Tabla de contenido

1. Propósito

El presente documento tiene como objetivo establecer el marco normativo y las directrices técnicas para asegurar que todo software desarrollado, adquirido o mantenido por el H. Ayuntamiento de Morelia sea intrínsecamente seguro. Se busca integrar la seguridad como un pilar fundamental en cada fase del **Ciclo de Vida de Desarrollo de Software (SDLC)**,

desde la concepción de la idea hasta el despliegue en producción y su posterior mantenimiento.

A través de esta política, se persiguen los siguientes objetivos estratégicos:

- **Seguridad desde el Diseño (Security by Design):** Garantizar que los requisitos de seguridad sean definidos y documentados desde el inicio mediante la **Bitácora de Requerimientos de Desarrollo (BRD)**, evitando parches de seguridad costosos en etapas avanzadas.
- **Mitigación de Vulnerabilidades Críticas:** Reducir la superficie de ataque de las aplicaciones municipales mediante la aplicación de estándares internacionales (como **OWASP Top 10**), asegurando que el código sea resistente a inyecciones, ataques de autenticación y fugas de datos.
- **Validación Técnica Rigurosa:** Institucionalizar la ejecución de pruebas de seguridad tanto estáticas como dinámicas, utilizando el **Formato Pruebas de Seguridad SAST / DAST** para identificar y remediar fallos antes de que el software sea expuesto a internet.
- **Cumplimiento de Privacidad:** Asegurar que el manejo de los datos de la ciudadanía de Morelia cumpla con las leyes de protección de datos, integrando controles de enmascaramiento y cifrado desde la fase de codificación.
- **Estandarización y Calidad del Código:** Proveer a los desarrolladores de un conjunto de reglas claras para una codificación limpia y segura, verificando su cumplimiento a través del **Formato Verificación de Requisitos de Desarrollo Seguro**.
- **Gobernanza del Desarrollo:** Mantener una trazabilidad total sobre quién, qué y cómo se modifica el software institucional, asegurando que cada liberación a producción sea el resultado de un proceso controlado y auditado.

2. Alcance

Esta política es de **carácter obligatorio** para todo el personal y entes externos involucrados en la creación y mantenimiento de software. El alcance se define bajo los siguientes criterios:

2.1. Alcance sobre el Personal

- **Desarrolladores Internos:** Personal de base, confianza o contrato que realice labores de programación para el Ayuntamiento.
- **Proveedores y Fábricas de Software:** Empresas externas contratadas para desarrollar soluciones a medida o implementar módulos adicionales.
- **Administradores de Bases de Datos (DBAs) y DevOps:** Responsables de la configuración de los entornos donde reside el código.

2.2. Alcance sobre el Software y Activos Lógicos

La aplicación de los controles (BRD, SAST/DAST y Verificación) rige sobre:

- **Aplicaciones Web y Portales Ciudadanos:** (Ej. Portal de pagos, ventanilla única de trámites, Catastro en línea).
- **Sistemas Administrativos Internos:** (Ej. Sistemas de nómina, control de inventarios, gestión de archivos).
- **Servicios Web y APIs:** Cualquier interfaz de comunicación entre sistemas internos o con entes externos (Gobierno Estatal/Federal).
- **Aplicaciones Móviles:** Apps oficiales desarrolladas para uso de la ciudadanía o del personal operativo.

2.3. Alcance sobre el Ciclo de Vida (Entornos)

La seguridad no es exclusiva de producción; los controles se aplican en:

- **Entorno de Desarrollo:** Aplicación de guías de codificación segura.
- **Entorno de Pruebas / QA:** Ejecución obligatoria de pruebas de seguridad estáticas y dinámicas.
- **Entorno de Producción:** Monitoreo de vulnerabilidades en tiempo de ejecución.

2.4. Alcance Tecnológico

Aplica a cualquier lenguaje de programación (PHP, Java, Python, .NET, JavaScript, etc.), frameworks, gestores de bases de datos y librerías de terceros (Open Source) que se utilicen en los proyectos institucionales.

