Instrumento	*Práctica de ejercicios*

Alumno: Alan David Garcia Zamorano	**Fecha: 26/05/2023**
Carrera: Ingenieria en Desarrollo y Gestion de Software	**Grupo: IDGS91D**
Asignatura: Seguridad en el Desarrollo de Aplicaciones	**Unidad temática: Unidad 2. Aplicaciones seguras**
Profesor: Geovanni Machuca Pereida	

I.- Ejercicios a resolver:

En esta actividad, el alumno realizará una investigación sobre el Ciclo de vida de Desarrollo del Software Seguro (SSDLC).

II.-Procedimientos y resultados:

Seguridad en el ciclo de vida del desarrollo de software

Según (Barrio Puyo José Antonio, 2022) las organizaciones están tomando conciencia de la importancia de asumir la seguridad como un elemento esencial en los procesos del negocio debido a diversos tales como el incremento de servicios expuestos en internet, utilización de plataformas en la nube, globalización y movilidad, mientras que otros están asociados al cumplimiento normativo. Ambos pueden tener impacto económico para la organización si no se adopta una adecuada estrategia de seguridad de la información.

Modelo de desarrollo seguro de aplicaciones

- *Ciclo de Vida de Desarrollo Seguro de Software (S-SDLC):* se trata de un conjunto de principios y buenas prácticas tendentes a detectar, prevenir y corregir defectos de seguridad en el desarrollo de aplicaciones, de forma que se obtenga software de confianza y robusto frente a ataques maliciosos, que realice las funciones para las que fue diseñado, que esté libre de vulnerabilidades y asegure cumplir con los tres pilares sobre los que se asienta la seguridad de la información: integridad, disponibilidad y confidencialidad.
- *Modelo de Madurez para el Aseguramiento de Software (SAMM):* se trata de un marco de trabajo abierto y flexible que facilita la implementación de una estrategia de seguridad para el desarrollo de software diseñado por OWASP.

Estos modelos tienen su traslación en la metodología de trabajo DevSecOps que surge para integrar la seguridad en el proceso de desarrollo de software.

Herramientas para testear las aplicaciones

- *Static Application Security Testing (SAST):* estas permiten revisar el código fuente durante la fase de desarrollo de la aplicación y prevenir problemas de seguridad futuros.
- *Dynamic Application Security Testing (DAST):* este tipo de herramientas permiten realizar el escaneo tras desplegar la aplicación, no necesitando tener acceso al código fuente. Se caracterizan por tener un menor número de falsos positivos, si bien su eficacia es menor según aumenta la complejidad de la arquitectura o las funcionalidades de la aplicación. La velocidad de escaneo es menor respecto a las herramientas SAST.

Seguridad en el ciclo de vida de desarrollo de software

Según la página web (Red Hat, 2022) el ciclo de vida de desarrollo del software (SDLC) es un marco que se enfoca en el proceso para mejorar la calidad del software y

formaliza las tareas o actividades en etapas para que sea posible medir y analizar el sistema, incorporar mejoras y a su vez supervisar el progreso y los costos.

Etapas del SDLC:

- Planificación: determinar el alcance y la finalidad del software.
- Análisis de los requisitos: definir las funciones que debe ejecutar el software.
- Diseño: decidir los parámetros clave, como la arquitectura, las plataformas y las interfaces de usuario.
- Desarrollo: crear e implementar el software.
- Documentación: producir la información para que los usuarios y las partes interesadas puedan utilizar el sistema.
- Pruebas: verificar que el software cumpla con los requisitos.
- Implementación: poner el software a disposición de los usuarios.
- Mantenimiento: solucionar los errores y los puntos vulnerables que se descubran en el sistema.

Relación del SDLC con el enfoque ágil y DevOps

En la misma pagina web (Red Hat, 2022) habla sobre la ejecución de las ocho etapas del SDLC en orden secuencial, parecen describir el proceso en cascada, pero es importante recordar que además del proceso en cascada, el enfoque ágil, el eficaz, el repetible, el de espiral y el de DevOps también son metodologías de SDLC. Estas pueden variar según cuáles etapas incluyen, el orden de ejecución o el nombre que se le da a cada una de ellas.

