

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

**Preguntas de Auditoría**

**Codificación Segura**

**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

Incorporar controles en las prácticas de codificación asegura que el software cumpla con estándares de seguridad, minimizando los riesgos asociados a errores comunes de programación. Desde la perspectiva de auditoría, es esencial que los desarrolladores sigan procedimientos establecidos, utilicen herramientas automatizadas y mantengan entornos controlados para garantizar la seguridad del código producido.

**1.1 Validación y Manejo de Entradas y Salidas**

Todas las entradas deben ser validadas y las salidas correctamente codificadas para prevenir ataques comunes, como inyecciones de código o desbordamientos de búfer. Las funciones inseguras deben ser reemplazadas por alternativas seguras, y cualquier error encontrado durante la ejecución debe manejarse de manera elegante y documentada. Desde el punto de vista de un auditor, se debe verificar que estas prácticas sean consistentes y que los registros generados permitan rastrear problemas en caso de incidentes.

**1.2 Uso de Entornos y Herramientas Automatizadas**

Los entornos de desarrollo deben incorporar herramientas que promuevan prácticas de codificación seguras, como linters y formateadores, además de incluir capacitación en tiempo real. Estas herramientas deben estandarizar el código fuente y buscar vulnerabilidades específicas del lenguaje y del entorno de desarrollo. Un auditor revisará si estas herramientas están correctamente configuradas y si los desarrolladores las utilizan de manera efectiva en el flujo de trabajo.

**1.3 Control de Compiladores y Herramientas de Compilación**

La organización debe utilizar versiones actualizadas de compiladores, intérpretes y herramientas de compilación, habilitando funciones que generen advertencias sobre código inseguro. El concepto de "compilación limpia", donde todas las advertencias se tratan como errores hasta que se justifiquen, debe implementarse para fortalecer la seguridad del código. Desde la perspectiva de auditoría, es crucial validar periódicamente la autenticidad e integridad de estas herramientas y garantizar que las configuraciones aprobadas estén disponibles y sean utilizadas consistentemente.

**1.4 Configuración y Entornos Controlados**

Las compilaciones deben realizarse en entornos dedicados y altamente controlados, donde las configuraciones de herramientas aprobadas estén disponibles como código para facilitar su uso por parte de los desarrolladores. Estas configuraciones deben incluir medidas como la aleatorización u ofuscación de características de ejecución para reducir la predictibilidad y la explotabilidad del software. Un auditor evaluará la documentación y trazabilidad de estos entornos, asegurándose de que se mantengan alineados con las políticas de seguridad.

**1.5 Revisión y Mejora Continua**

El código debe ser revisado por los desarrolladores mismos y por otros miembros del equipo o herramientas especializadas para identificar posibles vulnerabilidades. Además, se deben probar las funciones del software para garantizar que cumplan con los requisitos esperados sin introducir problemas operativos. Desde una perspectiva de auditoría, estos procesos de revisión deben estar documentados, y los hallazgos deben ser gestionados para su resolución en el menor tiempo posible.

# Modelo de Madurez

|  |  |
| --- | --- |
| NIVEL 1 [Impredecible y reactivo]: La actividad se completa, pero a menudo se retrasa y supera el presupuesto. | 1. Validar todas las entradas y validar y codificar correctamente todas las salidas. 2. Evitar el uso de funciones y llamadas no seguras. 3. Detectar errores y manejarlos con elegancia. 4. Proporcionar capacidades de registro y seguimiento. 5. Utilizar entornos de desarrollo con funciones automatizadas que fomenten o requieran el uso de prácticas de codificación seguras con capacitación en el lugar justo a tiempo. 6. Seguir los procedimientos para garantizar manualmente el cumplimiento de prácticas de codificación segura cuando los métodos automatizados sean insuficientes o no estén disponibles. 7. Utilizar herramientas (por ejemplo, linters, formateadores) para estandarizar el estilo y el formato del código fuente. 8. Buscar otras vulnerabilidades que sean comunes a los lenguajes y el entorno de desarrollo. 9. Hacer que el desarrollador revise su propio código legible por humanos para complementar (no reemplazar) la revisión del código realizada por otras personas o herramientas. |
| NIVEL 2 [Gestionado a nivel de proceso]: Las actividades se planifican, ejecutan, miden y controlan. | 1. Utilizar versiones actualizadas del compilador, intérprete y herramientas de compilación. 2. Seguir los procesos de gestión de cambios al implementar o actualizar el compilador, el intérprete y las herramientas de compilación, y auditar todos los cambios inesperados en las herramientas. 3. Validar periódicamente la autenticidad y la integridad del compilador, el intérprete y las herramientas de compilación. |
| NIVEL 3 [Proactivo antes que reactivo]: Los estándares de toda la organización brindan orientación a través de proyectos, programas y carteras. | 1. Habilitar las funciones del compilador que generan advertencias sobre código mal protegido durante el proceso de compilación. 2. Implementar el concepto de "compilación limpia", donde todas las advertencias del compilador se tratan como errores y se eliminan, excepto aquellas que se consideran falsos positivos o irrelevantes. 3. Realizar todas las compilaciones en un entorno de compilación dedicado y altamente controlado. 4. Habilitar funciones del compilador que aleatorizan u ofuscan las características de ejecución, como el uso de la ubicación de la memoria, que de otro modo serían predecibles y, por lo tanto, potencialmente explotables. 5. Probar para asegurarse de que las funciones funcionen como se esperaba y no causen inadvertidamente problemas operativos u otros problemas. 6. Verificar continuamente que se estén utilizando las configuraciones aprobadas. 7. Hacer que las configuraciones de herramientas aprobadas estén disponibles como configuración como código para que los desarrolladores puedan usarlas fácilmente. |
| NIVEL 4 [Gestionado Cuantitativamente]: La organización está basada en datos con objetivos cuantitativos de mejora del desempeño que son predecibles |  |
| 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. **Preguntas de Auditoría para Codificación Segura**

