un análisis y conocimiento previo de los *Common Vulnerabilities and Exposures* (CVE) y *Common Vulnerabilities and Exposures* (CVE) a través de plataformas como CVE *Details*, redes sociales, entre otros (Le, Wang et al. 2019).

Materiales y métodos

Para el desarrollo de la presente investigación se realizó una recopilación y revisión bibliográfica para la cual fueron necesarios ciertos recursos, detallados a continuación:

Open Web Security Project

La fundación OWASP es una organización sin fines de lucro que trabaja para mejorar la seguridad del software, dedicada a permitir que las organizaciones conciban, desarrollen, adquieran, operen y mantengan aplicaciones en las que se pueda confiar. Todos sus proyectos, herramientas, documentos, foros y capítulos son gratuitos y abiertos a cualquier persona interesada en mejorar la seguridad de las aplicaciones (Helmiawan, Firmansyah et al. 2020). La fundación OWASP se lanzó el 1 de diciembre de 2001 y se incorporó como una organización benéfica de los Estados Unidos el 21 de abril de 2004. (Idris, Syarif et al. 2022)

OWASP *Application Security Verification Standard*

El OWASP —*Application Security Verification Standard*— es una lista de requisitos o pruebas de seguridad de aplicaciones que puede ser utilizada por arquitectos, desarrolladores, evaluadores, profesionales de seguridad, proveedores de herramientas y consumidores para definir, construir, probar y verificar aplicaciones seguras (Wen and Katt 2023) (Li 2020).

Vulnerabilidades comunes en aplicaciones web

El OWASP Top 10 (Helmiawan, Firmansyah et al. 2020) es el informe presentado por la fundación OWASP el cual se actualiza con regularidad donde se exponen los diez riesgos más importantes que presentan las aplicaciones web (Subedi, Alsadoon et al. 2016). Este sirve de referencia para las organizaciones a la

hora de garantizar la seguridad de sus aplicaciones web. Siendo este el primer paso para fomentar de manera efectiva una cultura en el desarrollo de software seguro (Sołtysik-Piorunkiewicz and Krysiak 2020):

A01:2021-*Broken Access Control*
A02:2021-*Cryptographic Failures*
A03:2021-*Injection*
A04:2021-*Insecure Design*
A05:2021-*Security Misconfiguration*
A06:2021-*Vulnerable and Outdated Components*
A07:2021-*Identification and Authentication*
A08:2021-*Software and Data Integrity Failures*
A09:2021-*Security Logging and Monitoring Failures*
A10:2021-*Server-Side Request Forgery*

Inteligencia de vulnerabilidades

La gestión de la superficie de ataque es de vital importancia para las organizaciones para mantener un sólido proceso de ciberseguridad. Es aquí donde entra en juego la inteligencia de amenazas la que debe contar como parte de las acciones que las organizaciones deberían emprender en el desarrollo de aplicaciones web de manera segura, gracias a que la ya mencionada inteligencia proporciona información crítica sobre posibles ataques y vulnerabilidades que pueden presentar los *frameworks* y tecnologías empleadas en las mismas. Para ello, como se aprecia en la figura 1, se puede contar con numerosas plataformas como CVE *Details*, *Exploit Database*, CVE *Website*, *National Vulnerability Database* (NVD) (Cascavilla, Tamburri et al. 2021).

