ESTÁNDAR DE DESARROLLO SEGURO DE APLICACIONES

Ministerio de Coordinación

Revisión 01 - 220521

Guía de controles y aspectos a cumplimentar para el desarrollo de aplicaciones seguras a implementar en los ambientes tecnológicos del Gobierno de la Provincia de Córdoba

Contenido

1 Objetivo

Definir un estándar para la seguridad de la información con tecnologías de información que proteja la confidencialidad, integridad y disponibilidad de la información perteneciente tanto al Gobierno de la Provincia de Córdoba como a los ciudadanos que acceden a sus sistemas de información.

2 Alcance

El presente estándar debe ser aplicado al desarrollo del software y la infraestructura a ser utilizada por todas las unidades organizacionales que componen el Gobierno de la Provincia de Córdoba.

3 Consideraciones Generales

Este estándar define una colección de requerimientos de seguridad relacionados con las siguientes actividades inherentes al desarrollo de software: gestión de configuración, análisis de requerimientos y diseño, codificación, pruebas, despliegue, monitoreo y control, gestión de datos.

3.1 Marco conceptual para la aplicación del estándar

Para la aplicación de este estándar se establece un marco conceptual [1] para la aplicación de los requerimientos de seguridad en el software basado en tres niveles de verificación en función de la profundidad de los controles. Cada nivel está constituido por un grupo de requerimientos que deben ser cumplidos.

	Nivel 1	**Nivel 2**	**Nivel 3**
Aplicabilidad	Todas las aplicaciones	Todas las aplicaciones	Alto Aseguramiento
Desarrollo		Arquitectura de Seguridad y Revisiones	Arquitectura de Seguridad y Revisiones
	Código Seguro	Código Seguro	Código Seguro
	Estándares y Listas de Verificación	Estándares y Listas de Verificación	Estándares y Listas de Verificación

[1] Este estándar toma como base algunos de los requerimientos establecidos en el proyecto *Application Security Verification Standard (ASVS)* de *OWASP*.

Desarrollo, Configuración, Despliegue, Aseguramiento y Verificación	Revisiones de Pares y de Seguridad	Revisiones de Pares y de Seguridad	Revisiones de Pares y de Seguridad
	DevSecOps	DevSecOps	DevSecOps
	Pruebas de Unidad y de Integración	Pruebas de Unidad y de Integración	Pruebas de Unidad y de Integración
Aseguramiento y Verificación	Pentest	Revisiones Híbridas	Revisiones Híbridas
	DAST	SAST	SAST

Código de colores

Aceptable	**Adecuado**

Figura 1 - Niveles de requerimientos de seguridad del estándar.

3.1.1 Requerimientos de seguridad de nivel 1

Alcanza a todo el software y es el nivel mínimo al cual todas las aplicaciones deben aspirar. Es un nivel aceptable para aplicaciones que no manejan datos sensibles, pero insuficiente para aplicaciones de alta criticidad.

Los controles requeridos en este nivel están relacionados con el aseguramiento mínimo aceptable.

Para cumplir con este nivel, las aplicaciones deben probarse para asegurar que poseen defensas contra vulnerabilidades fáciles de explotar y que están en el Top 10 de OWASP.

3.1.2 Requerimientos de seguridad de nivel 2

Asegura que la aplicación posee defensas contra la mayoría de los riesgos actuales asociados al software.

Asegura que los controles están implementados efectivamente y son utilizados por la aplicación.

Este nivel es el adecuado para aplicaciones que manejan información sensible (por ejemplo, datos de salud) o manejan transacciones de negocio relevantes o interactúan con recursos o procesos críticos; aplicaciones relacionadas con la industria de los juegos también entran en este nivel.

3.1.3 Requerimientos de seguridad de nivel 3

Alcanza a aplicaciones críticas que requieren un nivel alto de seguridad.

Se deberá aplicar este nivel si la aplicación ejecuta funciones críticas; si la falla de la aplicación puede resultar en un impacto significativo para las operaciones de la organización; o si se requiere un alto nivel de confianza sobre la aplicación.

Una aplicación logra este nivel cuando se defiende adecuadamente contra vulnerabilidades avanzadas y demuestra principios de buen diseño de seguridad.

Las aplicaciones en este nivel requieren de análisis de arquitectura, infraestructura y diseño; de revisión de código y de testing durante todo el ciclo de vida de su desarrollo.

Una aplicación segura tiene un diseño modular (para facilitar la resiliencia, la escalabilidad y la seguridad en capas), Cada módulo tiene responsabilidades sobre su seguridad (defensa en profundidad) que deben ser documentadas apropiadamente. Estas responsabilidades incluyen controles para asegurar la confidencialidad (cifrado, por ejemplo), integridad (transacciones, validación de entradas), disponibilidad (balanceo de cargas), autenticación (incluso entre sistemas), no repudio, autorización y auditoría (logs). Adicionalmente, puede contar con funcionalidades adicionales de seguridad como WAF.

Ejemplos de aplicabilidad: aplicaciones utilizadas en infraestructura, medicina, seguridad u operaciones militares; en dispositivos y vehículos automatizados, en tecnología operacional (energía, agua, plantas químicas, nucleares, etc.); software de control; aplicaciones financieras.

3.2 Proceso de aplicación del estándar

La aplicación de este estándar comprende las siguientes tres etapas de ejecución:

1. Elección del nivel de seguridad requerido para la aplicación.
2. Implementación de los controles de seguridad.
3. Auditar la conformidad con los requerimientos de seguridad.

3.2.1 Elección del nivel de seguridad requerido para la aplicación

En cada proyecto de software se deberá llevar adelante un análisis de las características de riesgo de acuerdo a la naturaleza de las actividades a las que el software dará soporte, y con base en dicho riesgo y los requerimientos organizacionales, determinar el nivel de seguridad apropiado. Para ello, se deben tener en cuenta las siguientes consideraciones:

- existen diferentes amenazas con diferentes motivaciones,
- puede existir información y recursos únicos,
- pueden existir requerimientos regulatorios para dominios específicos.

3.2.2 Implementación de los controles de seguridad

Luego de haberse elegido el nivel de seguridad al que la aplicación deberá adaptarse, se deberán incorporar al desarrollo la implementación de los controles necesarios para cumplir con los requerimientos definidos por el estándar para ese nivel de seguridad (ver sección Requerimientos de seguridad por nivel adoptado).

3.2.3 Auditar la conformidad con los requerimientos de seguridad

Dependiendo del nivel requerido de seguridad, se podrá definir una o más acciones para determinar si los requerimientos se cumplen.

3.2.3.1 Validación de la conformidad con los requerimientos de nivel 1

Para validar que los requerimientos de seguridad de nivel 1 se cumplen, se debe llevar a cabo como mínimo pentest de caja negra.

