Lecciones aprendidas y recomendaciones de seguridad en el desarrollo de aplicaciones

***Abstract**: la exposición a ciberamenazas en la que se encuentran las diferentes organizaciones subraya la importancia del desarrollo seguro de las aplicaciones con el fin de reducir el riesgo derivado de las brechas de seguridad. Con la metodología del desarrollo seguro se hace posible analizar y evaluar las amenazas, identificar los puntos débiles, implementar soluciones seguras, acelerar el desarrollo, evitar vulnerabilidades típicas, crear procesos de actuación ante vulnerabilidades e incidentes y, en definitiva, mejorar la postura de seguridad de la organización.*

Contenido:

1. INTRODUCCIÓN

La **seguridad** es una **necesidad básica** en el ser humano y ocupa el segundo nivel de la pirámide de Maslow. Es inherente a cualquier actividad y nunca podrá ser alcanzada en su totalidad, aunque es posible reducirlo hasta unos niveles aceptables de riesgo (riesgo residual).

El siglo XXI se caracteriza por el avance y expansión de la digitalización, así como del control de la información a nivel global. El concepto de **ciberseguridad** nace por la **necesidad** de las compañías **de proteger** sus **sistemas contra ataques informáticos** que pudiesen poner en riesgo su correcto funcionamiento o los datos almacenados en ellos.

La **ciberseguridad es la práctica dirigida a proteger sistemas**, **redes** y **datos** contra el acceso no autorizado, el uso indebido, la interrupción y/o disrupción de servicios y otros tipos de amenazas que causan los ciberataques. Su **objetivo** no solamente es aplicar diferentes sistemas de seguridad con el fin de **prevenir** y **contrarrestar** dichos ataques, sino que también consiste en **educar** y **capacitar** a los usuarios sobre cómo evitar riesgos innecesarios.

En sus inicios, las aplicaciones se desarrollaban con una total ausencia de seguridad:

- Sin requisitos ni controles de seguridad.
- Desarrolladores con insuficientes o nulos conocimientos de desarrollo seguro.
- Sin verificaciones de los parámetros de entrada a nivel de seguridad.
- Sin herramientas automáticas para cubrir defectos del código.
- Sin realizar pruebas de seguridad manuales.
- Sin tener en cuenta el flujo correcto de la lógica de navegación.

- Sin verificar las vulnerabilidades de las librerías dependientes.
- Incorrecto tratamiento del control de versiones en producción.
- Sin planificar una correcta gestión de incidentes de seguridad.
- **En definitiva, problemas de seguridad resueltos siempre de forma reactiva**.

Con la aparición de las ciberamenazas, causantes de pérdidas económicas y de prestigio en las organizaciones, **surge la necesidad** de incrementar los esfuerzos en la seguridad de los sistemas y la protección de los datos. Aparece el concepto de **seguridad en las aplicaciones**, para tratar activamente todas las potenciales amenazas y vulnerabilidades **desde su diseño hasta su implantación**, incluyendo controles de seguridad tanto a nivel de aplicación como en los sistemas encargados de dar soporte a estas aplicaciones o que dependan de ella.

La **seguridad en las aplicaciones comprende hardware, software y procesos** que identifican o minimizan las vulnerabilidades de seguridad.

2. VULNERABILIDADES Y AMENAZAS

La **falta o deficiencia de seguridad en las aplicaciones informáticas se denomina vulnerabilidad**. Las vulnerabilidades ponen en riesgo el servicio y la seguridad de la información, permitiendo que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad de los datos, por lo que se hace necesario que sean identificadas, eliminadas o mitigadas lo antes posible.

Las vulnerabilidades, cuando son aprovechadas para atentar contra la seguridad de un sistema de información, se conocen con el nombre de **amenazas**. El estudio de las amenazas ha permitido identificar y clasificar muchas de ellas para poder establecer controles de prevención. Algunas de las más conocidas son: denegación de servicio (DoS/DDoS), inyección SQL, suplantación de identidad, *Cross-site scripting*, etc.

Para ayudar a las organizaciones a construir un esquema de defensa contra las amenazas surge la necesidad de definir un **modelado de amenazas** que sirva para identificar, comunicar y comprender estas amenazas y sus mitigaciones. El modelado de amenazas es una representación estructurada de toda la información que afecta a la seguridad de una aplicación con el fin de mejorar la toma decisiones sobre los riesgos identificados.

Cuando coexisten vulnerabilidades y amenazas aumenta la probabilidad de que se produzca un **incidente de seguridad**, que es la manifestación de la amenaza y la posible causa directa de pérdidas o daños. A esta probabilidad se le denomina **riesgo** y se mide asumiendo ciertas vulnerabilidades frente a determinadas amenazas.