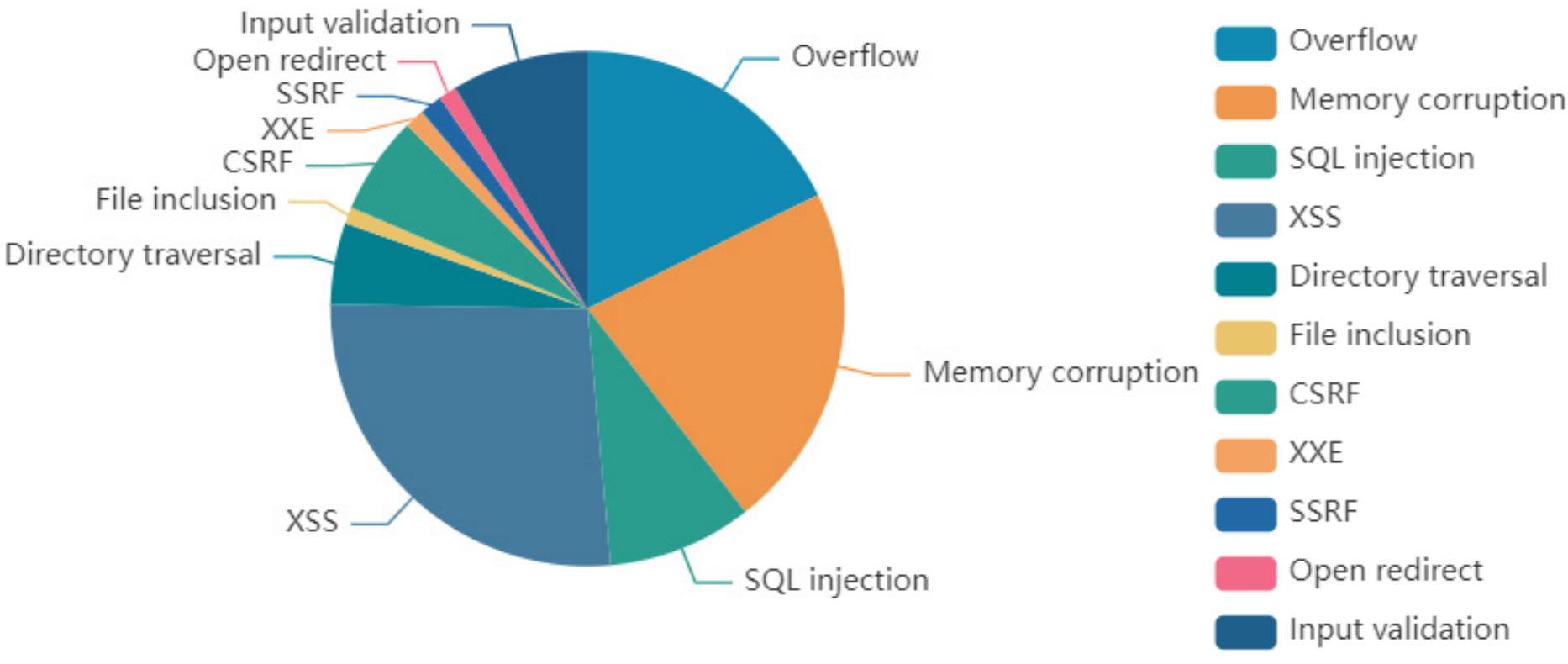

Figura 1. Vulnerabilidades más comunes registradas en cvedetails.com

Aplicaciones Web

En esta sección se dará la descripción de la estructura con la que cuentan las aplicaciones web generalmente y las tecnologías más comunes que se utilizan en estas.

Frontend

El *Frontend* es utilizado para mostrar el contenido principal de un sitio web, es lo que el usuario ve. En el contexto de los sitios web se refiere a todas las tecnologías que se ejecutan en el navegador y se encargan de brindar interactividad con el sitio. Por esto se emplean varias tecnologías como HTML-5, CSS, *JavaScript*. Los sitios web que solo cuentan con el *Frontend* suelen ser propensos a ataques de *Cross-Site-Scripting* (XSS). Algunos *Frameworks* modernos como Angular y *ReactJs* cuentan con mecanismos incorporados para prevenir los ataques XSS (Lala, Kumar et al. 2021).

Backend

El *Backend* es necesario a la hora de implementar sitios web dinámicos. Suele encargarse de manejar la lógica y el procesamiento de datos indispensables para que todo funcione de manera correcta y segura, ocupándose de tareas como almacenar y leer datos de una base datos, procesar formularios, autenticar usuarios y gestionar la seguridad del sitio. Por este motivo existen numerosos lenguajes y *frameworks* utilizados, como *Python* (*Django*, como aparece en la figura 2, *Flask*, *FastAPI*, como en la figura 3), C# (ASP.NET como se ve en la figura 4), *Java* (Spring Framework, como aparece en la figura 5), PHP (*Symfony*, *Laravel*), entre otros (Lala, Kumar et al. 2021).