3.2.3.2 Validación de la conformidad con los requerimientos de nivel 2

Para validar que los requerimientos de seguridad de nivel 2 se cumplen, se deben llevar a cabo como mínimo las siguientes acciones:

- Pentest de caja negra.
- Pentest de caja blanca: revisión de código. Acceso completo al código y a documentación de todo el proceso de desarrollo.
- Implementación de herramientas automatizadas de tipo DAST y SAST durante el proceso de desarrollo, para encontrar problemas de seguridad continuamente.

3.2.3.3 Validación de la conformidad con los requerimientos de nivel 3

Para validar que los requerimientos de seguridad de nivel 3 se cumplen, se deben llevar a cabo como mínimo las siguientes acciones:

- Pentest de caja negra.
- Pentest de caja blanca: revisión de código. Acceso completo al código y a documentación de todo el proceso de desarrollo.
- Implementación de herramientas automatizadas de tipo DAST y SAST durante el proceso de desarrollo, para encontrar problemas de seguridad continuamente.
- Análisis en profundidad de la arquitectura, codificación, testing, infraestructura.

4 Condiciones Particulares

4.A - Codificación: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con la codificación del software a ser implementados por el proveedor del mismo.

4.A.1 Validación de entradas y salidas

El objetivo de la validación de entradas es asegurar que solo datos con el formato apropiado ingresan al workflow del sistema, de forma tal de prevenir que datos mal formados persistan en la base de datos y provoquen un funcionamiento incorrecto del sistema.

La validación de entradas debe ocurrir lo más temprano posible en el flujo de datos, tan pronto como se recibe la información de la contraparte externa. Esto permite detectar entradas no autorizadas antes de que sean procesadas por la aplicación.

A pesar de que los mecanismos de validación de entradas no deberían utilizarse como el principal método de prevención de ataques (como XSS, inyección SQL y otros), pueden reducir significativamente su impacto si se implementan correctamente. Es importante que estos mecanismos se implementen de manera proactiva.

4.A.1.1 Requerimientos de validación de entradas

ID	Descripción	Nivel			CWE
		1	2	3	
C1	La aplicación debe poseer defensas contra ataques de polución de parámetros HTTP (en métodos GET y POST, cookies, encabezados y variables de entorno).	✓	✓	✓	235
C2	Se debe verificar que las librerías, módulos o cualquier stack tecnológico del cual el código dependa protegen contra ataques de asignación masiva de parámetros (ej.: determinando campos o variables privados).	✓	✓	✓	915
C3	Todas las entradas (campos de formulario HTML, peticiones REST, parámetros de URL, cabeceras HTTP, cookies, archivos batch, feeds RSS, etc.) deben ser validadas utilizando validación positiva (lista blanca / allow-list).	✓	✓	✓	20
C4	Los datos estructurados deben ser fuertemente tipados, validados teniendo en cuenta caracteres, longitud y patrones (ej.: números de teléfono o tarjetas de crédito) o validando que lo ingresado en campos relacionados sea razonable (ej.: que el código postal y el barrio coincidan).	✓	✓	✓	20
C5	Las redirecciones URL solo deben llevar a destinos que están determinados en una lista blanca o mostrar una advertencia cuando se redirige a contenido potencialmente inseguro.	✓	✓	✓	601

4.A.1.2 Requerimientos de saneamiento y sandboxing

ID	Descripción	Nivel			CWE
		1	2	3	

C6	Toda entrada de HTML proveniente de editores debe ser saneada apropiadamente mediante librerías de sanitización de HTML o cualquier otro stack tecnológico.	✓	✓	✓	116
C7	Los datos no estructurados deben ser saneados para reforzar las medidas de seguridad (ej.: caracteres permitidos y longitud).	✓	✓	✓	138
C8	La entrada del usuario debe ser saneada antes de pasarla a sistemas de correo electrónico para proteger contra inyecciones SMTP o IMAP.	✓	✓	✓	147
C9	No se debe usar la función eval() o cualquier otra característica que permita ejecución de código dinámica.	✓	✓	✓	95
C10	La aplicación debe proteger contra ataques de template injection asegurando que cualquier entrada del usuario a ser incluida es saneada o su procesamiento aislado (sandboxed).	✓	✓	✓	94
C11	La aplicación debe proteger contra ataques SSRF, validando o saneando datos no confiables o metadatos de archivos (ej.: nombres de archivos y campos de entrada en URL), y utiliza listas blancas de protocolos, dominios, rutas y puertos.	✓	✓	✓	918
C12	Se debe sanear, deshabilitar, o aislar (sandboxing) el contenido provisto mediante SVG por el riesgo de XSS resultante de scripts inline y foreignObject.	✓	✓	✓	159
C13	La aplicación debe quitar o deshabilitar contenido de tipo script.	✓	✓	✓	94

4.A.1.3 Requerimientos de codificación de salidas y prevención de inyección

ID	Descripción	Nivel			CWE
		1	2	3	
C14	La codificación de salidas debe ser relevante para el intérprete y el contexto requerido (ej.: utilizar codificadores especialmente en valores HTML, atributos HTML, JavaScript, parámetros URL, cabeceras HTTP, SMTP) principalmente para entradas no confiables.	✓	✓	✓	116
C15	La codificación de salidas debe preservar los caracteres elegidos por el usuario, de forma que cualquier caracter Unicode es válido y manejado de forma segura.	✓	✓	✓	176
C16	La codificación de salidas debe protege contra XSS reflejado, almacenado y basado en DOM.	✓	✓	✓	79
C17	La selección de datos o consultas de base de datos debe hacerse mediante consultas parametrizadas, ORM, entity frameworks o cualquier otra forma que proteja contra ataques de inyección en base de datos.	✓	✓	✓	89

C18	Donde no haya mecanismos seguros o parametrizados, se debe utilizar codificación de salidas específicas del contexto para proteger contra ataques de inyección (ej.: utilización de escape de SQL para proteger contra inyección SQL).	✓	✓	✓	89
C19	La aplicación debe proteger contra ataques de inyección JSON o JavaScript (incluyendo ataques eval, inclusiones de JavaScript remotas, bypasses de CSP, DOM XSS, y evaluación de expresiones JavaScript).	✓	✓	✓	830
C20	La aplicación debe proteger contra vulnerabilidades de inyección LDAP o se deben implementar controles de seguridad específicos para prevenir esta inyección.	✓	✓	✓	90
C21	La aplicación debe proteger contra inyección de comandos de sistema operativo y que las llamadas al sistema utilizan consultas de sistema operativo o codificación de la salida de la línea de comandos.	✓	✓	✓	78
C22	La aplicación debe proteger contra ataques Local File Inclusion (LFI) o Remote File Inclusion (RFI).	✓	✓	✓	829
C23	La aplicación debe proteger contra inyección XPath o ataques de inyección XML.	✓	✓	✓	643

4.A.1.4 Requerimientos de memoria, cadenas y código no administrado

Estos requerimientos solo se aplicarán cuando la aplicación utilice un lenguaje de sistemas o código no administrado.