|  |
| --- |
| Temática: Validación y Codificación Segura de Entradas y Salidas |
| La validación y codificación correcta de entradas y salidas es esencial para prevenir vulnerabilidades en el software, como inyecciones y desbordamientos de búfer. Implementar estos controles asegura que el software maneje adecuadamente los datos que recibe y genera, reduciendo el riesgo de explotación por parte de atacantes. La auditoría de estos controles es crucial para verificar que todas las entradas sean validadas y todas las salidas sean codificadas correctamente, asegurando la robustez y seguridad del software.  Este levantamiento puede incluir análisis específicos tales como:   * ¿Se validan todas las entradas de usuario y de sistemas externos antes de ser procesadas? * ¿Se codifican correctamente todas las salidas antes de ser enviadas a los usuarios o sistemas externos? * ¿Existen procedimientos documentados para la validación y codificación de datos? * ¿Se evita el uso de funciones y llamadas conocidas por ser inseguras en el desarrollo de software? * ¿Se utilizan bibliotecas y funciones seguras como alternativas a las no seguras? * ¿Se detectan y manejan los errores de manera elegante y segura? * ¿Existen mecanismos para registrar y analizar los errores detectados? * ¿Se proporcionan capacidades adecuadas de registro y seguimiento para monitorear eventos y actividades de seguridad? * ¿Se revisan regularmente los registros de eventos para detectar posibles incidentes de seguridad? * ¿Se utilizan entornos de desarrollo que fomenten o requieran el uso de prácticas de codificación seguras con capacitación en el lugar justo a tiempo? * ¿Están disponibles estas funciones automatizadas para todos los desarrolladores? * ¿Se siguen procedimientos para garantizar manualmente el cumplimiento de prácticas de codificación segura cuando los métodos automatizados son insuficientes o no están disponibles? * ¿Se realizan revisiones manuales del código para complementar las herramientas automatizadas? * ¿Se utilizan herramientas como linters y formateadores para estandarizar el estilo y el formato del código fuente? * ¿Se revisa regularmente el código para asegurar que cumpla con los estándares establecidos? * ¿Se buscan y mitigan otras vulnerabilidades que sean comunes a los lenguajes y el entorno de desarrollo utilizado? * ¿Se realizan auditorías periódicas para identificar nuevas vulnerabilidades? * ¿Se requiere que los desarrolladores revisen su propio código para complementar la revisión realizada por otras personas o herramientas? * ¿Se documentan y abordan las observaciones realizadas durante estas revisiones? |

|  |
| --- |
| Temática: Gestión y Validación de Herramientas de Compilación |
| Utilizar versiones actualizadas de compiladores, intérpretes y herramientas de compilación es crucial para garantizar la seguridad y eficiencia del proceso de desarrollo. Evaluar estos controles asegura que las herramientas utilizadas sean seguras, auténticas y estén correctamente gestionadas, minimizando el riesgo de vulnerabilidades y errores en el software final.  Este levantamiento puede incluir análisis específicos tales como:   * ¿Se utilizan versiones actualizadas del compilador, intérprete y herramientas de compilación en el proceso de desarrollo? * ¿Existen procedimientos para mantener estas herramientas actualizadas y seguras? * ¿Se siguen los procesos de gestión de cambios al implementar o actualizar el compilador, intérprete y herramientas de compilación? * ¿Se auditan todos los cambios inesperados en las herramientas para identificar posibles problemas de seguridad? * ¿Se valida periódicamente la autenticidad y la integridad del compilador, intérprete y herramientas de compilación utilizadas? * ¿Se utilizan mecanismos como firmas digitales para asegurar la autenticidad de estas herramientas? |

|  |
| --- |
| Temática: Prácticas de Compilación Segura |
| Implementar prácticas de compilación segura es esencial para garantizar que el software compilado sea seguro y libre de vulnerabilidades. Evaluar estos controles ayuda a asegurar que las configuraciones y procesos de compilación se adhieran a los estándares de seguridad, minimizando el riesgo de explotación de vulnerabilidades.  Este levantamiento puede incluir análisis específicos tales como:   * ¿Se habilitan las funciones del compilador que generan advertencias sobre código mal protegido durante el proceso de compilación? * ¿Cómo se gestionan y solucionan estas advertencias? * ¿Se implementa el concepto de "compilación limpia", donde todas las advertencias del compilador se tratan como errores y se eliminan? * ¿Se documentan las excepciones como falsos positivos o advertencias irrelevantes? * ¿Se realizan todas las compilaciones en un entorno de compilación dedicado y altamente controlado? * ¿Se aseguran de que este entorno esté protegido contra accesos no autorizados y otras amenazas? * ¿Se habilitan funciones del compilador que aleatorizan u ofuscan características de ejecución para reducir la predictibilidad? * ¿Se prueban estas funciones para asegurar que no causen problemas operativos o de rendimiento? * ¿Se prueban las funcionalidades del software para asegurarse de que funcionen como se esperaba y no causen problemas inadvertidos? * ¿Cómo se gestionan los problemas detectados durante estas pruebas? * ¿Se verifica continuamente que se estén utilizando las configuraciones de compilación aprobadas? * ¿Se documentan y revisan las configuraciones para asegurar su adherencia a los estándares de seguridad? * ¿Se hacen disponibles las configuraciones de herramientas aprobadas como configuración como código para facilitar su uso por los desarrolladores? * ¿Se revisan y actualizan estas configuraciones regularmente para mantener la seguridad y la eficiencia? |