3. Políticas de Desarrollo y Codificación Segura

3.1. Requisitos de Seguridad desde la Concepción

La seguridad no es un accesorio, es un requisito funcional.

- **Documentación Obligatoria:** Todo proyecto de software, interno o externo, debe iniciar con el llenado de la **Bitácora de Requerimientos de Desarrollo (BRD)**. En este documento se deben identificar las necesidades de cifrado, niveles de acceso, tipos de datos personales a manejar y registros de auditoría necesarios.
- **Aprobación de Requisitos:** Ningún desarrollo podrá pasar a la fase de construcción sin que la Dirección de TI haya validado que los requerimientos de seguridad en la BRD son suficientes para proteger la información institucional.

3.2. Estándares de Codificación Segura

Se prohíbe la programación basada únicamente en la funcionalidad sin considerar la higiene del código.

- **Adopción de Frameworks Seguros:** Se priorizará el uso de frameworks y lenguajes actualizados que cuenten con protecciones nativas contra ataques comunes.
- **Alineación con OWASP:** Todos los desarrollos deben seguir las guías de **OWASP (Open Web Application Security Project)** para mitigar el "Top 10" de riesgos de seguridad, tales como Inyecciones SQL, Cross-Site Scripting (XSS) y exposición de datos sensibles.
- **Manejo de Errores y Excepciones:** El software no debe mostrar mensajes de error técnicos (stack traces, rutas de archivos o versiones de BD) al usuario final, ya que estos pueden ser utilizados por atacantes para realizar reconocimiento del sistema.

3.3. Segregación de Ambientes y Funciones

Para evitar errores humanos y sabotajes, se establecen fronteras claras:

- **Entornos Aislados:** Es obligatorio contar con al menos tres entornos separados físicamente: **Desarrollo (Dev)**, **Pruebas (Staging/QA)** y **Producción (Prod)**.
- **Prohibición de Datos Reales:** Queda estrictamente prohibido utilizar datos reales de los ciudadanos en los entornos de Desarrollo o QA. Se deben utilizar datos ficticios o aplicar el procedimiento de **Enmascaramiento de Datos**.

3.4. Pruebas de Seguridad y Análisis de Vulnerabilidades

La confianza no es un control; la verificación sí lo es.

- **Análisis Estático y Dinámico:** Todo software debe ser sometido a revisión mediante el **Formato Pruebas de Seguridad SAST / DAST**.
  - **SAST:** Para revisar el código fuente en busca de debilidades lógicas.
  - **DAST:** Para probar la aplicación en tiempo de ejecución contra ataques externos.
- **Remediación de Hallazgos:** No se permitirá el paso a producción de aplicaciones que presenten vulnerabilidades clasificadas como "Críticas" o "Altas" en el reporte de pruebas.

3.5. Seguridad en el Uso de Software de Terceros y Open Source

Dado que muchas brechas ocurren en librerías externas:

- **Inventario de Dependencias:** Todo proyecto debe listar las librerías de terceros utilizadas. Estas deben provenir de fuentes oficiales y ser verificadas en busca de vulnerabilidades conocidas (CVEs).
- **Actualización Constante:** Es responsabilidad de los desarrolladores mantener las dependencias actualizadas a la última versión estable de seguridad disponible.

3.6. Control de Versiones y Repositorios

- **Uso de Repositorios:** Todo código fuente debe residir en un repositorio centralizado (ej. Git) con control de acceso nominal.
- **Protección de Secretos:** Queda terminantemente prohibido incluir contraseñas, llaves de API o secretos de conexión dentro del código fuente (Hardcoding). Estos deben gestionarse a través de variables de entorno o bóvedas de secretos seguras.

3.7. Verificación Final de Requisitos

Antes de la liberación, se debe completar el **Formato Verificación de Requisitos de Desarrollo Seguro**, el cual actúa como una lista de control (checklist) para asegurar que nada de lo declarado en la BRD fue omitido durante la programación.