ID	Descripción	Nivel			CWE
		1	2	3	
C24	La aplicación debe utilizar cadenas seguras para la memoria, una copia de memoria más segura y aritmética de punteros para detectar o evitar desbordamientos de pila o búfer.		✓	✓	120
C25	Las cadenas de formato no deben recibir entradas potencialmente hostiles y deben ser constantes.		✓	✓	134
C26	Se deben utilizar técnicas de validación de signo, rango y entrada para evitar desbordamientos de enteros.		✓	✓	190

4.A.1.5 Requerimientos de prevención de deserialización

ID	Descripción	Nivel			CWE
		1	2	3	

C27	Se deben utilizar chequeos de integridad sobre los objetos serializados o los mismos deben estar cifrados para prevenir creación de objetos hostiles o manipulación de datos.	✓	✓	✓	502
C28	La aplicación debe restringir los parsers XML a la configuración más restrictiva posible y deshabilitar características inseguras como resolución de entidades externas para prevenir ataques XXE.	✓	✓	✓	611
C29	La deserialización de datos no confiables debe ser evitada o protegida tanto en el código de desarrollo propio o de librerías de terceros (ej.: parsers JSON, XML o YAML).	✓	✓	✓	502
C30	Se debe utilizar la función JSON.parse cuando se parsea JSON en navegadores o backends basados en JavaScript. No se debe usar eval() para parsear JSON.	✓	✓	✓	95

4.A.2 Archivos y Recursos

Respecto de la gestión de archivos y recursos, se debe asegurar que el software satisfaga los siguientes requerimientos de alto nivel:

- La información de archivos debe ser manejada apropiadamente y de manera segura.
- Para los casos de aplicaciones web, la información de archivos obtenida de fuentes no confiables debe ser almacenada fuera de la ruta raíz del sitio y con permisos limitados.

4.A.2.1 Requerimientos relacionados con la subida de archivos

ID	Descripción	Nivel			CWE
		1	2	3	
C31	La aplicación no debe aceptar archivos de gran tamaño que puedan llenar el almacenamiento y causar denegación de servicio.	✓	✓	✓	400
C32	Los archivos comprimidos deben ser chequeados contra zip bombs.		✓	✓	409
C33	La cuota de tamaño y cantidad máxima de archivos por usuario configurada debe asegurar que un único usuario no pueda llenar el almacenamiento con demasiados archivos o archivos excesivamente grandes.		✓	✓	70

4.A.2.2 Requerimientos de integridad de los archivos

ID	Descripción	Nivel			CWE
		1	2	3	

C34	Los archivos obtenidos de fuentes no confiables deben ser validados de manera tal que posean el tipo esperado basado en el contenido del archivo.		✓	✓	434

4.A.2.3 Requerimientos relacionados con la ejecución de archivos

ID	Descripción	Nivel			CWE
		1	2	3	
C35	El nombre del archivo provisto por el usuario no debe ser utilizado directamente por el sistema o cualquier stack tecnológico utilizado y que la API utilizada protege contra path traversal.	✓	✓	✓	22
C36	Loa nombres de los archivos provistos por el usuario deben ser validados o ignorados para prevenir la divulgación, creación, actualización o borrado de archivos locales (LFI).	✓	✓	✓	73
C37	El nombre del archivo provisto por el usuario debe ser validado o ignorado para prevenir la divulgación o ejecución de archivos remotos vía RFI o SSRF.	✓	✓	✓	98
C38	La aplicación debe proteger contra RFD validando o ignorando el nombre del archivo provisto por el usuario; la cabecera de respuesta Content-Type debería configurarse a text/plain y la cabecera Content-Disposition debería tener un nombre de archivo fijo.	✓	✓	✓	641
C39	Los metadatos del archivo no deben ser utilizados directamente por las librerías o API, para proteger contra inyección de comandos del sistema operativo.	✓	✓	✓	78
C40	La aplicación no debe incluir ni ejecutar funcionalidad proveniente de fuentes no confiables, como redes de distribución de contenido no verificadas, librerías JavaScript, librerías node o DLL.		✓	✓	829

4.A.2.4 Requerimientos relacionados con el almacenamiento de archivos

ID	Descripción	Nivel			CWE
		1	2	3	
C41	Los archivos obtenidos de fuentes no confiables deben ser almacenados fuera de la raíz del sitio web, con permisos limitados.	✓	✓	✓	922

C42	Los archivos obtenidos de fuentes no confiables deben ser escaneados mediante antivirus para prevenir la subida de contenido malicioso.	✓	✓	✓	509

4.A.2.5 Requerimientos relacionados con la descarga de archivos

ID	Descripción	Nivel			CWE
		1	2	3	
C43	La capa web debe estar configurada para servir archivos que posean extensiones específicas, para prevenir divulgación de información. Por ejemplo, debería bloquearse la bajada de archivos de backup, temporales, comprimidos y otras extensiones.	✓	✓	✓	522
C44	Las peticiones directas a archivos subidos nunca deben ser ejecutadas como contenido HTML/JavaScript	✓	✓	✓	434

4.A.2.6 Requerimientos de protección contra SSRF

ID	Descripción	Nivel			CWE
		1	2	3	
C45	La web o el servidor de aplicaciones debe estar configurado con una lista blanca de recursos o sistemas a los cuales el servidor puede enviar peticiones o cargar archivos/información desde allí.	✓	✓	✓	918

4.A.3 Manejo de errores y auditoría de logs

La gestión de errores es parte de la seguridad general de las aplicaciones.

Los errores no manejados correctamente facilitan la etapa de reconocimiento para el atacante, ya que se puede revelar bastante información acerca del objetivo y pueden utilizarse para identificar puntos de inyección en las funcionalidades.

El objetivo del manejo de errores y logging es proveer información útil para el usuario, administradores y equipos de respuesta a incidentes.

El objetivo de este estándar es la creación de logs de calidad.

A menudo, los logs poseen datos sensibles que deberían ser protegidos mediante directivas. Esto incluye:

- No registrar información sensible en logs a menos que sea específicamente requerido.

- Asegurar que toda la información registrada en logs se gestiona de manera segura y es protegida de acuerdo a su clasificación de datos.
- Asegurarse de que los logs no se almacenan para siempre, sino que tienen un tiempo de vida lo más corto posible.