Las metodologías de SDLC, como la ágil y la de DevOps, se centran en el enfoque repetitivo del desarrollo de software en lugar de la estrategia lineal en cascada.

Importancia de la seguridad en el SDLC

En el desarrollo de software es común que la seguridad se aborda en una etapa demasiado avanzada del proceso: la de pruebas, después de haber completado las tareas más importantes de diseño e implementación. En muchos casos, los controles de

seguridad que se ejecutan en esa etapa son superficiales, es decir, se limitan al análisis y las pruebas de intrusión. Es posible que se pasen por alto problemas de seguridad más complejos que de detectarse podrían retrasar la llegada del sistema a la producción. La resolución de los problemas lleva mucho tiempo y es más costosa, ya que puede requerir que se vuelva a desarrollar y probar todo el software.

Ciclo de vida de desarrollo de software seguro (SSDLC)

La implementación de procesos de seguridad eficaces requiere que los equipos tomen las medidas de protección desde las primeras etapas del desarrollo del software y en cada una de ellas.

Planificación

- Evaluar los riesgos y el panorama de las amenazas a la seguridad.
- Evaluar el posible impacto de los incidentes de seguridad, como el riesgo para la reputación de la empresa.

Análisis de los requisitos

- Incluir los requisitos de seguridad.
- Comprender e incorporar los requisitos de cumplimiento normativo.

Diseño

- Elaborar modelos de amenazas.
- Incluir las consideraciones de seguridad como parte integral del plan de arquitectura.
- Evaluar el impacto que tiene la seguridad en las decisiones de la etapa de diseño, como la plataforma y la interfaz de usuario.

Desarrollo

- Capacitar a los desarrolladores sobre las prácticas de codificación seguras.
- Incorporar herramientas de prueba de la seguridad en el proceso de desarrollo.

- Evaluar las dependencias del software y reducir los riesgos de seguridad.

Documentación

- Documentar los procesos y los controles de seguridad.
- Reunir la información para preparase para las auditorías, los controles de cumplimiento normativo y las revisiones de seguridad.

Pruebas

- Implementar procesos de revisión del código.
- Realizar pruebas estáticas o dinámicas de la seguridad de las aplicaciones.

Implementación

- Evaluar la seguridad del entorno de implementación.
- Revisar las configuraciones de seguridad.

Mantenimiento

- Supervisar el sistema para detectar amenazas.
- Prepararse para eliminar los puntos vulnerables y solucionar las intrusiones.

Implementación del SSDLC con DevSecOps y la automatización

En el SSDLC, los puntos y los controles de seguridad deben implementarse desde el principio del proceso de desarrollo e implementación. Las empresas adoptan el enfoque de DevOps y los canales automatizados de integración e implementación continuas (CI/CD) para poder implementar mejoras de manera permanente y rápida. Para evitar los bloqueos, la seguridad debe ser un proceso constante y automatizado, y los equipos de desarrollo deben encargarse de la protección de las aplicaciones, además del diseño, el desarrollo, las operaciones y el mantenimiento.

DevSecOps es un conjunto de practicas que incluye personas, procesos y tecnologías para mejorar la velocidad y la eficiencia del desarrollo del software,

mientras que ofrecen mayor seguridad, uniformidad, capacidad de repetición y colaboración. La clave está en compartir la responsabilidad entre los equipos de desarrollo, de operaciones y de seguridad. Estos son algunos de los objetivos de DevSecOps:

- Mejorar la seguridad y disminuir los riesgos gracias a la eliminación de más puntos vulnerables desde el inicio del ciclo de vida de la infraestructura y el desarrollo de las aplicaciones.
- Aumentar la eficiencia y la velocidad de los ciclos de lanzamiento de DevOps con la eliminación de las prácticas y las herramientas de seguridad heredadas. El uso de la automatización; la adopción de una cadena de herramientas de manera estandarizada; y la implementación de la infraestructura, la seguridad y el cumplimiento normativo como código para mejorar la capacidad de repetición y la uniformidad pueden optimizar el proceso de desarrollo.
- Disminuir los riesgos y aportar claridad mediante la implementación de controles de seguridad desde el comienzo del ciclo de vida de la infraestructura y el desarrollo de las aplicaciones, lo cual reduce la probabilidad de que se cometan errores humanos y mejora la seguridad, el cumplimiento normativo y la capacidad para anticipar los inconvenientes y repetir los procesos que permitan solucionarlos, y disminuye los problemas de auditoría.

Estas cuatro etapas permiten garantizar que se incorpore la seguridad al canal de CI/CD y que se adapte a medida que cambien las condiciones en la empresa o en el mundo.

Funcionamiento del marco de desarrollo seguro de software

Según la pagina web (Tarlogic, 2023) el National Institute of Standards and Tecnology de Estados Unidos (NIST) dice que muchos desarrolladores se centran en la calidad del software que diseñan y producen en su usabilidad p en su complejidad técnica. Sin darle a la seguridad lo posición central que debería tener a la hora de desarrollar software.

Abriendo puertas a los ciberataques y permitiendo la propagación de ataques de cadena de suministro que no solo buscan dañar a los productores de software sino también a las empresas que lo usan.

El NIST propone con su marco de desarrollo seguro de software implementar una serie de buenas practicas a lo largo de todas las fases del ciclo de vida del software.

Frente a este peligroso panorama, el NIST propone, con su Marco de desarrollo seguro de software, implementar una serie de buenas prácticas a lo largo de todas las fases del ciclo de vida del software.

- Reducir el número de vulnerabilidades en el software presente en el mercado.
- Mitigar el impacto que podría tener la explotación de vulnerabilidades no subsanadas.
- Analizar las causas raíz de las vulnerabilidades para prevenir futuras debilidades.
- Facilitar la comunicación entre proveedores y consumidores de software y ayudar a las empresas a establecer sistemas de control del software de terceros que emplean.

Buenas practicas

El marco de desarrollo seguro de software propone cuatro recomendaciones que articulan, a su vez, el conjunto de buenas prácticas que conforman este framework:

- Las compañías deben asegurarse de que sus profesionales, procesos y herramientas están preparados para llevar a cabo un desarrollo de software seguro durante todo el ciclo de vida de las soluciones que producen.
- Las organizaciones deben proteger todos los componentes empleados en su software, atendiendo especialmente a la manipulación de los mismos, así como ante accesos no autorizados.
- Las empresas tienen que producir software seguro no solo en las primeras fases del ciclo de vida, sino también en todas las versiones de sus soluciones, reduciendo las vulnerabilidades al mínimo.

- Las compañías han de ser capaces de identificar vulnerabilidades residuales en las versiones del software y contar con mecanismos eficaces para subsanarlas.

Preparar a la organización

La preparación de una empresa para facilitar que el software que desarrolla es seguro, requiere implementar cinco prácticas:

- Definir los requisitos de seguridad para el desarrollo de software. Para ello es necesario recopilar tanto fuentes internas, por ejemplo, la estrategia de gestión de riesgos de seguridad, como externas, tal como la normativa en vigor que se debe cumplir. Asimismo, es necesario comunicar estos requisitos a los proveedores de componentes de software que van a ser utilizados en el software propio de la compañía.
- Establecer funciones y responsabilidades. Es importante que tanto los profesionales internos, como los externos, tengan claro cuál es su papel, reciban la formación adecuada y asuman sus responsabilidades a lo largo de todo el ciclo de vida del software.
- Implementar cadenas de herramientas de apoyo. La automatización juega un papel esencial en la protección del software en todo su siclo de vida. Ayudando a los profesionales a detectar vulnerabilidades, optimizar las prácticas y documentarlas.
- Definir y usar criterios para verificar la seguridad del software durante todas sus fases de desarrollo.
- Poner en marcha y mantener entornos seguros para el desarrollo de software. Es fundamental asegurarse de que todos los componentes de los entornos están protegidos, tanto frente ataques externos como internos.