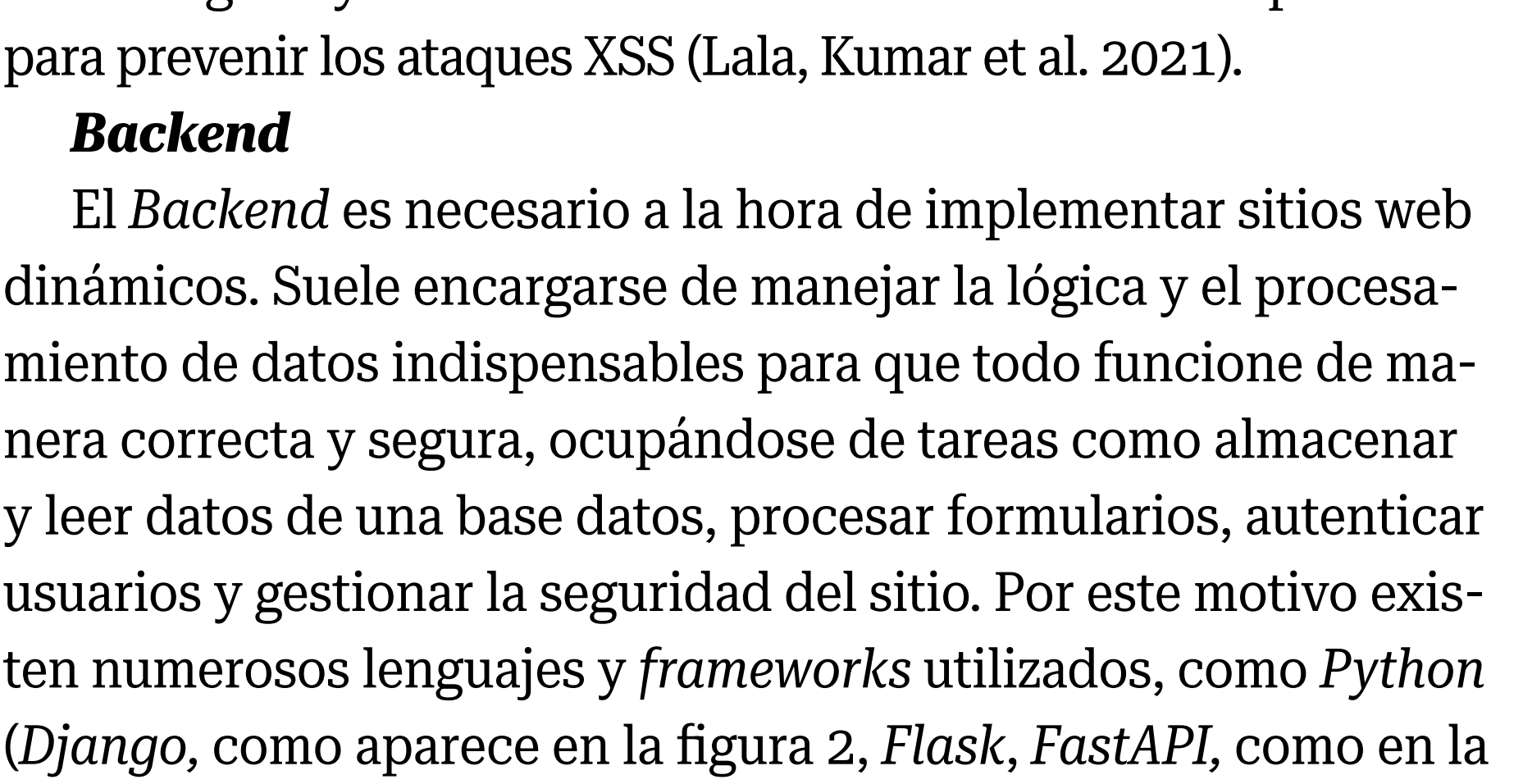

Figura 2. CVE Details de Django Framework

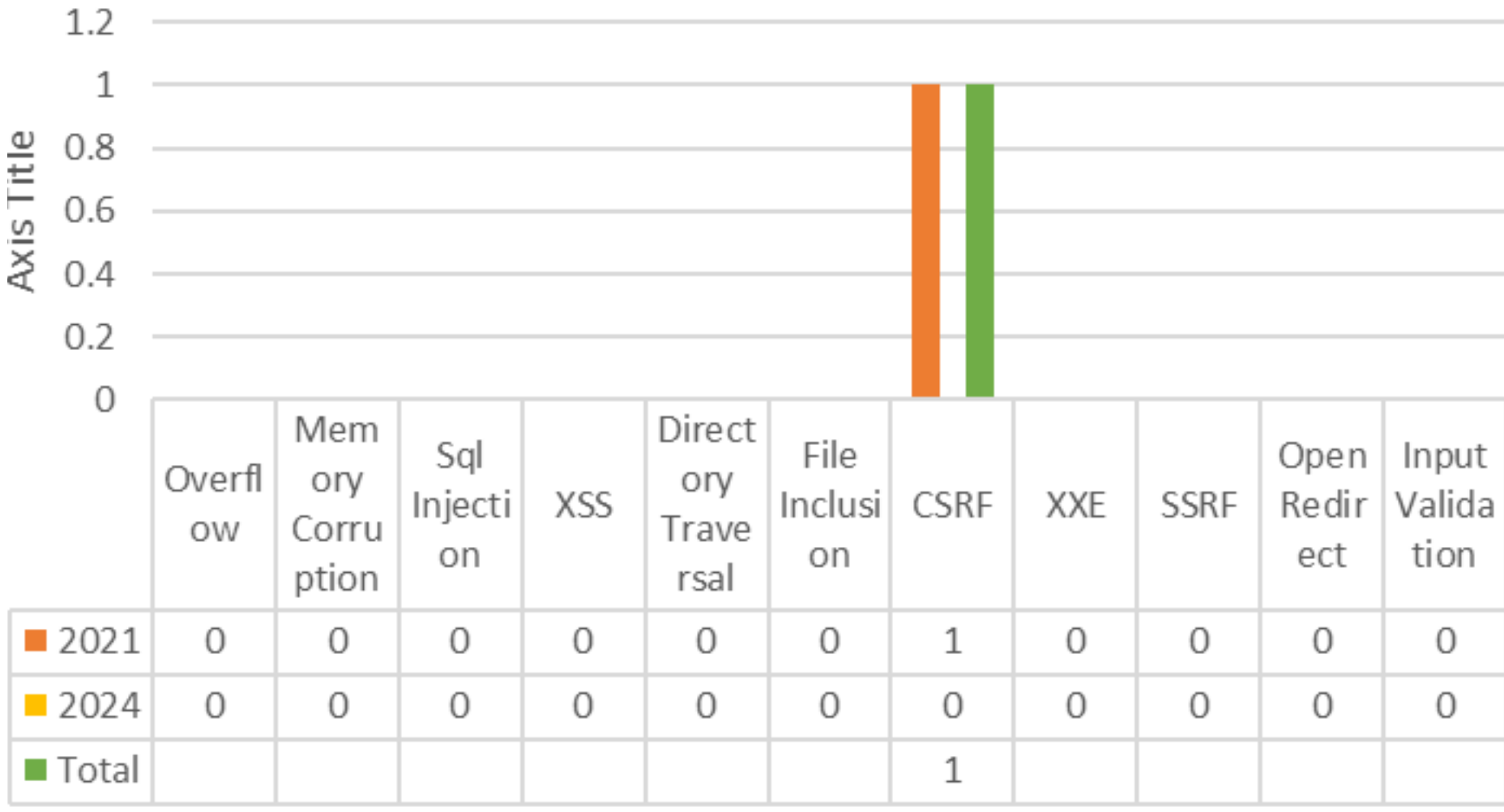

Figura 3. CVE Details de FastAPI

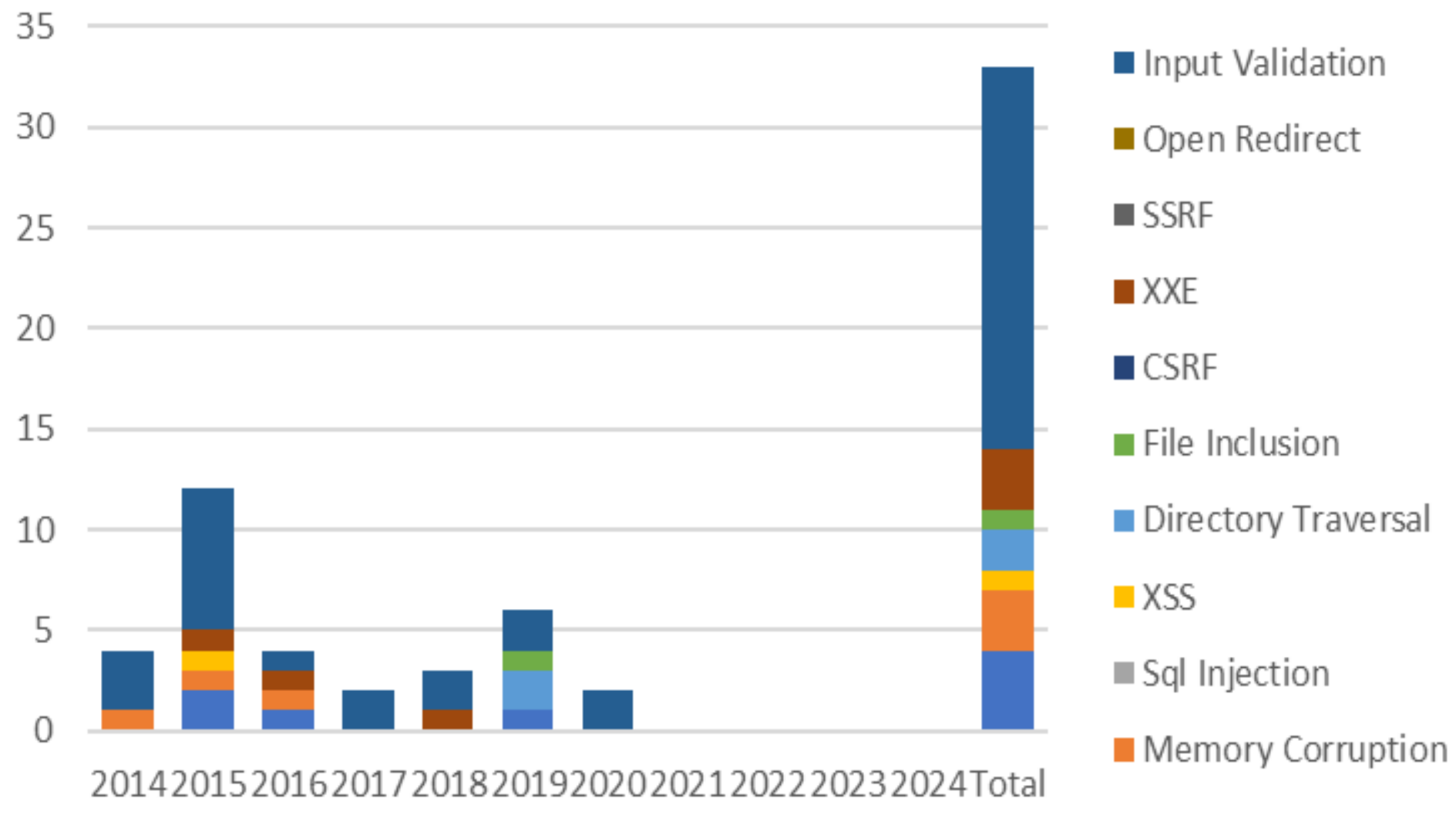

Figura 4. CVE Details de .NET Framework

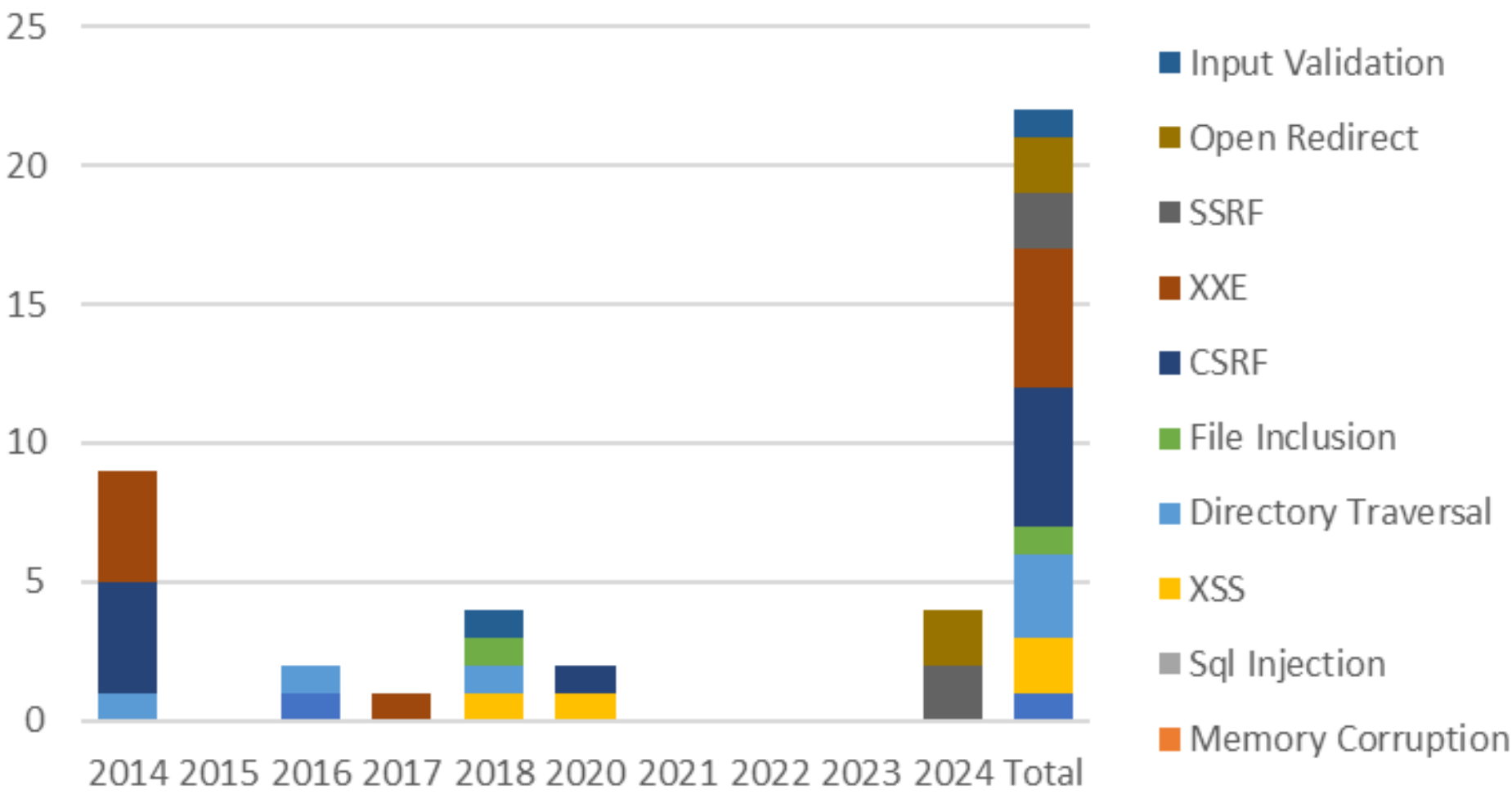

Figura 5. CVE Details de Spring Framework

Bases de Datos

Las bases de datos almacenan la información que debe ser conservada y no debe ser accesible para el cliente. Son compatibles con los requisitos de privacidad y conformidad asociados a cualquier dato. Por ejemplo, para acceder a la base de datos, los usuarios deben iniciar sesión. Los diferentes usuarios también pueden tener diferentes niveles de acceso, como el de solo