4.A.3.1 Requerimientos acerca del contenido de logs

ID	Descripción	Nivel			CWE
		1	2	3	
C46	No se deben registrar en logs credenciales o detalles de pago. En caso de tokens de sesión, se deberán registrar en forma de hash.	✓	✓	✓	532
C47	No se deben registrar en logs datos sensibles definidos por leyes de protección de datos o por la política de seguridad.	✓	✓	✓	532
C48	Se deben registrar en logs eventos relevantes de seguridad (incluyendo: eventos de autenticación exitosa y fallida, controles de acceso fallidos, validaciones de entrada fallidas).		✓	✓	778

4.A.3.2 Requerimientos de protección de logs

ID	Descripción	Nivel			CWE
		1	2	3	
C49	La aplicación debe codificar los datos provistos por el usuario para prevenir inyección de logs.		✓	✓	117
C50	Los logs de seguridad deben estar protegidos contra accesos y modificaciones no autorizadas.		✓	✓	200

4.A.3.3 Requerimientos de manejo de errores

ID	Descripción	Nivel			CWE
		1	2	3	
C51	Se debe mostrar un mensaje genérico cuando un error sensible o inesperado ocurre, potencialmente con un ID único que el personal de soporte pueda utilizar para investigar.	✓	✓	✓	210
C52	Se debe utilizar el manejo de excepciones en todo el código teniendo en cuenta las condiciones de error esperadas e inesperadas.		✓	✓	544

C53	Se debe definir un controlador de errores de último recurso, que detectará todas las excepciones no controladas.		✓	✓	431

4.A.4 Verificación de código malicioso

4.A.4.1 Controles de integridad de código

ID	Descripción	Nivel			CWE
		1	2	3	
C54	Se deben usar herramientas de análisis de código que puedan detectar código potencialmente malicioso.			✓	749

4.A.4.2 Búsqueda de código malicioso

ID	Descripción	Nivel			CWE
		1	2	3	
C55	El código fuente y las librerías de terceros no deben contener funciones de recopilación de datos.		✓	✓	359
C56	Se debe hacer un análisis minucioso acerca de las funcionalidades y permisos relacionados con la privacidad del usuario (ej.: acceso a contactos, cámara, micrófono, ubicación) a los que la aplicación tendrá acceso, de manera que sean estrictamente los suficientes para cumplir con el alcance definido para la misma y no resulten excesivos.		✓	✓	272
C57	El código fuente y las librerías utilizadas no deben contener back doors (como claves o cuentas no documentadas, código ofuscado, rootkits, anti-debugging, o funcionalidades obsoletas escondidas que podrían utilizarse de forma maliciosa).			✓	507

4.A.6 Criptografía

4.A.6.1 Algoritmos

ID	Descripción	Nivel			CWE
		1	2	3	
C58	Las fallas de todos los módulos criptográficos deben ocurrir de manera segura y los errores deben ser manejados apropiadamente.	✓	✓	✓	310

C59	Se deben utilizar los algoritmos criptográficos y librerías aprobadas por la industria, en lugar de criptografía creada por el equipo.		✓	✓	327
C60	El vector de inicialización, el cifrado y los modos de cifrado de bloque deben estar configurados de acuerdo a las últimas recomendaciones.		✓	✓	326
C61	Los algoritmos de generación de números aleatorios, cifrado y hashing puedan ser reconfigurados, actualizados y cambiados en cualquier momento para proteger contra fallas criptográficas.		✓	✓	326
C62	No se deben utilizar mecanismos inseguros y algoritmos débiles de hashing (ej.: ECB, PKCS#1 v1.5, Triple-DES, Blowfish, MD5, SHA1), salvo que sean requeridos por retrocompatibilidad.		✓	✓	326
C63	Los nonces, vectores de inicialización y otros números de único uso no pueden utilizarse más de una vez dada una clave de cifrado.		✓	✓	326
C64	Los datos cifrados deben ser autenticados vía firmas, modos de cifrado autenticado o HMAC, para asegurar que el texto cifrado no es alterado por partes no autorizadas.			✓	326

4.A.6.2 Valores aleatorios

ID	Descripción	Nivel			CWE
		1	2	3	
C65	Todos los números aleatorios, nombres de archivo aleatorios, GUID aleatorios y cadenas aleatorias deben ser generadas utilizando generadores de números aleatorios de módulos de criptografía aprobados como seguros, cuando lo que se pretende es que los valores no sean adivinables.		✓	✓	338
C66	Verificar que los GUID aleatorios son creados utilizando la versión 4 del algoritmo y un generador de números pseudo aleatorios criptográficamente seguro.		✓	✓	338
C67	Verificar que los números aleatorios son creados con entropía apropiada.			✓	338

4.A.6.3 Gestión de secretos

ID	Descripción	Nivel			CWE
		1	2	3	
C68	Se debe utilizar una solución de gestión de secretos como un baúl de claves para crear, almacenar, controlar accesos y destruir secretos.	✓	✓	✓	798

C69	Las claves no deben ser expuestas a la aplicación, sino que se debe utilizar un módulo aislado como un baúl de claves para las operaciones criptográficas.	✓	✓	✓	320

4.A.7 API y web services

4.7.1 Requerimientos genéricos de web services

ID	Descripción	Nivel			CWE
		1	2	3	
C70	Todos los componentes deben utilizar los mismos codificadores y parsers.	✓	✓	✓	116
C71	Los accesos a funciones de administración y gestión deben ser limitados a administradores autorizados.	✓	✓	✓	419
C72	Las URL de las API no deben exponer información sensible como tokens de sesión o API keys.	✓	✓	✓	598
C73	Las decisiones de autorización se deben realizar tanto en la URI y el controlador como a nivel de recursos.		✓	✓	285
C74	Las peticiones que posean tipos de contenido inesperado o faltante deben ser rechazadas con cabeceras apropiadas (status 406 - 415).		✓	✓	434

4.A.7.2 Requerimientos de RESTful web services

ID	Descripción	Nivel			CWE
		1	2	3	
C75	Los métodos HTTP habilitados deben ser válidos para la acción o el usuario.	✓	✓	✓	650
C76	Se debe validar el esquema JSON antes de aceptar las entradas.	✓	✓	✓	20
C77	Los servicios web que utilizan cookies deben estar protegidos contra CSRF mediante el uso de al menos uno de los siguientes mecanismos: CSRF nonces o chequeos de origen de la request.	✓	✓	✓	352
C78	Los servicios REST deben poseer controles contra llamadas excesivas, específicamente si la API está autenticada.	✓	✓	✓	352
C79	Los servicios REST deben validar que el valor de Content-Type sea el esperado (application/xml o application/json).		✓	✓	436
C80	Las cabeceras y el cuerpo del mensaje deben ser confiables y no modificados en tránsito. Utilizar TLS puede ser suficiente en muchos		✓	✓	345

	casos, ya que provee confidencialidad e integridad. Firmas digitales por mensaje pueden brindar seguridad adicional, aunque agregan complejidad.				

