Plan de Aseguramiento de Calidad

Proyecto Final

Familiarización con el caso

La empresa CyberX ha pedido desarrollar una aplicación para modelar las interacciones entre atacantes y víctimas en el proceso de ciberseguridad. Explore los principios de la garantía de calidad del software.

El objetivo es garantizar que la aplicación cumpla con altos estándares de calidad, seguridad y confiabilidad. La guía proporcionará instrucciones paso a paso para realizar diversas actividades de control de calidad.

Configuración del entorno:

A. Instalar el software necesario y las herramientas requeridas para la aplicación de ciberseguridad.

Softwares y Herramientas a instalar:

GitHub Desktop: Permite gestionar y sincronizar repositorios, controlar versiones de archivos y facilita la colaboración entre desarrolladores. Al utilizar GitHub Desktop, puedes mantener un registro de los cambios realizados en el código de tu aplicación, facilitando el seguimiento y la colaboración en el desarrollo del software.

Visual Studio Code: Proporciona una amplia gama de características y extensiones que ayudan en el desarrollo de software. Visual Studio Code es conocido por su flexibilidad, soporte para múltiples lenguajes de programación y capacidad para integrarse con herramientas y servicios externos. Puede ser utilizado para escribir, depurar y mantener el código de la aplicación de ciberseguridad.

SonarQube: SonarQube ayuda a identificar problemas comunes en el código, como vulnerabilidades, errores de programación y malas prácticas. Proporciona métricas y análisis detallados para mejorar la calidad del código y ofrece recomendaciones para remediar los problemas encontrados.

ZAP (Zed Attack Proxy): ZAP permite detectar vulnerabilidades y realizar pruebas de penetración en aplicaciones web. Puede ser utilizado para encontrar vulnerabilidades conocidas, realizar pruebas de inyección, escanear en busca de configuraciones incorrectas y mucho más. ZAP ayuda a identificar posibles puntos débiles en la seguridad de la aplicación de ciberseguridad y proporciona recomendaciones para solucionarlos.

Al instalar estos software y herramientas,se proporcionará al equipo de desarrollo las capacidades necesarias para asegurar la calidad, seguridad y confiabilidad de la aplicación de ciberseguridad. Además, se permitirán realizar un seguimiento eficiente al código fuente, analizar la calidad del software y realizar pruebas de seguridad para garantizar que la aplicación cumpla con los estándares requeridos.

Análisis de requerimientos:

A. Revisar los requisitos funcionales y no funcionales de la aplicación de ciberseguridad.

Las herramientas que se van a utilizar para analizar los requisitos funcionales y no funcionales serán SonarQube y ZAP para la aplicación web.

SonarQube es una herramienta de análisis estático de código que permite detectar y corregir problemas de seguridad del código fuente. Con SonarQube, se podrá escanear el código PHP y HTML en busca de vulnerabilidades conocidas, como posibles inyecciones de SQL o problemas de Cross-Site Scripting (XSS). La herramienta proporciona informes detallados sobre los problemas encontrados, lo que te permitirá corregirlos y mejorar la seguridad de tu aplicación.

Por otro lado, ZAP (Zed Attack Proxy) es una herramienta de prueba de seguridad que permite simular ataques a la seguridad de la aplicación web. Con ZAP, se podrán realizar pruebas de penetración en los escenarios de interacción, como el intento de inyección SQL o ataques de Cross-Site Scripting. ZAP ayudará a identificar posibles vulnerabilidades en tu aplicación y te proporcionará informes detallados sobre los problemas encontrados.

Al utilizar SonarQube y ZAP en conjunto, se podrá abordar los requisitos funcionales y no funcionales de seguridad de tu aplicación de manera más efectiva. SonarQube permitirá identificar y corregir problemas de seguridad del código fuente, mientras que ZAP ayudará a probar y validar la seguridad de los escenarios interactivos. Trabajando en conjunto, estas herramientas proporcionarán una cobertura más completa en términos de seguridad, lo que permitirá mejorar la seguridad de la aplicación web.

B. Identificar los requisitos relacionados con la seguridad, como el control de acceso, la autenticación, el cifrado, etc.