lectura. Dichas bases se pueden clasificar de varias maneras, siendo las más populares: SQL y NoSQL (Lala, Kumar et al. 2021).

Resultados y discusión

En esta sección se detallan los resultados obtenidos a lo largo de la presente investigación.

Requisitos de seguridad

Siguiendo las especificaciones establecidas en la ASVS de forma que se pueda garantizar la disponibilidad, confidencialidad, integridad, no repudio y privacidad de la información y sistemas web, se destacan los siguientes requisitos a modo de lista de verificación para aplicaciones.

Ciclo de vida de desarrollo de software seguro

No	Descripción	CWE
1.1.2	Verifique el uso del modelado de amenazas para cada cambio de diseño.	1053
1.1.3	Verifique que todas las historias y características de usuario contienen restricciones de seguridad funcionales.	1110

Tabla 1. Requisitos de seguridad esenciales en el ciclo de desarrollo de software

Autenticación

No	Descripción	CWE
2.1.4	Verifique que cualquier carácter *Unicode* imprimible, incluidos los caracteres neutros del idioma, como espacios y emojis esté permitido en las contraseñas.	521
2.1.6	Verifique que la funcionalidad de cambio de contraseña requiere la contraseña actual y nueva del usuario.	620
2.1.9	Verifique que no hay reglas de composición de contraseñas que limiten el tipo de caracteres permitidos. No debe haber ningún requisito para mayúsculas o minúsculas o números o caracteres especiales.	521

