📘 Glosario: Desarrollo Seguro y Automatización

🔐 1. Desarrollo Seguro de Software

Conjunto de prácticas que integran la **seguridad como requisito fundamental** desde las etapas más tempranas del ciclo de desarrollo de software, con el fin de minimizar riesgos y vulnerabilidades.

🧱 2. SDLC (Software Development Life Cycle)

Modelo estructurado que define las fases del desarrollo de software: **planificación, diseño, codificación, pruebas, implementación y mantenimiento**. La seguridad puede y debe integrarse en cada etapa.

🛠️ 3. DevOps

Metodología que combina **desarrollo (Dev)** y **operaciones (Ops)** para lograr entregas rápidas, colaborativas y continuas. Automatiza despliegues y pruebas.

🔐 4. DevSecOps

Evolución de DevOps que **integra la seguridad como una responsabilidad compartida**, automatizando su implementación desde el primer commit hasta el despliegue.

🧪 5. SAST (Static Application Security Testing)

Análisis de seguridad que revisa el **código fuente sin ejecutarlo**, identificando vulnerabilidades como inyecciones, XSS o malas prácticas. Ejemplo: SonarQube.

🔍 6. DAST (Dynamic Application Security Testing)

Técnica de análisis de seguridad que evalúa la aplicación **durante su ejecución**, simulando ataques desde el exterior. Ejemplos: OWASP ZAP, Burp Suite.

📦 7. Análisis de Dependencias (SCA – Software Composition Analysis)

Proceso que identifica **librerías externas vulnerables o desactualizadas** utilizadas en un proyecto. Herramientas: Snyk, OWASP Dependency-Check.

🧰 8. SonarQube

Herramienta de análisis estático que evalúa la calidad y seguridad del código fuente, detectando errores, vulnerabilidades y problemas de mantenibilidad.

🔒 9. Snyk

Herramienta de seguridad que analiza automáticamente las **dependencias de proyectos** y alerta sobre vulnerabilidades conocidas, proponiendo soluciones.

⚙️ 10. GitHub Actions

Plataforma de automatización dentro de GitHub que permite ejecutar **workflows CI/CD** para pruebas, análisis de código y despliegues, integrando seguridad como parte del pipeline.

📋 11. Pipeline CI/CD (Integración y Entrega Continua)

Conjunto de procesos automatizados que permiten compilar, probar y desplegar software de forma continua, asegurando **entregas rápidas y seguras**.

🔄 12. Security by Design

Principio según el cual la seguridad debe ser considerada **desde la fase de diseño del software**, y no añadirse al final del desarrollo.

🛡️ 13. Defensa en Profundidad

Estrategia que aplica **múltiples capas de protección**, de modo que si una capa falla, otras sigan protegiendo el sistema (ej. validación, autenticación, cifrado).

⚠️ 14. Vulnerabilidad

Debilidad en un sistema, proceso o código que puede ser explotada para comprometer la seguridad (confidencialidad, integridad o disponibilidad).

⏱️ 15. MTTR (Mean Time to Remediate)

Tiempo promedio que toma **detectar y corregir una vulnerabilidad** desde el momento en que se identifica. Cuanto menor sea el MTTR, mayor la madurez del equipo.

🔎 16. STRIDE

Modelo para identificar amenazas en la fase de diseño:
Spoofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege.

📊 17. Reporte de Seguridad Automatizado

Documento o panel generado por herramientas como SonarQube o Snyk que resume hallazgos de seguridad y propone acciones para mitigarlos.

🧠 18. Conciencia de Seguridad

Nivel de conocimiento y responsabilidad que los equipos de desarrollo y operaciones tienen respecto a las **amenazas y mejores prácticas de seguridad**.