Proteger el software

Es crucial proteger el código del software y contar con protocolos y metodologías para verificar la seguridad de cada versión del software:

- Proteger el código frente al acceso no autorizado y evitar su manipulación. Los cambios no autorizados del código pueden anular las características de seguridad implementadas o introducir nuevas vulnerabilidades y, además, el acceso indebido puede facilitar la búsqueda de vulnerabilidades por parte de los delincuentes e, incluso, conllevar el robo del software o el código fuente y atentar contra la propiedad intelectual.
- Establecer un mecanismo para verificar la integridad de cada versión del software. No solo es importante que la organización verifique internamente la seguridad de cada versión que lanza, sino que los compradores también deben contar con los mecanismos necesarios para asegurarse de que el software que adquieren es legítimo y no ha sufrido ningún tipo de manipulación.
- Recopilar y archivar cada versión de software para detectar y subsanar vulnerabilidades. Si se documenta y conserva cada versión lanzada, se facilita la posibilidad de analizar, mitigar y eliminar vulnerabilidades descubiertas después de que la versión haya sido publicada.

Producir software bien protegido

Este grupo contiene el mayor número de prácticas a implementar:

- Diseñar el software para cumplir con los requisitos de seguridad y mitigar los riesgos. Tener en cuenta los requisitos y riesgos desde la fase de requisitos y diseño del software es clave para mejorar el desarrollo.
- Revisar el diseño para verificar que se cumplen con los requisitos y se informa sobre los riesgos de forma debida.
- Reutilizar software previo y que esté bien protegido, cuando sea factible, en vez de reimplementar funcionalidades. Esto no solo reduce los costes y los tiempos de desarrollo, sino que disminuye la posibilidad de introducir nuevas vulnerabilidades al software.
- Emplear prácticas de codificación segura a la hora de crear código fuente del software.

- Configurar los procesos de compilación, interpretación y construcción con el objetivo de mejorar la seguridad de los ejecutables. De nuevo, esta práctica permite reducir costes y vulnerabilidades.
- Realizar una auditoría del código fuente (SAST), los scripts y demás código legible por humanos para buscar vulnerabilidades y asegurarse del cumplimiento de los requisitos de seguridad antes de que el software sea liberado.
- Testear el código ejecutable previamente a su puesta en producción para detectar vulnerabilidades y comprobar el cumplimiento de los requisitos de seguridad antes de que el software sea liberado y que las vulnerabilidades puedan ser identificadas y explotadas. En este sentido la ejecución de pruebas DAST (Dynamic application security testing) juegan un papel fundamental.
- Configurar los parámetros del software de forma segura. El objetivo es que, en el momento de la instalación, el software no sea desplegado con configuraciones de seguridad débiles, susceptibles de ser atacadas de forma exitosa. En este sentido, se pueden ejecutar auditorías de seguridad de infraestructura o servicios de pentesting previamente a la puesta en producción.

Reducir la exposición ante ataques de cadena de suministro

Muchas compañías deben considerar contratar servicios para reducir la exposición ante los ataques de cadena de suministro. Dichos servicios deben ejecutarse en múltiples niveles, actuando tanto en las fases de desarrollo de software, como durante el despliegue o cuando ya se ha implantado en producción. Algunos de esos servicios son:

- Servicios SAST (Static Application Security Testing) /SCA (Software composition análisis).
- Identificación de vulnerabilidades en el software o posibles dependencias que hayan sido integradas como parte de la solución final.

- Análisis DAST (Dynamic Application Security Testing). Pruebas dinámicas automatizadas o manuales para garantizar que no se publican vulnerabilidades en entornos productivos o soluciones finales de software.
- Ejecución de auditorías de seguridad para la identificación de posibles vulnerabilidades, facilitando la ejecución del framework de desarrollo seguro de software.