Existe un **sistema de categorías** para las **debilidades del software** denominado **CWE** [1]. Se sustenta en un proyecto comunitario con el objetivo de comprender las fallas en el software y crear herramientas automatizadas que se puedan utilizar para identificar, corregir y prevenir estas fallas.

Así mismo, existe una lista pública de **vulnerabilidades de seguridad** denominada **CVE** [2] . Están registradas mediante un identificador (CVE-ID), una descripción, las versiones del software afectadas, su posible solución o mitigación y las referencias a publicaciones, foros o blogs donde se hizo pública la vulnerabilidad y donde se demostró su explotación.

Para detectar, eliminar y/o mitigar las debilidades de una aplicación se pueden realizar análisis de seguridad en diferentes fases del ciclo de vida del desarrollo de software (SDLC):

- **SAST (*Static Application Security Testing*)**: análisis estático del código fuente. Puede ser automático o manual, y se realiza en las fases de desarrollo, despliegue y *testing*. Cuando se realiza en fase de *testing* de forma manual se denomina pruebas de **caja blanca**.
- **SCA (*Software Composition Analysis*)**: análisis de los componentes y librerías de terceros de los que depende la aplicación. Suele ser automático y se realiza en las fases de desarrollo, despliegue y *testing*.
- **DAST (*Dynamic Application Security Testing*)**: análisis dinámico de la aplicación en un entorno pre-productivo mediante la realización de pruebas de abuso y penetración sobre la aplicación con el fin de encontrar vulnerabilidades reales. Suele ser automático o manual, y se realiza en las fases de despliegue y *testing*. Cuando se realiza en fase de *testing* de forma manual se denomina pruebas de **caja negra,** siempre que se realicen sin conocimiento del código de la aplicación; si se conociese se denominaría de **caja gris**.

3. DESARROLLO SEGURO EN LAS APLICACIONES

La **metodología de desarrollo seguro de aplicaciones** es un conjunto de procesos y procedimientos diseñados para facilitar la identificación de los puntos débiles de seguridad en algunas o en todas las etapas del SDLC, mediante la aplicación de controles de seguridad que sirvan para tomar las acciones necesarias con el fin de asegurar el software lo máximo posible, desde su diseño hasta su puesta en producción y durante todo el tiempo de servicio.

S-SDLC (*Secure Software Development Life Cycle*) es una metodología para el desarrollo de software que incorpora las mejores prácticas de seguridad durante todo el ciclo de vida de la aplicación con el fin de que sea segura desde su inicio hasta el final de su servicio.

Existen muchos modelos de definición de fases para estos procesos, pero las comúnmente identificadas son:

- **Diseño**: identificar los requisitos de seguridad, establecer políticas y estándares de seguridad para la aplicación, y diseñar la aplicación con perspectiva de seguridad aplicando todas las recomendaciones del diseño seguro.
- **Implementación**: codificar la aplicación teniendo en cuenta los requisitos y estándares de seguridad, así como todas las recomendaciones relativas al desarrollo seguro.
- **Despliegue**: despliegue de la aplicación en un primer entorno de ejecución similar a producción donde verificar automáticamente pruebas funcionales, requisitos de seguridad y otros controles.
- **Testing**: realización de las pruebas funcionales y de seguridad para detectar errores y vulnerabilidades. Normalmente suelen ser pruebas de verificación manuales en un entorno semejante al de producción.
- **Operaciones**: monitorizar, detectar amenazas y actualizar continuamente la seguridad de la aplicación. En esta fase también se tienen en cuenta los sistemas que soportan la aplicación, así como su configuración de seguridad/bastionado.

Los **estándares de seguridad del software seguro** son guías desarrolladas por gobiernos, instituciones u organizaciones que permiten medir y evaluar el grado de cumplimiento de un sistema respecto los requisitos de la propia guía con el fin de garantizar la seguridad del software en un contexto determinado. Algunos estándares de seguridad los tenemos en ENS, PCI-DSS o NIST.

Los **modelos de madurez de desarrollo del software seguro** proporcionan un marco de organizado de requisitos de seguridad cuyo nivel de cumplimiento permite evaluar la posición de madurez de la implementación de la seguridad en: una aplicación, un proyecto o una organización. Los modelos de madurez más utilizados son: OWASP SAMM, ISO 33000 (parte 2 y 7)

Las organizaciones que aplican metodologías de desarrollo seguro ven reducidos de manera significativa sus costes de gestión de la configuración y de respuesta a incidentes en un 75% (**Figura 1**).