2.2.1	Verifique que los controles anti-automatización son efectivos para mitigar las pruebas de credenciales filtradas, fuerza bruta y ataques de bloqueo de cuentas. Estos controles incluyen el bloqueo de las contraseñas filtradas más comunes, bloqueos suaves, limitación de velocidad, CAPTCHA, retrasos cada vez mayores entre intentos, restricciones de direcciones IP o restricciones basadas en riesgos.	307
2.2.2	Verifique que el uso de autenticadores débiles (como SMS y correo electrónico) se limita a la verificación secundaria y la aprobación de transacciones y no como un reemplazo para métodos de autenticación más seguros.	304
2.3.2	Verifique que se admite la inscripción y el uso de dispositivos de autenticación proporcionados por el suscriptor, como *tokens* U2F o FIDO.	308

Tabla 2. Requisitos de seguridad esenciales para la autenticación

Gestión de Sesiones

Las sesiones deben ser únicas para cada usuario y no pueden ser compartidas. Así como estas deben ser invalidadas cuando ya no sean necesarias y se agota el tiempo de espera. En la tabla 3 se muestran los requisitos de seguridad esenciales en la gestión de sesiones

No	Descripción	CWE
3.1.1	Verifique que la aplicación nunca revela *tokens* de sesión en parámetros de dirección URL.	598
3.2.3	Verifique que la aplicación solo almacena *tokens* de sesión en el navegador mediante métodos seguros, como proteger las *cookies* adecuadamente.	539
3.2.4	Verifique que los *tokens* de sesión se generan mediante algoritmos criptográficos aprobados.	331
3.3.1	Verifique que el cierre de sesión y la expiración invalidan el *token* de sesión, de modo que el botón “Atrás” o un usuario de confianza posterior no reanude una sesión autenticada, incluso entre los usuarios de confianza.	613
3.4.1	Verifique que los *tokens* de sesión basados en *cookies* tengan el atributo ‘*Secure*' establecido.	614
3.4.2	Verifique que los *tokens* de sesión basados en *cookies* tienen el atributo ‘HttpOnly’ establecido.	1004
3.4.3	Verifique que los *tokens* de sesión basados en *cookies* utilizan el atributo ‘*SameSite*' para limitar la exposición a ataques de falsificación de solicitudes entre sitios.	16

3.5.1	Verifique que la aplicación permite a los usuarios revocar *tokens* de OAuth que forman relaciones de confianza con aplicaciones vinculadas.	290
3.5.3	Verifique que los *tokens* de sesión sin estado utilizan firmas digitales, cifrado y otras contramedidas para protegerse contra ataques de manipulación, envolvente, reproducción, cifrado nulo y sustitución de claves.	345

Tabla 3. Requisitos de seguridad esenciales en la gestión de sesiones

Control de Acceso

No	Descripción	CWE
4.1.1	Verifique que la aplicación utiliza las reglas de control de acceso en una capa de servicio de confianza, especialmente si el control de acceso del lado cliente está presente y podría ser *bypaseado*.	602
4.1.2	Verifique que todos los atributos de usuario y datos y la información de directiva utilizada por los controles de acceso no pueden ser manipulados por los usuarios finales a menos que se autorice específicamente.	639
4.1.3	Verifique que existe el principio de privilegios mínimos: los usuarios solo deben poder acceder a funciones, archivos de datos, direcciones URL, controladores, servicios y otros recursos, para los que poseen una autorización específica.	285
4.3.2	Verifique que la exploración de directorios está deshabilitada a menos que se desee deliberadamente. Además, las aplicaciones no deben permitir la detección o divulgación de metadatos de archivos o directorios, como *Thumbs.db*, *.DS Store*, .git o .svn.	584

Tabla 4. Requisitos de seguridad esenciales en el control de acceso

Validación. Desinfección y Codificación

No	Descripción	CWE
5.1.1	Verifique que la aplicación tiene defensas contra los ataques de contaminación de parámetros HTTP, especialmente si el marco de la aplicación no hace ninguna distinción sobre el origen de los parámetros de solicitud.	235
5.1.3	Verifique que todas las entradas se validan mediante validación positiva.	20
5.1.4	Verifique que las estructuras de datos están fuertemente tipados y validados con un esquema definido que incluya caracteres permitidos, longitud y patrón.	20

5.2.3	Verifique que la aplicación filtra la entrada del usuario antes de pasar a los sistemas de correo para protegerse contra la inyección SMTP o IMAP.	147
5.2.4	Verifique que la aplicación evita el uso de eval() u otras características de ejecución de código dinámico. Cuando no hay alternativa, cualquier entrada de usuario debe filtrarse, y ponerlo en *sandbox* antes de ejecutarse.	95
5.2.6	Verifique que la aplicación protege contra ataques SSRF, validando o desinfectando datos que no son de confianza o metadatos de archivos HTTP, como nombres de archivo y campos de entrada de URL, y utiliza listas de protocolos permitidos, dominios, rutas de acceso y puertos.	918
5.2.7	Verifique que la aplicación desinfecta, deshabilita o pone en *sandbox* el contenido proporcionado por el usuario, con *scripts* de gráficos vectoriales escalables especialmente en lo que se refiere a XSS resultante de *scripts* en línea y *foreignObject*.	159
5.2.8	Verifique que la aplicación desinfecta, deshabilita o pone en *sandbox* el contenido proporcionado por el usuario, con expresiones en lenguaje de plantilla o *script* como *Markdown*, CSS o las hojas de estilo XSL, *BBCode* o similares.	94
5.3.3	Verifique que el escape de salida basado en contexto, preferiblemente automatizado —o en el peor de los casos, manual— protege contra XSS reflejado, almacenado y basado en DOM.	79
5.3.4	Verifique que la selección de datos o las consultas de base de datos (por ejemplo, SQL, HQL, ORM, NoSQL) utilizan consultas parametrizadas, ORM, marcos de entidades o están protegidas de los ataques de inyección de base de datos.	89
5.3.9	Verifique que la aplicación protege contra ataques de inclusión de archivos locales (LFI) o de inclusión remota de archivos (RFI).	829