III.-Bibliografía:

Barrio, A. (2022, November 16). *Seguridad en el ciclo de vida del desarrollo de software - Global Technology*. Global Technology. https://globalt4e.com/seguridad-ciclo-de-vida-del-desarrollo-software/

Seguridad en el ciclo de vida de desarrollo del software. (2022). Redhat.com. https://www.redhat.com/es/topics/security/software-development-lifecycle-security

All. (2023, March 22). *NIST y el desarrollo seguro de software*. Tarlogic Security; Tarlogic. https://www.tarlogic.com/es/blog/nist-desarrollo-seguro-de-software/

RÚBRICA DE PRÁCTICA DE EJERCICIOS

Criterios	Escala de Calificación				
Procedimiento / Desarrollo y Resultados	70	60	35	8	0
(70 Puntos)	AUTONOMO (70 puntos) Evidenciar los razonamientos detallados y ordenados, así como las estrategias que se han empleado en el proceso de solución de los ejercicios solicitados. Además, debe presentar los resultados correctos obtenidos de cada ejercicio. Se detecta máxima una falta.	DESTACADO (60 puntos) Se detectan 2 faltas en los aspectos indicados.	SATISFACTORIO (35 puntos) Se detectan 3 faltas en los aspectos indicados.	COMPETENTE (8 puntos) Se detectan 4 o más faltas en los aspectos indicados. (8 puntos)	NO COMPETENTE El alumno no desarrolló el criterio
Formato y Ortografía	15	12	7	4	0
(15 puntos)	AUTONOMO (15 puntos) Sin errores ortográficos y el formato mínimo siguiente: - Tipo de letra legible, tamaño 12. - Interlineado 1.5 - Párrafos justificados - Paginado	DESTACADO (12 puntos) Se detectan 2 faltas en los aspectos indicados.	SATISFACTORIO (7 puntos) Se detectan 3 faltas en los aspectos indicados.	COMPETENTE (4 puntos) Se detectan 4 o más faltas en los aspectos indicados.	NO COMPETENTE El alumno no desarrolló el criterio
Datos de identificación	5	4	3	2	0
(5 puntos)	AUTONOMO (5 puntos) El alumno especifica los siguientes datos: - Alumno - Fecha - Carrera - Grupo - Unidad Temática - Asignatura - Profesor - Título de la práctica Se detecta máximo una falta.	DESTACADO (4 puntos) Se detectan dos faltas en los aspectos indicados	SATISFACTORIO (3 puntos) Se detectan tres faltas en los aspectos indicados.	COMPETENTE (2 puntos) Se detectan cuatro o más faltas en los aspectos indicados.	NO COMPETENTE El alumno no desarrolló el criterio
Bibliografía	10	8	5	2	0
(10 puntos)	AUTONOMO (10 puntos) Reporta la bibliografía que haya sido utilizada para la elaboración del reporte, de acuerdo a las normas establecidas por el APA (American Psychological Association) para citar autores. Se detecta máxima una falta.	DESTACADO (8 puntos) Se detectan 2 faltas en los aspectos indicados.	SATISFACTORIO (5 puntos) Se detectan 3 faltas en los aspectos indicados.	COMPETENTE (2 puntos) Se detectan 4 o más faltas en los aspectos inidicados.	NO COMPETENTE El alumno no desarrolló el criterio

MGTI. César Geovanni Machuca Pereida

Criterios	Clasificación					Puntos
Procedimiento / Desarrollo y Resultados	70 ✓	60	35	8	0	70
Formato y Ortografía	15 ✓	12	7	4	0	15
Datos de Identificación	5 ✓	4	3	2	0	5
Bibliografía	10 ✓	8	5	2	0	10
					Total de puntos	100 /100

Retroalimentación

Mr. Machuca Pereida
Buen trabajo

Vie 23 Jun, 2023 at 10:29 am