

**GASIC 8– Seguridad en el Desarrollo y Adquisición de Software**

**Preguntas de Auditoría**

**Pruebas y Verificación**

**Competencias Básicas del Auditor**

1. **Conocimiento Profundo en Desarrollo de Software Seguro:** Entender los principios de seguridad en el desarrollo de software, incluyendo buenas prácticas de codificación, diseño seguro, y consideraciones de seguridad en el ciclo de vida del desarrollo.
2. **Experiencia en Identificación de Vulnerabilidades:** Tener habilidades para identificar y evaluar vulnerabilidades en el código y el diseño del software, incluyendo análisis estático y dinámico, pruebas de penetración y revisión de arquitecturas.
3. **Conocimiento en Criptografía y Protección de Datos:** Comprender los principios de la criptografía y cómo se aplica en la protección de datos sensibles dentro del software, incluyendo la gestión segura de claves y la integridad de los datos.
4. **Habilidades en Pruebas de Seguridad y Evaluación de Riesgos:** Ser capaz de realizar pruebas de seguridad exhaustivas para evaluar la resistencia del software frente a posibles ataques, así como evaluar los riesgos asociados con el software.
5. **Conocimientos en Cumplimiento y Normativas:** Estar familiarizado con los requisitos de cumplimiento y normativas relevantes en el ámbito de la seguridad del software, como GDPR, PCI DSS, y normas ISO/IEC.
6. **Capacidad Analítica:** Ser capaz de analizar datos de seguridad y tendencias para identificar posibles amenazas y vulnerabilidades emergentes en el software.
7. **Habilidades de Comunicación Técnica:** Ser capaz de comunicar de manera efectiva los hallazgos de seguridad a los desarrolladores, gerentes y otros interesados, y proponer recomendaciones claras y prácticas.
8. **Conocimientos en Gestión de Incidentes y Respuesta a Incidentes:** Entender los procedimientos y prácticas para la gestión de incidentes de seguridad en el software, incluyendo la capacidad de respuesta rápida y eficaz.

# Visión General

Las pruebas y la verificación son componentes esenciales para identificar vulnerabilidades y garantizar la seguridad del software durante su desarrollo. Un enfoque estructurado que combine revisiones de código, análisis automatizados y pruebas funcionales permite validar que el software cumpla con los estándares organizacionales y mitigue los riesgos asociados. Desde una perspectiva de auditoría, la documentación, el cumplimiento de políticas y la gestión adecuada de los resultados son fundamentales para evaluar la efectividad de estas actividades.

**1.1 Revisión y Análisis de Código**

Las revisiones de código deben realizarse siguiendo las políticas y pautas de la organización, asegurando que se adapten a la etapa de desarrollo del software. Estas revisiones deben incluir evaluaciones por pares y el uso de herramientas especializadas para detectar puertas traseras, contenido malicioso y vulnerabilidades. Un auditor verificará que los procesos de revisión estén documentados y que las herramientas empleadas cumplan con los estándares organizacionales, evaluando si los resultados se han manejado adecuadamente.

**1.2 Pruebas Funcionales y de Seguridad**

Las pruebas deben abarcar tanto las funcionalidades generales del software como características específicas de seguridad, integrando casos de prueba para vulnerabilidades previamente identificadas. Herramientas como pruebas difusas y análisis dinámico deben ser utilizadas para identificar problemas relacionados con la entrada de datos y vulnerabilidades en tiempo de ejecución. Las pruebas de penetración, especialmente en escenarios de alto riesgo, deben realizarse con personal cualificado y casos de prueba adaptados a la aplicación. Desde el punto de vista de auditoría, es crucial revisar la trazabilidad de estas pruebas y garantizar que los resultados se gestionen conforme a la gestión de riesgos de la organización.

**1.3 Automatización y Monitoreo Continuo**

El uso de herramientas automatizadas permite identificar y abordar prácticas de software inseguras de manera continua, integrando pruebas dinámicas de vulnerabilidad en los flujos de trabajo automatizados. Estas herramientas deben estar configuradas para generar informes claros y precisos que se puedan revisar periódicamente. Un auditor verificará que estas pruebas automatizadas estén alineadas con las políticas de seguridad y que sus resultados se integren en la gestión central de defectos y riesgos.