4.A.7.3 Requerimientos de SOAP web services

ID	Descripción	Nivel			CWE
		1	2	3	
C81	Antes del procesamiento se debe validar el esquema XSD, que el documento XML esté formado apropiadamente y cada campo de entrada.	✓	✓	✓	20
C82	La payload debe ser firmada utilizando WS-Security para asegurar transporte confiable entre cliente y servicio.		✓	✓	345
C83	No debe utilizarse validación DTD y la evaluación mediante DTD debe estar deshabilitada.	✓	✓	✓	345

4.A.7.4 Requerimientos para GraphQL u otros servicios web

ID	Descripción	Nivel			CWE
		1	2	3	
C84	Se debe utilizar listas blancas de consultas o una combinación de límites de profundidad y cantidad para prevenir denegaciones de servicio como resultado de consultas anidadas.		✓	✓	770
C85	Se debe implementar una capa de autorización a nivel de capa de lógica de negocios en lugar de en la capa de GraphQL.		✓	✓	285

4.A.8 Autenticación[2]

La autenticación es el acto de establecer o confirmar que alguien (o algo) es auténtico y asegurar que las solicitudes realizadas por una persona o dispositivo son correctas, resistentes a suplantación de identidad y previenen la recuperación o intercepción de contraseñas.

4.A.8.1 Requerimientos de seguridad de contraseñas

El concepto de contraseña[3] para este estándar incluye contraseñas, PIN, patrones de desbloqueo, elementos de imagen a seleccionar y passphrases. Se considera “algo que sabes” y a menudo son utilizados como

[2] Algunos de los requerimientos son tomados a partir de *NIST, “Estándar de autenticación basado en evidencias”, Special Publication 800-63*. Este estándar es moderno y representa las mejores recomendaciones disponibles actualmente, independientemente de su aplicabilidad. Es valioso para las organizaciones en general y particularmente para las gubernamentales.

[3] Llamada secreto memorizado por NIST 800-63.

autenticadores de un solo factor. Las aplicaciones deberían animar a los usuarios a utilizar la autenticación multi-factor.

ID	Descripción	Nivel			CWE
		1	2	3	
C86	El sistema debe validar que el usuario crea contraseñas de al menos 12 caracteres de longitud.	✓	✓	✓	521
C87	Se deben permitir contraseñas de hasta 128 caracteres como límite máximo.	✓	✓	✓	285
C88	No debe ocurrir truncamiento de contraseñas.	✓	✓	✓	521
C89	Se debe permitir como parte de una contraseña cualquier caracter Unicode (incluyendo espacios y emojis).	✓	✓	✓	521
C90	Los usuarios deben poder cambiar sus contraseñas.	✓	✓	✓	620
C91	La funcionalidad de cambio de contraseña debe requerir la contraseña actual y la nueva.	✓	✓	✓	620
C92	Las contraseñas creadas durante el registro, inicio de sesión y cambio de contraseña deben ser chequeadas contra un conjunto de contraseñas comprometidas en brechas de seguridad, ya sea de manera local (ej.: contra el top 1000 de contraseñas comunes) o contra una API externa).		✓	✓	620
C93	Se debe proveer al usuario de un medidor visual de robustez de la contraseña elegida.	✓	✓	✓	521
C94	No deben existir requerimientos de rotación periódica o historial de contraseñas.	✓	✓	✓	263
C95	La funcionalidad de “pegar” y los gestores de contraseña externos deben ser permitidos.	✓	✓	✓	521
C96	El usuario debe poder elegir visualizar temporalmente la contraseña sin máscara o el último caracter tipeado, al momento de ingresar la misma.	✓	✓	✓	521

4.A.8.2 Requerimientos generales de autenticadores[4]

ID	Descripción	Nivel			CWE
		1	2	3	

[4] Tener en cuenta que NIST considera tanto a los *emails* como a los SMS como autenticadores “restringidos” y es probable que sean quitados del estándar en el futuro. Las aplicaciones deberían planificar una hoja de ruta para la eliminación del uso del correo electrónico o el SMS y hacer la transición hacia otro tipo de autenticadores.

C97	Deben existir controles contra fuerza bruta y ataques de bloqueo de cuentas (ej.: bloqueos “soft”, límites de intentos, CAPTCHA, aumento del retardo entre intentos, restricciones por IP o restricciones basadas en ubicación, primer inicio de sesión en un dispositivo, intentos recientes para desbloquear la cuenta, o similares). No se deben permitir más de cien intentos fallidos por hora en una única cuenta.	✓	✓	✓	307
C98	El uso de autenticadores débiles (como SMS o email) debe estar limitado a verificación secundaria y aprobación de transacciones y no como reemplazo de métodos más seguros.	✓	✓	✓	305
C99	Se deben enviar notificaciones a los usuarios luego de actualizaciones de detalles de autenticación (ej.: reseteo de credenciales, cambios de dirección, correo, o inicios de sesión desde ubicaciones desconocidas o riesgosas). Se prefiere el uso de notificaciones push, pero en su ausencia, es aceptable utilizar SMS o email (mientras no se divulgue información sensible en ellos).	✓	✓	✓	620
C100	Debe existir resistencia a la suplantación de identidad contra el phishing mediante el uso de autenticación multi-factor, dispositivos criptográficos, o certificados de cliente.			✓	308
C101	Cuando un proveedor de servicios de credenciales (CSP) y la aplicación que verifica la autenticación estén separados, debe haber TLS mutuamente autenticado entre los dos extremos.			✓	319
C102	La aplicación debe presentar resistencia a replay mediante el uso de dispositivos One-Time Passwords (OTP), autenticadores criptográficos o códigos lookup.			✓	308
C103	La aplicación debe verificar la intención de autenticarse solicitando la entrada de un token OTP o una acción iniciada por el usuario (como presionar un botón en una llave de hardware FIDO).			✓	308

4.A.8.3 Requerimientos del ciclo de vida del autenticador

ID	Descripción	Nivel			CWE
		1	2	3	
C104	Las contraseñas iniciales generadas por el sistema o los códigos de activación deben ser generados aleatoriamente de manera segura. Deben ser de al menos 6 caracteres de longitud y pueden contener letras y números. Deben expirar luego de un período corto de tiempo. Estos secretos iniciales no deben ser admitidos como contraseñas a largo plazo.	✓	✓	✓	330
C105	Se debe dar soporte a enrolamiento y al uso de dispositivos de autenticación (como tokens U2F o FIDO).		✓	✓	308