4. Procedimientos

4.1. Definición de Requisitos de Seguridad

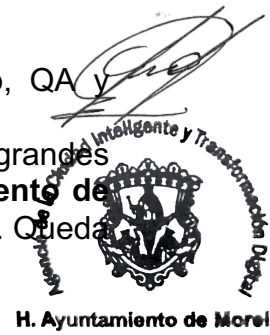
Este es el primer paso para cualquier proyecto nuevo o modificación mayor:

1. **Llenado de la BRD:** El líder del proyecto debe completar la **Bitácora de Requerimientos de Desarrollo (BRD)**, especificando:
   - ¿Qué datos personales se manejarán? (Nombre, RFC, CURP).
   - ¿Quiénes tendrán acceso y con qué roles?
   - ¿Qué nivel de cifrado se requiere para los datos en reposo y tránsito?
2. **Validación de Seguridad:** El Oficial de Seguridad de la Información revisa la BRD para asegurar que no se omitieron controles críticos (ej. registros de auditoría o validación de entradas).

4.2. Preparación del Entorno de Desarrollo

Para proteger el código y los datos institucionales:

1. **Aislamiento de Ambientes:** Se configuran los servidores de Desarrollo, QA y Producción de forma que no tengan comunicación directa entre sí.
2. **Aprovisionamiento de Datos Sintéticos:** Si se requiere probar con grandes volúmenes de datos, se debe ejecutar el procedimiento de **Enmascaramiento de Datos** sobre una copia de producción para alimentar el entorno de pruebas. Queda prohibido el uso de credenciales de producción en el código de desarrollo.



4.3. Ejecución de Pruebas de Seguridad (SAST y DAST)

Una vez que el código es funcional, debe pasar por las "pruebas de fuego":

1. **Análisis Estático (SAST):** Se escanea el código fuente (sin ejecutar la app) para detectar malas prácticas, como contraseñas en texto plano o funciones vulnerables a inyecciones. Se documentan los hallazgos en el **Formato Pruebas de Seguridad SAST / DAST**.
2. **Análisis Dinámico (DAST):** Se ejecuta la aplicación en el entorno de Staging y se simulan ataques externos (escaneo de puertos, pruebas de denegación de servicio, intentos de bypass de login). Los resultados se integran en el mismo formato.
3. **Remediación:** Si se encuentran vulnerabilidades de nivel "Crítico" o "Alto", el desarrollo regresa a la fase de codificación para su corrección obligatoria.

4.4. Verificación y Validación Final

Antes de considerar el software como "terminado":

1. **Checklist de Seguridad:** Se utiliza el **Formato Verificación de Requisitos de Desarrollo Seguro** para cruzar los requerimientos iniciales de la BRD contra lo implementado.
2. **Revisión de Pares:** (Opcional pero recomendado) Un desarrollador distinto al autor del código realiza una revisión rápida para identificar errores de lógica que las herramientas automáticas podrían omitir.

4.5. Paso a Producción y Liberación

1. **Firma de Liberación:** Una vez que todas las pruebas son satisfactorias y los requisitos están verificados, se genera el **Acta de Liberación Segura a Producción (Go/No-Go)**.
2. **Despliegue Controlado:** El personal de Operaciones/DevOps realiza el despliegue en producción, preferentemente mediante procesos automatizados. Se debe verificar que las configuraciones de seguridad (certificados SSL/TLS, permisos de carpetas) se activen correctamente en el servidor real.