**1.4 Gestión de Resultados y Mejora Continua**

La documentación de los resultados de las pruebas, incluyendo las lecciones aprendidas y las causas fundamentales de los problemas, es esencial para mejorar continuamente el proceso de pruebas. Los resultados deben consolidarse en un panel central accesible para las partes interesadas, quienes deben revisarlos y gestionarlos según las políticas de riesgos de la organización. Además, los hallazgos deben incorporarse en la capacitación y los manuales de pruebas de seguridad para fortalecer las capacidades futuras. Un auditor evaluará si estos procesos están documentados y si se siguen de manera consistente.

**1.5 Integración de Actividades de Seguridad**

Las pruebas de seguridad deben beneficiarse de los resultados de otras actividades de seguridad, como auditorías o análisis previos, para optimizar su alcance y efectividad. Este enfoque asegura que las vulnerabilidades detectadas se gestionen adecuadamente y se incorporen a un ciclo de mejora continua. Desde el punto de vista de auditoría, se revisará si esta integración es efectiva y si los resultados se utilizan para fortalecer la postura de seguridad de la organización.

# Modelo de Madurez

|  |  |
| --- | --- |
| NIVEL 1 [Impredecible y reactivo]: La actividad se completa, pero a menudo se retrasa y supera el presupuesto. | 1. Seguir las políticas o pautas de la organización sobre cuándo se debe realizar la revisión del código y cómo se debe realizar. 2. Seguir las políticas o pautas de la organización sobre cuándo se debe realizar el análisis de código y cómo se debe realizar. 3. Elegir métodos de revisión y/o análisis de código según la etapa del software. 4. Seguir políticas o pautas para realizar pruebas de código y cómo se deben hacer. |
| NIVEL 2 [Gestionado a nivel de proceso]: Las actividades se planifican, ejecutan, miden y controlan. | 1. Realizar una revisión por pares del código y revisar cualquier revisión, análisis o resultado de prueba de código existente como parte de la revisión por pares. 2. Utilizar revisores expertos para comprobar el código en busca de puertas traseras y otro contenido malicioso. 3. Utilizar herramientas de revisión por pares que faciliten el proceso y documenten las discusiones y comentarios. 4. Utilizar herramientas de análisis estático para verificar automáticamente el código en busca de vulnerabilidades y cumplimiento de estándares, con una revisión humana de los problemas. 5. Utilizar listas de verificación de revisión para verificar el cumplimiento de los requisitos. 6. Realizar pruebas funcionales sólidas de las características de seguridad. 7. Incorporar pruebas para vulnerabilidades informadas previamente en el conjunto de pruebas. 8. Considerar la infraestructura y tecnología para desarrollar planes de prueba. 9. Utilizar herramientas de prueba difusa para encontrar problemas con el manejo de entradas. 10. Realizar pruebas de penetración en escenarios de alto riesgo si hay recursos disponibles. |
| NIVEL 3 [Proactivo antes que reactivo]: Los estándares de toda la organización brindan orientación a través de proyectos, programas y carteras. | 1. Utilizar herramientas automatizadas para identificar y remediar prácticas de software inseguras continuamente.   1.1 Integrar pruebas dinámicas de vulnerabilidad en el conjunto de pruebas automatizadas.  2. Las pruebas de penetración utilizan casos de prueba de seguridad específicos de la aplicación para evaluar la seguridad   2.1 Las pruebas de penetración buscan problemas técnicos y lógicos en la aplicación.   2.2 Las partes interesadas revisan los resultados de las pruebas y los manejan de acuerdo con la gestión de riesgos de la organización.   2.3 Personal cualificado realiza las pruebas de penetración |
| NIVEL 4 [Gestionado Cuantitativamente]: La organización está basada en datos con objetivos cuantitativos de mejora del desempeño que son predecibles | 1. Documentar las lecciones aprendidas en una wiki accesible para los desarrolladores.  2. Identificar y documentar las causas fundamentales de los problemas.  3. La dirección y las partes interesadas hacen un seguimiento y revisan los resultados de las pruebas a lo largo del ciclo de desarrollo.   3.1 Los resultados de las pruebas se fusionan en un panel central y se introducen en la gestión de defectos. |
| NIVEL 5 [Optimizado]: La organización se centra en la mejora continua y está diseñada para girar y responder a las oportunidades y los cambios. | 1. Utiliza los resultados de otras actividades de seguridad para mejorar las pruebas de seguridad integradas durante el desarrollo.  2.1 Revisa los resultados de las pruebas y los incorpora a la formación sobre seguridad y a los manuales de pruebas de seguridad  3.2 Las partes interesadas revisan los resultados de las pruebas y los tratan de acuerdo con la gestión de riesgos de la organización |