- Autenticación: En el formulario de inicio de sesión de los escenarios 1 y 2, se debe implementar un mecanismo de autenticación seguro para verificar la identidad de los usuarios. Esto puede involucrar el uso de contraseñas seguras, técnicas de hashing y almacenamiento seguro de credenciales.

- Validación de entrada: Es fundamental validar y filtrar cualquier entrada de usuario para prevenir ataques de inyección, como la inyección SQL en los escenarios 1 y 2, y el XSS en el escenario 3. La validación de entrada debe asegurarse de que los datos ingresados cumplan con el formato esperado y no contengan contenido malicioso.

- Escapado de salida: En el escenario 4, se requiere el escapado de salida para evitar el XSS. Esto implica aplicar funciones de escape, como htmlspecialchars(), al mostrar los comentarios en la página. De esta manera, cualquier código malicioso incrustado en los comentarios se mostrará como texto seguro en lugar de ejecutarse como código.

- Protección contra vulnerabilidades conocidas: Es importante aplicar las mejores prácticas de seguridad en el desarrollo de PHP y HTML para prevenir vulnerabilidades conocidas, como las vulnerabilidades de inyección SQL y XSS. Esto implica mantenerse actualizado sobre las últimas recomendaciones de seguridad y parches, y aplicar técnicas de codificación segura.

C. Analice el comportamiento esperado del usuario y del atacante en función del propósito de la aplicación.

Modelado de amenazas:

A. Identificar posibles amenazas y vulnerabilidades en la aplicación.

Durante el proceso de análisis y evaluación de la aplicación web, se identificaron las siguientes posibles amenazas y vulnerabilidades:

- Inyección SQL:
  - Amenaza: Un atacante intenta insertar comandos SQL maliciosos a través de los campos de entrada.
  - Vulnerabilidad: Falta de validación y filtrado adecuado de la entrada del usuario en las consultas SQL.
- Cross-Site Scripting (XSS):
  - Amenaza: Un atacante intenta insertar código JavaScript malicioso en los campos de texto para que se ejecute en el navegador de la víctima.
  - Vulnerabilidad: Falta de escape o filtrado adecuado de la salida de los comentarios antes de mostrarlos en la página.

B. Cree modelos de amenazas para comprender los posibles vectores de ataque.

Los modelos de amenazas identificados incluyen:

- Modelo de amenaza para Inyección SQL:
  - Actor: Atacante
  - Objetivo: Ejecutar comandos SQL maliciosos en la base de datos.
  - Método: Intentar inyectar comandos SQL a través de los campos de inicio de sesión.
  - Medidas de mitigación: Validar y filtrar adecuadamente la entrada del usuario, utilizar consultas preparadas y evitar concatenar directamente los datos del usuario en las consultas SQL.
- Modelo de amenaza para Cross-Site Scripting (XSS):
  - Actor: Atacante
  - Objetivo: Insertar código JavaScript malicioso en los comentarios para que se ejecute en el navegador de la víctima.
  - Método: Intentar insertar código JavaScript en los campos de comentarios.
  - Medidas de mitigación: Escapar y filtrar adecuadamente la salida de los comentarios antes de mostrarlos en la página, utilizando funciones como htmlspecialchars().

C. Priorice las amenazas en función de su impacto y probabilidad de ocurrencia.

- Inyección SQL:
    - Impacto: Alto (compromiso de la integridad y confidencialidad de la base de datos).
    - Probabilidad de ocurrencia: Moderada (debido a la falta de validación y filtrado en las consultas SQL).
- Cross-Site Scripting (XSS):
    - Impacto: Moderado (puede permitir el robo de información o el compromiso de la sesión del usuario).
    - Probabilidad de ocurrencia: Moderada (debido a la falta de escape o filtrado en la salida de los comentarios).

Las amenazas identificadas se han priorizado en función de su impacto potencial y la probabilidad de ocurrencia. La priorización se ha realizado considerando los siguientes factores:

- Impacto: Se ha evaluado el impacto potencial de cada amenaza en términos de la confidencialidad, integridad y disponibilidad de los datos y sistemas. Se ha asignado una puntuación alta a las amenazas que podrían comprometer datos sensibles o interrumpir significativamente la funcionalidad del sistema.