Tabla 5. Requisitos de seguridad esenciales en la validación de datos

Manejo y Registro de Errores

No	Descripción	CWE
7.1.1	Verifique que la aplicación no registra las credenciales ni los detalles de pago. Los *tokens* de sesión solo deben almacenarse en registros de forma irreversible y *hasheados*.	532
7.1.2	Verifique que la aplicación no registra otros datos confidenciales tal como se definen en las leyes de privacidad locales o la política de seguridad pertinente.	532

7.1.3	Verifique que la aplicación registra eventos relevantes para la seguridad, incluidos los eventos de autenticación correctos y con errores, los errores de control de acceso, de deserialización y los errores de validación de entrada.	778
7.2.1	Verifique que se registran todas las decisiones de autenticación, sin almacenar *tokens* o contraseñas de sesión confidenciales. Esto debe incluir solicitudes con los metadatos relevantes necesarios para las investigaciones de seguridad.	778
7.3.3	Verifique que los registros de seguridad están protegidos contra el acceso y la modificación no autorizados.	200
7.3.4	Verifique que la fuente donde se lee el tiempo están sincronizados con la hora y la zona horaria correctas. Considere firmemente el registro solo en UTC si los sistemas son globales para ayudar con el análisis forense posterior al incidente.	

Tabla 6. Requisitos de seguridad esenciales para el manejo y registro de errores

Protección de Datos

No	Descripción	CWE
8.1.1	Verifique que la aplicación protege los datos confidenciales de la caché en componentes del servidor, como balanceadores de carga y cachés deaplicaciones.	524
8.1.2	Verifique que todas las copias almacenadas en caché o temporales de datos confidenciales almacenados en el servidor están protegidas contra el acceso no autorizado o purgadas/invalidadas después de que el usuario autorizado acceda a los datos confidenciales.	524
8.1.4	Verifique que la aplicación puede detectar y alertar sobre números anormales de solicitudes, como por IP, usuario, total por hora o día, o lo que tenga sentido para la aplicación.	770
8.1.5	Verifique que se realizan copias de seguridad periódicas de datos importantes y que se realizan pruebas de la restauración de datos.	19
8.2.1	Verifique que la aplicación establece suficientes encabezados anti-almacenamiento en caché para que los datos confidenciales no se almacenen en caché en los navegadores modernos.	525
8.2.2	Verifique que los datos almacenados en el almacenamiento del navegador (como *localStorage*, *sessionStorage*, *IndexedDB* o *cookies*) no contengan datos confidenciales.	922
8.2.3	Verifique que los datos autenticados se borran del almacenamiento del cliente, como el DOM del explorador, después de que se termine el cliente o la sesión.	922

Tabla 7. Requisitos de seguridad esenciales para la protección de los datos

Archivos y Recursos

No	Descripción	CWE
12.1.1	Verifique que la aplicación no aceptará archivos grandes que puedan llenar el almacenamiento o provocar una denegación de servicio.	400
12.1.3	Verifique que se aplica una cuota de tamaño de archivo y un número máximo de archivos por usuario para asegurarse de que un solo usuario no puede llenar el almacenamiento con demasiados archivos o archivos excesivamente grandes.	770
12.2.1	Verifique que los archivos obtenidos de orígenes que no son de confianza se validan para que sean del tipo esperado en función del contenido del archivo.	434
12.3.3	Verifique que los metadatos del nombre de archivo enviados por el usuario se validan u omiten para evitar la divulgación o ejecución de archivos remotos a través de ataques de inclusión remota de archivos (RFI) o falsificación de solicitudes del lado del servidor (SSRF).	98
12.4.2	Verifique que los archivos obtenidos de fuentes no confiables se almacenen fuera de la raíz web, con permisos limitados.	552
12.5.1	Verifique que la capa web está configurado para transmitir solo archivos con extensiones específicas, para evitar la filtración accidental de información o código fuente. Por ejemplo, los archivos de copia de seguridad (p. ej. .bak), los archivos de trabajo temporales (p. ej. .swp), los archivos comprimidos (.zip, .tar.gz, etc.) y otras extensiones utilizadas comúnmente por los editores deben bloquearse a menos que sea necesario.	552
12.5.2	Verifique que las solicitudes directas a los archivos cargados nunca se ejecutarán como contenido HTML/*JavaScript*.	434

Tabla 8. Requisitos de seguridad esenciales en el manejo de archivos y recursos

API y Servicios Web

No	Descripción	CWE
13.1.3	Verifique que las direcciones URL de la API no exponen información confidencial, como *API keys*, los *tokens* de sesión, etc.	598
13.1.5	Verifique que las solicitudes que contienen tipos de contenido inesperados o contenido que falta se rechazan con encabezados adecuados (estado de respuesta HTTP 406 Inaceptable o 415 Tipo de medio no compatible).	434