1. **Prácticas de auditoría para Pruebas y Verificación**

|  |
| --- |
| Temática: Pruebas de penetración |
| Es un proceso crítico que involucra la evaluación exhaustiva de las medidas de seguridad implementadas en un sistema. Durante esta auditoría, se analizan las pruebas de penetración realizadas para identificar vulnerabilidades y posibles amenazas a la integridad, confidencialidad y disponibilidad de la información. Este proceso implica revisar detalladamente los informes de las pruebas, validar la efectividad de las contramedidas implementadas y asegurarse de que se hayan abordado adecuadamente todas las debilidades identificadas. Además, la auditoría se centra en garantizar que el proceso de desarrollo y adquisición de software cumpla con las mejores prácticas de seguridad, estándares y regulaciones aplicables. La finalidad es fortalecer la resiliencia del sistema ante posibles ataques cibernéticos, asegurando un entorno de software robusto y protegido contra amenazas potenciales.  Este levantamiento puede incluir análisis específicos tales como:   * ¿Cómo se seleccionan los objetivos de las pruebas de penetración en el desarrollo de software? * ¿Cuáles son los criterios y metodologías utilizados para identificar los objetivos de las pruebas de penetración, y están alineados con las características y vulnerabilidades potenciales del software? * ¿Cuál es el enfoque para la realización de pruebas de penetración en el ciclo de desarrollo de software? * ¿Cómo se planifican, ejecutan y documentan las pruebas de penetración a lo largo del ciclo de desarrollo, y están integradas de manera efectiva en los procesos existentes? * ¿Cómo se documentan y comunican los hallazgos de las pruebas de penetración? * ¿Son claros, exhaustivos y relevantes los informes de pruebas de penetración, y se implementan efectivamente las recomendaciones resultantes? * ¿Cómo se gestionan y priorizan las vulnerabilidades identificadas en las pruebas de penetración? * ¿Qué procesos y herramientas se utilizan para la clasificación, priorización y gestión de vulnerabilidades, y se asignan recursos de manera proporcional a la gravedad y probabilidad de explotación? * ¿Cómo se garantiza la confidencialidad y seguridad de la información durante y después de las pruebas de penetración? * ¿Qué controles y medidas se implementan para proteger la información confidencial durante todas las fases de las pruebas de penetración, desde la planificación hasta la disposición segura de los datos recopilados? |

|  |
| --- |
| Temática: Infraestructura y Tecnología para Desarrollar Planes de Prueba. |
| Es un procedimiento esencial para garantizar la efectividad y robustez de los controles de seguridad. Esta auditoría implica evaluar minuciosamente la arquitectura tecnológica utilizada para llevar a cabo las pruebas, así como los recursos de hardware y software empleados en el entorno de desarrollo. Se busca asegurar que la infraestructura sea capaz de simular de manera precisa escenarios de amenazas realistas y proporcionar un entorno de prueba que refleje fielmente la complejidad del sistema final. Además, la auditoría se centra en verificar la integridad y confidencialidad de los datos utilizados en las pruebas, así como en garantizar que la infraestructura cumpla con los estándares de seguridad establecidos, contribuyendo así a la identificación temprana y mitigación de posibles vulnerabilidades en el software en desarrollo.  Este levantamiento puede incluir análisis específicos tales como:   * ¿Cómo se selecciona la infraestructura de prueba para reflejar de manera precisa el entorno de producción? * ¿La documentación del proceso de selección de la infraestructura de prueba refleja adecuadamente la arquitectura y configuración del entorno de producción? * ¿Cómo se asegura la integridad y confidencialidad de los datos utilizados en las pruebas de seguridad? * ¿Qué controles se han implementado, como la encriptación y la gestión de accesos, para proteger los datos de prueba y garantizar la integridad y confidencialidad de la información? * ¿Cómo se gestionan las configuraciones de la infraestructura de prueba y se asegura su consistencia con los entornos de desarrollo y producción? * ¿Los procedimientos y herramientas utilizadas para gestionar las configuraciones de la infraestructura de prueba están alineados con los entornos de desarrollo y producción? * ¿Cómo se monitorea y registra la actividad en la infraestructura de prueba para la detección temprana de posibles anomalías o ataques? * ¿Las políticas y herramientas de monitorización implementadas, junto con los registros de actividad, permiten una detección y respuesta efectiva ante eventos inesperados? * ¿Cómo se garantiza la redundancia y disponibilidad de la infraestructura de prueba para evitar interrupciones en el proceso de desarrollo? * ¿La arquitectura de la infraestructura incluye medidas de redundancia y procedimientos para asegurar la disponibilidad continua de los entornos de prueba? |