- Probabilidad de ocurrencia: Se ha evaluado la probabilidad de ocurrencia de cada amenaza considerando factores como la exposición a Internet, la popularidad de la aplicación y la criticidad de los datos que maneja. Se ha asignado una puntuación alta a las amenazas que tienen una alta probabilidad de ser explotadas.

Planificación de pruebas:
A. Desarrolle un plan de prueba integral que incluya pruebas funcionales y de seguridad.
B. Defina casos de prueba para cubrir varios escenarios de comportamiento de usuarios y atacantes.
C. Considere casos de prueba tanto positivos como negativos para evaluar la respuesta de la aplicación.
Pruebas funcionales:
A. Realice pruebas funcionales para garantizar que la aplicación cumpla con los requisitos previstos.
B. Pruebe varios escenarios de usuario y valide el comportamiento esperado.
C. Verifique que la aplicación maneje las excepciones y las condiciones de error de manera adecuada.
Pruebas de seguridad:
A. Realizar pruebas de seguridad para identificar vulnerabilidades y debilidades.

B. Realice pruebas de penetración para simular el comportamiento del atacante y evaluar la resiliencia del sistema.
C. Pruebe la aplicación contra amenazas de seguridad comunes como ataques de inyección, secuencias de comandos entre sitios, etc.

Pruebas de rendimiento:

A. Evaluar el desempeño de la aplicación de ciberseguridad bajo diferentes cargas y condiciones de estrés.
B. Mida los tiempos de respuesta, el rendimiento y la utilización de recursos.
C. Identifique y aborde cualquier cuello de botella de rendimiento o problemas de escalabilidad.

Registro y Monitoreo:

A. Implemente mecanismos de registro para capturar actividades relevantes de usuarios y atacantes.
B. Configure herramientas de monitoreo para rastrear el comportamiento del sistema y detectar patrones sospechosos.
C. Defina alertas y notificaciones para posibles incidentes de seguridad.

Documentación e informes:

A. Documente todos los resultados de las pruebas, incluidos los problemas identificados, las vulnerabilidades y su gravedad.
B. Preparar un informe completo que resuma las actividades de aseguramiento de la calidad y sus resultados.
C. Proporcione recomendaciones para mejorar la seguridad de la aplicación y la calidad general.

Establecimiento de indicadores clave de rendimiento (KPI):

- Utilizar herramientas de seguimiento y monitoreo de proyectos como Jira o Trello para registrar y medir el cumplimiento de los objetivos de calidad en cada etapa del desarrollo.
- Establecer métricas específicas, como la tasa de vulnerabilidades identificadas y resueltas, el tiempo promedio de respuesta ante incidencias de seguridad, etc.

Implementación de un sistema de revisión y mejora continua de los procesos:

- Realizar revisiones periódicas de código con la colaboración de los desarrolladores, utilizando herramientas como GitHub o GitLab para facilitar la revisión de cambios y la detección temprana de problemas de seguridad.
- Realizar retrospectivas de proyectos para identificar oportunidades de mejora y planificar acciones correctivas o preventivas para futuros desarrollos.

Definición de roles y responsabilidades claros:

- Crear un documento formal de Roles y Responsabilidades que defina las funciones del equipo de calidad, así como las responsabilidades de los desarrolladores y otros miembros del equipo.

- Utilizar herramientas de gestión de tareas para asignar y controlar las responsabilidades de cada miembro del equipo.

Procedimientos para la gestión de la configuración y cambios:

- Utilizar sistemas de control de versiones como Git para gestionar el código fuente y la documentación relacionada.
- Establecer un flujo de trabajo claro para la revisión y aprobación de cambios en el código y la documentación.

Políticas y directrices para el aseguramiento de la calidad:

- Crear una guía de buenas prácticas de desarrollo seguro que incluya técnicas de validación de entrada, escapado de salida y sanitización de datos para prevenir vulnerabilidades comunes.
- Proporcionar capacitación y talleres sobre temas de seguridad y calidad para el equipo de desarrollo.