C106	Las instrucciones de renovación de autenticadores con límite de tiempo deben ser enviadas con suficiente antelación.		✓	✓	287

4.A.8.4 Requerimientos de almacenamiento de credenciales

ID	Descripción	Nivel			CWE
		1	2	3	
C107	Las contraseñas deben ser almacenadas de una manera resistente a ataques offline y se DEBE utilizar salt y hash utilizando funciones de hashing o derivación de claves one-way aprobadas.		✓	✓	916
C108	El salt debe ser de al menos 32 bits de longitud y elegido arbitrariamente para minimizar colisiones de salt entre los hashes almacenados. Para cada credencial, se debería almacenar solamente un único salt y el hash resultante (la credencial en sí no debe ser almacenada).		✓	✓	916
C109	Se debe utilizar PBKDF2. La cuenta de iteración debe ser tan larga como la performance del servidor de verificación lo permita (típicamente al menos 100000 iteraciones).		✓	✓	916
C110	Si se utiliza bcrypt, el factor de trabajo debe ser tan largo como la performance del servidor de verificación lo permita (típicamente, al menos 13).		✓	✓	916
C111	Se debe realizar una iteración adicional en la función de derivación de clave, utilizando un salt que es secreto y conocido solo para el verificador. Generar el salt mediante un generador de bits aleatorio aprobado y proveer al menos la fortaleza de seguridad mínima especificada en la última revisión de SP 800-131A[5]. El valor salt secreto debe almacenarse separadamente de las contraseñas hasheadas.		✓	✓	916

4.A.8.5 Requerimientos de Recuperación de Credenciales

ID	Descripción	Nivel			CWE
		1	2	3	
C112	La activación inicial generada por el sistema o el secreto de recuperación no debe ser enviado al usuario en texto plano.	✓	✓	✓	640
C113	No deben usarse pistas de contraseña o autenticación basada en conocimiento (“preguntas de seguridad”).	✓	✓	✓	640

[5] NIST, “*Transitioning the Use of Cryptographic Algorithms and Key Lengths*”, Special Publication 800-131A.

C114	La funcionalidad de recuperación de contraseña no debe de ninguna manera revelar la contraseña actual.	✓	✓	✓	640
C115	No deben estar presentes las cuentas por defecto o compartidas (ej.: “root”, “admin”, “sa”).	✓	✓	✓	16
C116	Si un factor de autenticación es cambiado o reemplazado, el usuario debe ser notificado de este evento.	✓	✓	✓	304
C117	Los olvidos de contraseña y otros flujos de recuperación deben utilizar un mecanismo de recuperación seguro (como TOTP u otro soft token) u otro mecanismo offline de recuperación.	✓	✓	✓	640

4.A.8.6 Requerimientos de verificador de secretos lookup

Los secretos lookup son listas pre-generadas de códigos secretos, similares a códigos de recuperación de redes sociales, que son distribuidos de manera segura al usuario. Estos códigos son utilizados una sola vez, y una vez ocurra esto, la lista de secretos se descarta. Este tipo de autenticador se llama “algo que tienes”.

ID	Descripción	Nivel			CWE
		1	2	3	
C118	Los secretos lookup se deben poder utilizar una sola vez.		✓	✓	308
C119	Los secretos deben tener suficiente aleatoriedad (112 bits de entropía como mínimo) o en su defecto, utilizar un único salt aleatorio de 32 bits y hasheado mediante one-way.		✓	✓	330
C120	Los secretos lookup deben ser resistentes a ataques offline (ej.: valores predecibles).		✓	✓	310

4.A.9 Gestión de sesiones

Se debe asegurar que la aplicación satisfaga los siguientes requerimientos de alto nivel:

- Las sesiones deben ser únicas para cada individuo y no deben ser adivinables ni compartidas.
- Las sesiones deben ser invalidadas cuando no se requieren más y expirar durante períodos de inactividad.

4.A.9.1 Requerimientos Fundamentales de gestión de sesiones

ID	Descripción	Nivel			CWE
		1	2	3	
C121	La aplicación nunca debe revelar tokens de sesión en parámetros de URL.	✓	✓	✓	598

4.A.9.2 Requerimientos de vinculación de sesión

ID	Descripción	Nivel			CWE
		1	2	3	
C122	La aplicación debe generar un nuevo token de sesión en la autenticación del usuario.	✓	✓	✓	384
C123	Los tokens de sesión deben poseer al menos 64 bits de entropía.	✓	✓	✓	331
C124	La aplicación solo debe almacenar tokens de sesión en el navegador utilizando métodos seguros como cookies con las banderas correspondientes o session storage de HTML5.	✓	✓	✓	539
C125	El token de sesión debe ser generado utilizando algoritmos criptográficos aprobados.		✓	✓	331

4.A.9.3 Requerimientos de logout y timeout de sesiones

ID	Descripción	Nivel			CWE
		1	2	3	
C126	El logout y la expiración deben invalidar el token de sesión, de manera tal que el botón atrás o cualquier otro mecanismo no pueda continuar en sesión autenticada.	✓	✓	✓	613
C127	Si los autenticadores permiten a los usuarios el permanecer logueados, se debe autenticar nuevamente de forma periódica cuando la sesión es utilizada activamente o luego de un cierto período de tiempo de inactividad.	30 días.	12 h o 30 m de inactividad, 2FA opcional.	12 h o 15 m de inactividad con 2FA.	613
C128	La aplicación debe terminar cualquier otra sesión activa luego de un cambio exitoso de contraseña. Esto debe tener efecto a lo largo de toda la aplicación y de cualquier relying party.		✓	✓	613
C129	Los usuarios deben poder cerrar cualquier sesión concurrente activa en cualquier dispositivo.		✓	✓	613

4.A.9.4 Requerimientos de sesión basada en cookies

ID	Descripción	Nivel	CWE

		1	2	3	
C130	Los tokens de sesión basados en cookies deben poseer el atributo “Secure” configurado.	✓	✓	✓	614
C131	Los tokens de sesión basados en cookies deben poseer el atributo “HttpOnly” configurado.	✓	✓	✓	1004
C132	Los tokens de sesión basados en cookies deben poseer el atributo “SameSite” para limitar la exposición a ataques CSRF.	✓	✓	✓	16
C133	Los tokens de sesión basados en cookies deben utilizar el prefijo “__Host-” para dotar de confidencialidad a la cookie de sesión.	✓	✓	✓	16
C134	En caso de que la aplicación esté publicada bajo un nombre de dominio junto con otras aplicaciones que usan cookies que podrían sobreescribir o divulgar las cookies de la aplicación, se debe configurar el atributo “path” en los tokens de sesión basados en cookies especificando las rutas de la forma más precisa posible.	✓	✓	✓	16

4.A.9.5 Requerimientos de sesión basada en token

La gestión de sesión basada en tokens incluye JWT, OAuth, SAML y API keys. De estos casos, las API keys se consideran débiles y no deben utilizarse en el código fuente.