4.6. Mantenimiento y Monitoreo Post-Lanzamiento

1. **Monitoreo de Vulnerabilidades:** Se establece un calendario semestral para repetir las pruebas DAST sobre las aplicaciones activas, asegurando que nuevas vulnerabilidades descubiertas en el mercado no afecten al Ayuntamiento.
2. **Gestión de Parches de Librerías:** Mensualmente se revisará si las librerías de terceros (Open Source) utilizadas en el desarrollo tienen actualizaciones de seguridad disponibles.
3. **Gestión de Código Legacy:** El Ayuntamiento deberá procurar a través de la Agencia de la Ciudad Inteligente y Transformación Digital incluir en sus Planes Anuales de Trabajo la revisión y actualización o modernización de al menos un sistema con Código Legado para evitar arrastrar deuda técnica.

Agencia de la Ciudad Inteligente y Transformación Digital
H. Ayuntamiento de Morelia

5. Definiciones

- **BRD (Bitácora de Requerimientos de Desarrollo):** Documento maestro donde se capturan tanto las necesidades funcionales como las restricciones de seguridad que el software debe cumplir.
- **DAST (Dynamic Application Security Testing):** Pruebas de seguridad dinámicas que se realizan mientras la aplicación está en ejecución, simulando ataques externos para encontrar vulnerabilidades en tiempo real.
- **SAST (Static Application Security Testing):** Análisis estático del código fuente (sin ejecutar la aplicación) para identificar debilidades lógicas, malas prácticas de programación y posibles brechas de seguridad.
- **OWASP (Open Web Application Security Project):** Fundación sin fines de lucro que proporciona estándares y guías de seguridad para el desarrollo de software, siendo su "Top 10" la referencia mundial para mitigar riesgos web.
- **Inyección de Código (SQLi, XSS):** Tipo de ataque donde un usuario malintencionado inserta comandos en los formularios de la aplicación para extraer datos o ejecutar acciones no autorizadas.
- **S-SDLC (Secure Software Development Life Cycle):** Metodología que integra controles de seguridad en cada una de las fases del desarrollo de sistemas (Planeación, Diseño, Codificación, Pruebas y Despliegue).
- **Secretos (Secrets):** Información sensible como contraseñas de bases de datos, llaves de API o certificados que nunca deben estar "quemados" (hardcoded) en el código fuente.

6. Formatos

Para que el proceso sea auditable ante una certificación **ISO 27001**, se deben generar y custodiar los siguientes registros:

1. **F-DEV-01: Bitácora de Requerimientos de Desarrollo (BRD)**
   - *Función:* El punto de partida que dicta qué controles de seguridad llevará el sistema.
2. **F-DEV-02: Reporte de Pruebas de Seguridad SAST / DAST**
   - *Función:* Evidencia técnica de los escaneos realizados y el estado de las vulnerabilidades encontradas y corregidas.
3. **F-DEV-03: Formato de Verificación de Requisitos de Desarrollo Seguro**
   - *Función:* Checklist final que asegura que lo que se pidió en la BRD realmente se programó.

4. **F-DEV-04: Matriz de Trazabilidad de Requisitos de Seguridad (Sugerido)**
   - *Función:* Relaciona cada requisito de seguridad con su prueba correspondiente para no dejar cabos sueltos.
5. **F-DEV-05: Acta de Liberación Segura a Producción**
   - *Función:* El "Visto Bueno" final que autoriza el despliegue del software en el servidor oficial del Ayuntamiento.

7. Relación Normativa (ISO 27001:2022)

Este documento garantiza que el Ayuntamiento cumpla con creces los controles más exigentes de la norma:

Control	Título	Justificación del Cumplimiento
8.25	**Ciclo de vida de desarrollo seguro**	Establece el flujo obligatorio de seguridad desde la BRD hasta la liberación.
8.26	**Requisitos de seguridad de la aplicación**	Se materializa mediante el llenado y validación de la **BRD** en cada proyecto.
8.28	**Codificación segura**	Define el uso de frameworks, guías OWASP y la prohibición de secretos en el código.
8.29	**Gestión de la configuración**	Asegura que el entorno de desarrollo y producción estén aislados y documentados.
8.31	**Separación de entornos de desarrollo, pruebas y producción**	Política obligatoria de aislamiento físico y lógico de los servidores.