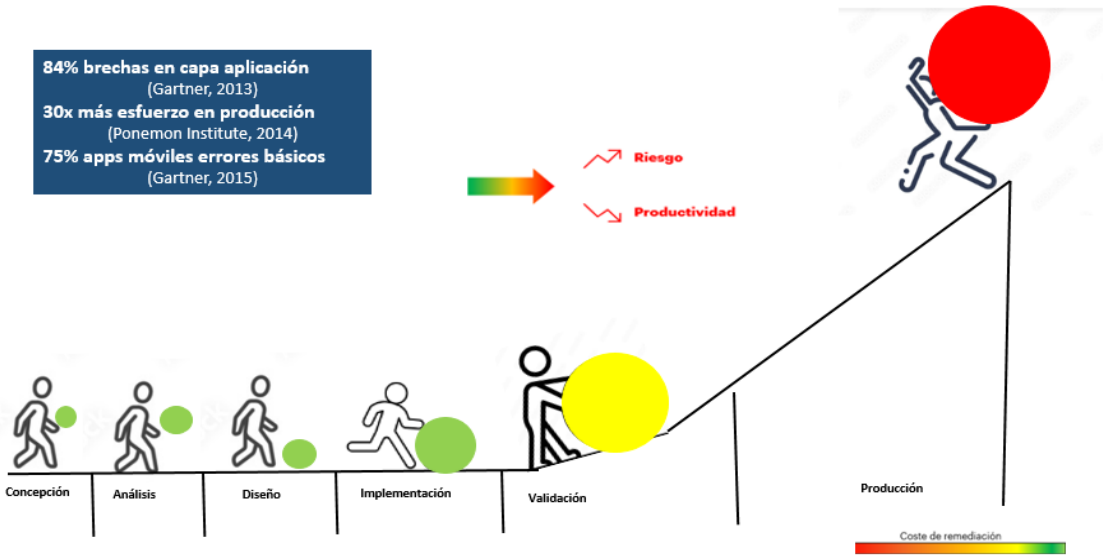

Figura 1.- Ventajas en la aplicación de metodologías de desarrollo seguro.

Desde este punto de vista, si la **incorporación de la seguridad sólo se realiza en las fases avanzadas del ciclo de vida** del desarrollo de las aplicaciones, sigue **existiendo un elevado riesgo, un mayor coste de remediación y una disminución de la productividad**, es decir, a medida que avanzamos en el ciclo de vida del desarrollo, la introducción de controles de seguridad y remediación de errores se vuelven cada vez más costosos [3] [4] [5]. Esto supone una dedicación de esfuerzos cada vez mayores por parte del equipo de desarrollo, lo que degrada su productividad e implica la asunción de riesgos mayores por parte de la dirección del proyecto.

Actualmente existe un sólido respaldo en cuanto a **entidades y organizaciones responsables del desarrollo de los estándares y buenas prácticas de seguridad**, como pueden ser OWASP, NIST, MITRE, ISO, etc.

Las **consecuencias que conlleva no aplicar suficientes controles de seguridad en las aplicaciones** son graves, múltiples y de distinta naturaleza, y casi siempre implican pérdidas económicas y/o daños que afectan muy negativamente a la imagen de la organización.

Algunas de las más comunes son:

- Pérdida o robo de información sensible o personal, como números de tarjetas de crédito, contraseñas, información de identificación personal, etc. que conllevan juicios y multas en daños y reparaciones, así como una importante pérdida de imagen comercial.
- Interrupción de los servicios mediante ataques DDoS causando pérdida de ingresos, clientes insatisfechos y daños en la reputación.
- Violación de cumplimiento normativo que es causa de sanciones y multas.
- Dificultad para obtener seguros de responsabilidad civil al considerarse que la ausencia de medidas de seguridad adecuadas aumenta el riesgo de incidentes.
- Pérdida de competitividad en el mercado. Los clientes y proveedores buscan trabajar con organizaciones que dispongan de una buena postura de seguridad.
- Daño en los sistemas y servicios que requirieran altos costes en reparaciones y recuperaciones de la integridad.
- Riesgo de extorsión. Puede requerir altos pagos de rescate para recuperar los datos o para evitar la divulgación de información sensible.