ID	Descripción	Nivel			CWE
		1	2	3	
C135	La aplicación debe permitir a los usuarios revocar los tokens OAuth que forman relaciones de confianza con aplicaciones vinculadas.		✓	✓	290
C136	La aplicación debe usar tokens de sesión en lugar de secretos estáticos o API keys, con excepción de implementaciones legacy.		✓	✓	798
C137	Los tokens de sesión sin estado deben utilizar firmas digitales, cifrado y otras contramedidas para proteger contra manipulación, replay o ataques de sustitución de claves.		✓	✓	345

4.A.9.6 Requerimientos de re-autenticación desde identidad federada

Esta sección se refiere a quienes desarrollan el código de tomado de Relying Parties (RP) o Credential Service Providers (Identity Providers). Si confía en el código que implementa estas características, asegúrese de que estos problemas se manejan correctamente.

ID	Descripción	Nivel			CWE
		1	2	3	
C138	Las Relying Parties deben especificar el tiempo máximo de autenticación al Identity Provider y este Identity Provider debe re-			✓	613

	autenticar al suscriptor si no se ha utilizado la sesión durante ese período de tiempo.				
C139	Los Identity Providers deben informar a las Relying Parties acerca del último evento de autenticación, para permitirles determinar si es necesario re-autenticar al usuario.			✓	613

4.A.9.7 Defensas contra explotación de la gestión de sesiones

ID	Descripción	Nivel			CWE
		1	2	3	
C140	La aplicación debe asegurar una sesión válida de login o requerir re-autenticación o verificación secundaria antes de permitir cualquier transacción o modificación de cuenta.	✓	✓	✓	306

4.B - Gestión de configuración: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con la gestión de configuración a ser implementados por el proveedor del mismo.

4.B.1 Gestión de claves y accesos

4.B.1.1 Almacenamiento de claves de la aplicación (API keys, clave de acceso a base de datos, etc.)

ID	Descripción	Nivel			CWE
		1	2	3	
GC1	Las claves utilizadas en la aplicación para conectarse a servicios externos o servicios internos deben almacenarse en variables de entornos o utilizar la bóveda de claves del framework (en caso de que existiera). No se deben embeber en el código fuente y no se deben guardar en un archivo de texto plano.	✓	✓	✓	N/D
GC2	Las claves de cada entorno (desarrollo, testing y producción) deben ser diferentes.	✓	✓	✓	N/D
GC3	Se deben regenerar las claves de producción de manera periódica.			✓	N/D
GC4	No se deben subir claves a los repositorios de código.	✓	✓	✓	N/D

4.B.1.2 Almacenamiento de claves de productos utilizados en el desarrollo de software y sus accesos

ID	Descripción	Nivel			CWE
		1	2	3	
GC5	Las claves de productos deben ser gestionadas y deben ser identificadas las personas a quienes se las comparte. Controlar los accesos autorizados a servidores, repositorios de código y de documentos, productos de gestión y otras herramientas.	✓	✓	✓	N/D
GC6	Los accesos a productos transversales al desarrollo de software deben estar correctamente configurados asegurando el mínimo privilegio a cada uno de sus usuarios. Ningún usuario debería poder hacer algo que no deba hacer.	✓	✓	✓	N/D
GC7	Cuando un integrante del equipo de desarrollo deja de ser parte del mismo deben ser eliminados todos sus accesos.	✓	✓	✓	N/D
GC8	Cada integrante del equipo de desarrollo de software debe usar un segundo factor de autenticación en cada producto usado.			✓	N/D

4.B.1.3 Transferencia de claves

ID	Descripción	Nivel			CWE
		1	2	3	
GC9	Las claves de productos deben ser transferidas a través de un medio seguro.	✓	✓	✓	N/D
GC10	Las claves de productos deben ser eliminadas del medio de comunicación una vez transferidas.	✓	✓	✓	N/D
GC11	Una clave debe ser transferida por un canal independiente del cual se envía el resto de los datos.	✓	✓	✓	N/D

4.B.2 Gestión de construcción de software

4.B.2.1 Requerimientos arquitecturales de configuración

ID	Descripción	Nivel			CWE
		1	2	3	

GC12	Se deben segregar los componentes de diferentes niveles de confianza a través de controles de seguridad bien definidos, reglas de firewall, API gateways, proxies reversos, grupos de seguridad basados en la nube o mecanismos similares.		✓	✓	923
GC13	Los builds de pipeline deben alertar sobre componentes inseguros y desactualizados. Y se deben tomar las acciones apropiadas ante esas alertas.		✓	✓	1004
GC14	Los builds deben contener un paso que realiza el build automáticamente y verifica el desarrollo seguro de la aplicación, particularmente si la infraestructura de la aplicación se define mediante código (ej.: scripts para hacer builds de entornos).		✓	✓	
GC15	La aplicación se debe desplegar, contenerizar y/o aislar a nivel de red para prevenir ataques inter aplicaciones, especialmente cuando se ejecutan acciones sensibles o peligrosas, como la serialización.		✓	✓	265
GC16	La aplicación no debe utilizar tecnologías del lado del cliente desactualizadas, no soportadas, o inseguras, tales como NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, o Java applets.		✓	✓	447

4.B.3.2 Gestión de versiones

ID	Descripción	Nivel			CWE
		1	2	3	
GC17	En la creación de una nueva versión de software se deben especificar las mejoras de seguridad implementadas en un registro de cambios de seguridad.			✓	N/D
GC18	El despliegue de versiones debe estar restringido a usuarios específicos.		✓	✓	N/D

4.B.3.2 Gestión de librerías (bibliotecas)

ID	Descripción	Nivel			CWE
		1	2	3	
GC19	Cada paquete de código externo que se agrega debe ser analizado si contiene vulnerabilidades y debe darse seguimiento de las mismas.	✓	✓	✓	N/D
GC20	Cada paquete de código externo debe mantener un ritmo de actualización frecuente ante posibles nuevas vulnerabilidades, o bien, justificar que las mismas no son explotables en el software desarrollado.	✓	✓	✓	N/D

GC21	Se debe analizar estáticamente cada paquete de código desarrollado por terceros del cual la aplicación dependa.			✓	N/D

4.C - Toma de requerimientos y Diseño: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con la Toma de Requerimientos y Diseño a ser implementados por el proveedor del mismo.