SUSTENTO TEÓRICO

SQA (Sofware Quality Assurance o Aseguramiento de la Calidad del Software) implica revisar y auditar los productos y actividades de software para verificar que se cumplen los procedimientos y los estándares, además de proveer a las gerencias apropiadas (incluyendo a la de proyectos) con los resultados de estas revisiones. Por lo tanto, SQA envuelve al PROCESO de desarrollo de software completo: monitoreando y mejorando el proceso; asegurándose que cualquier estándar y procedimientos adoptados sean seguidos; y, asegurándose que los problemas sean encontrados y tratados.

Definición

SQA es un conjunto de actividades sistemáticas que aseguran que el proceso del software y productos conformados por requerimientos, estándares, y procedimientos. Los procesos incluyen todas las actividades involucradas en el diseño, codificación, pruebas y mantenimiento; Los productos incluyen software, datos asociados, documentación, y toda la documentación para soporte y reportes.

Propósito

Proporcionar visibilidad sobre los procesos utilizados por el proyecto de software y sobre los productos que genera.

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

REQUERIMIENTOS DE LA APLICACIÓN

Requisitos Funcionales:

Portada:

- Mostrar información relevante sobre ciberseguridad, como consejos de seguridad, estadísticas o noticias recientes.
- Incluir botones o enlaces que redirigen a los diferentes escenarios de interacción entre víctima y atacante.

Escenario 1 - Vulnerabilidad de Inyección SQL:

- Mostrar un formulario de inicio de sesión que solicite un nombre de usuario y una contraseña.
- Permitir que un atacante intente realizar una inyección SQL en los campos de entrada.
- Implementar el código PHP que procese los datos ingresados y realice una consulta SQL vulnerable a inyección.
- Mostrar en la página los resultados de la consulta para que el atacante observe el éxito de la inyección.

Escenario 2 - Solución de Inyección SQL con PDO:

- Mostrar un formulario de inicio de sesión similar al escenario anterior.
- Utilizar PDO (PHP Data Objects) para conectarse a la base de datos y realizar consultas seguras.
- Implementar marcadores de posición en las consultas preparadas para evitar la inyección SQL.
- Mostrar en la página los resultados seguros de la consulta para que se observe la protección contra la inyección.

Escenario 3 - Ataque de Cross-Site Scripting (XSS):

- Mostrar una página donde los usuarios puedan ingresar comentarios o mensajes en un campo de texto.
- Permitir que un atacante intente insertar código JavaScript malicioso en los comentarios.
- Implementar el código PHP que muestre los comentarios ingresados sin escape o filtrado de contenido.
- Mostrar en la página cómo el código malicioso se ejecuta en el navegador de la víctima.

Escenario 4 - Solución para evitar el ataque de XSS:

- Mantener la página de comentarios del escenario anterior.
- Modificar el código PHP para aplicar la función htmlspecialchars() al mostrar los comentarios, escapando los caracteres especiales y evitando la ejecución de código JavaScript malicioso.
- Mostrar en la página cómo los comentarios ahora se muestran de forma segura, sin ejecutar el código malicioso.

Requisitos No Funcionales:

Seguridad:

- Implementar las mejores prácticas de seguridad en el desarrollo de PHP y HTML para prevenir vulnerabilidades.
- Utilizar técnicas de validación de entrada, escapado de salida y sanitización de datos para proteger contra ataques comunes.

Usabilidad:

- Diseñar una interfaz de usuario intuitiva y atractiva.
- Asegurarse de que los usuarios puedan navegar fácilmente entre la portada y los diferentes escenarios.

Rendimiento:

- Optimizar el rendimiento de la página web para una carga rápida.
- Minimizar el uso de recursos y optimizar las consultas SQL.

Compatibilidad:

- Garantizar que la página web sea compatible con los principales navegadores web (Chrome, Firefox, Safari, etc.).
- Asegurarse de que la página sea responsiva y se adapte a diferentes dispositivos y tamaños de pantalla.