13.2.1	Verifique que los métodos HTTP RESTful habilitados son una opción válida para el usuario o la acción, como impedir que los usuarios normales usen *DELETE* o *PUT* en recursos o API protegidos.	650
13.2.5	Verifique que los servicios *REST* comprueben explícitamente que el tipo de contenido entrante sea el esperado, como application/xml o application/json.	436

Tabla 9. Requisitos de seguridad esenciales en apis y servicios web

Seguridad de *Frameworks*

La utilización de *frameworks* web modernos agiliza significativamente el proceso de desarrollo de aplicaciones web. Estas herramientas ofrecen a las organizaciones y desarrolladores capacidades robustas para la creación de sus sistemas. Sin embargo, a pesar de que estos *frameworks* incluyen mecanismos de seguridad, no son completamente a prueba de fallos (Kornienko, Mishina et al. 2021). Un análisis retrospectivo de las vulnerabilidades de algunos de los *frameworks* de desarrollo web más utilizados revela que aquellos con mayor tiempo en el mercado y complejidad tienden a presentar un mayor número de vulnerabilidades. Por ello, es esencial que se adopten prácticas de desarrollo sólidas y mecanismos de seguridad adicionales. Por ejemplo, *Django* ha registrado un total de 172 vulnerabilidades, en contraste con *FastAPI*, que cuenta con 2 CVEs documentados en *CVEdetails*. Esta diferencia subraya la importancia de una implementación consciente y meticulosa de las medidas de seguridad, más allá de las que proporciona el *framework* por defecto. La fundación OWASP brinda varias herramientas y guías de buenas prácticas a la hora de desarrollar aplicaciones web con estos *frameworks* como la OWASP *Cheat Sheets Serie* (Almutairi, Mishra et al. 2022). Así como OWASP *Java Encoder*, entre otros.

Conclusiones

La seguridad en el desarrollo de aplicaciones web no debe ser vista como una etapa aislada, sino como un proceso integrado y esencial desde el inicio del ciclo de desarrollo. Los *frameworks* modernos ofrecen herramientas de seguridad significativas, pero no son suficientes para garantizar aplicaciones seguras si no se acompañan de prácticas de desarrollo adecuadas y un diseño de software bien fundamentado. La presente investigación ha

demostrado que las vulnerabilidades pueden surgir no solo a nivel de *framework*, sino también a través de la implementación inadecuada de políticas de acceso, validación de datos deficiente y mecanismos de autenticación insuficientes. La adopción de las guías de la *Open Web Security Project* (OWASP) y los estándares del (OWASP) *Application Security Verification Standard* (ASVS) es crucial para mitigar estas vulnerabilidades.

Además, ha demostrado que las organizaciones deben adoptar un enfoque proactivo en la inteligencia de vulnerabilidades, de manera tal que permita tener un conocimiento de la superficie de ataque. Esto facilita anticiparse a futuras amenazas. Al igual que brinda la posibilidad de desarrollar sistemas y *frameworks* para automatizar o tener un control de las mismas para así desarrollar futuras estrategias de desarrollo y protección.

En resumen, la seguridad debe ser una preocupación constante y un objetivo compartido entre desarrolladores y organizaciones, donde la educación continúa, la adopción de buenas prácticas y la colaboración con la comunidad de seguridad son claves para el desarrollo de aplicaciones web robustas y seguras. La presente investigación subraya la importancia de integrar la seguridad en todas las fases del desarrollo de software, promoviendo una cultura de seguridad que proteja tanto a los usuarios como a las organizaciones de los riesgos emergentes en el ciberespacio.

Referencias bibliográficas

Almutairi, A. A., et al. (2022). “Web Security: Emerging Threats and Defense.” Computer Systems Science & Engineering 40(3).

Cascavilla, G., et al. (2021). “Cybercrime threat intelligence: A systematic multi-vocal literature review.” Computers & Security 105: 102258.

Helmiawan, M. A., et al. (2020). *Analysis of web security using open web application security project 10. 2020 8th International Conference on Cyber and IT Service Management* (CITSM), IEEE.

Idris, M., et al. (2022). “Web application security education platform based on OWASP API security project.” EMITTER international journal of engineering technology: 246-261.

Kornienko, D., et al. (2021). “Principles of securing RESTful API web services developed with python frameworks”. *Journal of Physics: Conference Series*, IOP Publishing.

Lala, S. K., et al. (2021). *Secure web development using owasp guidelines. 2021 5th International Conference on Intelligent Computing and Control Systems* (ICICCS), IEEE.

Le, B. D., et al. (2019). “Gathering cyber threat intelligence from Twitter using novelty classification.” arXiv preprint arXiv:1907.01755.

Li, J. (2020). “Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST).” arXiv preprint arXiv:2004.03216.

Sołtysik-Piorunkiewicz, A. and M. Krysiak (2020). “The cyber threats analysis for web applications security in industry 4.0.” *Towards Industry 4.0*—Current Challenges in Information Systems: 127-141.

Subedi, B., et al. (2016). Subedi, B. et al. (2016). "Secure paradigm for web application development". 2016. 15th RoEduNet Conference: Networking in Education and Reasearch, IEEE.

Wen, S.-F. and B. Katt (2023). “A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard.” *Computers & Security* 135: 103532.