Sólo el coste que supuso corregir los defectos de seguridad encontrados en producción en lugar de haber sido corregidos durante el ciclo de desarrollo fue un importe de $1.04T durante el periodo 2019-2023. [6]

4. PRINCIPALES RETOS Y OPORTUNIDADES

La seguridad en las aplicaciones conlleva una **serie de retos** como son:

- Disponer de herramientas para modelados de la amenaza.
- Crear catálogos de requisitos de seguridad.
- Obtener guías de desarrollo seguro.
- Crear los procesos para la gestión de vulnerabilidades.
- Desarrollo de técnicas y habilidades en el uso de herramientas para el escaneo y testeo automático.
- Realizar pruebas de seguridad manuales que suponen un alto coste.
- Planes de formación en desarrollo seguro individualizados con diseños curriculares.

Sin embargo, al completar estos retos, se obtendrían las **siguientes oportunidades**:

- Procesos y procedimientos mejor definidos.
- Aplicaciones de mayor valor.
- Mayor velocidad de desarrollo.

- Mejores herramientas de análisis y control.
- Menos errores y menos coste en correcciones.
- Pruebas antes disponibles, predefinidas y menos costosas.
- Automatización en la identificación de las vulnerabilidades.
- Mayor conocimiento por parte de los desarrolladores en desarrollo seguro.
- Obtención de métricas para mejora de los procesos.
- Integración de elementos de mayor valor como la automatización y la ayuda de procesos basados en inteligencia artificial.

A pesar del incremento de inversión en gasto que suponen todas estas nuevas iniciativas en seguridad, las brechas de datos en las aplicaciones continúan en aumento (**Figura 2**). Aunque se ha mejorado mucho en los procesos y la seguridad del código, las herramientas de escaneo, detección y corrección aún son lentas e insuficientes.

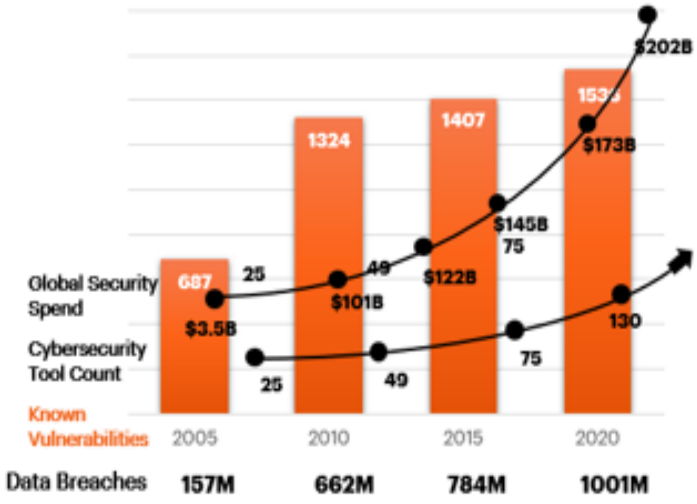

Figura 2.- Brechas de datos en aplicaciones.

5. GLOSARIO

AI: *Artificial Intelligence*

CVE: *Common Vulnerabilities and Exposures*

CWE: *Common Weakness Enumeration*

DAST: *Dynamic Application Security Testing*

ENS: Esquema Nacional de Seguridad

ISO: *International Organization for Standardization*

MITRE: *Massachusetts Institute of Technology Research Establishment*

NIST: *National Institute of Standards and Technology*

OWASP: *Open Web Application Security Project*

PCI-DSS: *Payment Card Industry Data Security Standard*

SAST: *Static Application Security Testing*

SCA: *Software Composition Analysis*

S-SDLC: *Secure Systems Development Life Cycle*

6. REFERENCIAS

[1] CWE - Common Weakness Enumeration, [Online]. Available: https://cwe.mitre.org.

[2] CVE - Common Vulnerabilities and Exposures," [Online]. Available: https://cve.mitre.org.

[3] Gartner, 2013, [Online]. Available: http://sast.se/q-moten/2015/q20/4_SAST_Q20_Jesper_Krakhede_AST.pdf.

[4] Ponemon institute, 2014," [Online]. Available: https://eu.usatoday.com/story/tech/2014/09/24/data-breach-companies-60/16106197/.

[5] Gartner, 2015," [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2014-09-14-gartner-says-more-than-75-percent-of-mobile-applications-will-fail-basic-security-tests-through-2015.

[6] State of Cybersecurity Report 2021, [Online]. Available: https://www.accenture.com/us-en/insights/security/cost-cybercrime-study.

[7] OWASP, OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation, [Online]. Available: https://owasp.org/

[8] NIST, National Institute of Standards and Technology, [Online]. Available: https://www.nist.gov/

[9] MITRE - Solving Problems for Safer World, [Online]. Available: https://www.mitre.org/

[10] ISO, International Organization for Standardization, [Online]. Available: https://www.iso.org