4.C.1 Arquitectura, Diseño y Modelado de Amenazas

4.C.1.1 Requerimientos del ciclo de vida de desarrollo seguro de software

ID	Descripción	Nivel			CWE
		1	2	3	
RD1	Se debe verificar que exista una clasificación de activos, identificando los componentes, elementos de la aplicación que ameritan defenderse.		✓	✓	N/D
RD2	Debe existir una identificación de casos de abuso, en paralelo a la definición de casos de uso.	✓	✓	✓	N/D
RD3	Se debe usar un modelado de amenazas para cada cambio en el diseño o planificación de sprint; para identificar amenazas, planificar contramedidas, facilitar respuestas apropiadas a los riesgos y guiar el testing de seguridad.		✓	✓	1053
RD4	Todas las historias de usuario y funcionalidades deben poseer restricciones de seguridad funcionales (ej.: “Como usuario, debería ser capaz de ver y editar mi perfil. No debería poder ver o editar el perfil de alguien más”).	✓	✓	✓	1110
RD5	Las historias de usuario deben poseer criterios de aceptación de seguridad.	✓	✓	✓	1110
RD6	Se deben documentar y justificar todos los límites de confianza de la aplicación, componentes y flujos de datos significativos.		✓	✓	1059
RD7	Se deben priorizar los requerimientos de seguridad / casos de abuso / historias de usuario, para decidir en qué orden se los incorpora al diseño y cuándo son descartados.	✓	✓	✓	N/D

RD8	Debe existir un análisis de riesgos, estimando la probabilidad de que ocurran ciertos eventos y evaluar cuál es el impacto para la organización. Debe existir asimismo un plan de acción.		✓	✓	N/D
RD9	Se debe definir y hacer un análisis de seguridad de la arquitectura de alto nivel de la aplicación y de los servicios remotos conectados.		✓	✓	1059
RD10	Se deben implementar controles de seguridad reutilizables, simples, seguros y centralizados, para evitar duplicados, omisiones o controles inseguros.		✓	✓	637
RD11	Debe haber disponible para todos los desarrolladores y testers un checklist de código seguro, requerimientos de seguridad, guías o políticas.		✓	✓	637
RD12	Se deberá evaluar en qué casos conviene la utilización de patrones de diseño tanto como buenas prácticas como prevención de vulnerabilidades. Ejemplo: patrón de estado para cambios de estado evita manipulaciones en el flujo.				N/D

4.C.1.2 Requerimientos arquitecturales de autenticación

ID	Descripción	Nivel			CWE
		1	2	3	
RD13	Se deben usar cuentas especiales únicas de sistema operativo con bajos privilegios para todos los componentes, servicios y servidores de aplicación.		✓	✓	250
RD14	La comunicación entre componentes de aplicación, incluyendo API, middleware y capas de datos, debe usar autenticación. Los componentes deben tener el mínimo privilegio requerido.		✓	✓	306
RD15	La aplicación debe utilizar un único mecanismo de autenticación verificado que se sepa que es seguro, que pueda extenderse para incluir autenticación más robusta, y que posea suficiente registro de logs y monitoreo para detectar fugas o abuso de cuentas.		✓	✓	306
RD16	Todos los caminos de la aplicación y API de gestión de identidades deben implementar controles robustos de seguridad de la autenticación, de tal manera que no haya alternativas más débiles según el riesgo de la aplicación.		✓	✓	306

4.C.1.3 Requerimientos arquitecturales de control de acceso

ID	Descripción	Nivel	CWE

		1	2	3	
RD17	Los gateways de control de acceso, servidores y funciones sin servidor deben cumplir con los controles de acceso. No se deben aplicar controles de acceso en el cliente únicamente, sino asegurarse que existen en el servidor dichos controles (defensa en profundidad).		✓	✓	602
RD18	La solución de controles de acceso elegida debe ser lo suficientemente flexible para cumplir con las necesidades de la aplicación.		✓	✓	284
RD19	Se debe comprobar que los privilegios de los componentes están aislados, para evitar que interfieran entre sí en caso de que alguno de ellos sea vulnerable.		✓	✓	
RD20	Se debe cumplir con el principio de mínimo privilegio en funciones, archivos de datos, controladores, servicios y otros recursos. Esto implica protección contra suplantación de identidad y escalamiento de privilegios.		✓	✓	272
RD21	La aplicación debe utilizar un único mecanismo de control para acceder a los datos y recursos protegidos. Todas las peticiones deben pasar por este único mecanismo para evitar rutas alternativas inseguras.		✓	✓	284
RD22	Se debe utilizar un control de acceso basado en funcionalidades mediante el cual el código verifique la autorización del usuario para el acceso a una funcionalidad o dato, en lugar de solo su función. Los permisos aún deben asignarse mediante roles.	✓	✓	✓	275

4.C.1.4 Requerimientos arquitecturales de entrada y salida

ID	Descripción	Nivel			CWE
		1	2	3	
RD23	Todos los requerimientos de entrada y salida deben definir claramente cómo manejar y procesar datos con base en el tipo, contenido y leyes aplicables, regulaciones y otras políticas de cumplimiento.		✓	✓	1029
RD24	No se debe utilizar serialización[6] en la comunicación con clientes no confiables[7]. Si esto no es posible, se debe asegurar que se aplican controles de integridad (con encriptación de ser posible, si se envía		✓	✓	502

[6] El término *serialización* refiere a información enviada como parte de arrays, objetos JSON o cualquier objeto complejo que contenga lógica.

[7] El concepto de *cliente no confiable* hace referencia a tecnologías del lado del cliente que renderizan la capa de presentación, comúnmente referida como *frontend*.

	información sensible), para prevenir ataques de deserialización incluyendo inyección de objetos.				
RD25	La validación de entradas se debe aplicar en una capa de servicio confiable[8].		✓	✓	602
RD26	La codificación de salida se debe producir cerca o por el intérprete para el que está destinada.		✓	✓	116

4.C.1.5 Requerimientos arquitecturales de criptografía

ID	Descripción	Nivel			CWE
		1	2	3	
RD27	Debe existir una política explícita para la gestión de claves criptográficas y el ciclo de vida de claves debe seguir un estándar.[9]		✓	✓	320
RD28	Los consumidores de los servicios criptográficos deben proteger el material referido a claves y otros secretos mediante baúles de claves o alternativas basadas en API.		✓	✓	320
RD29	Todas las claves y contraseñas deben ser reemplazables y parte de un proceso bien definido para cifrar datos sensibles.		✓	✓	320
RD30	La arquitectura debe tratar los secretos del lado del cliente (como claves simétricas, contraseñas, tokens de API) como inseguros y nunca los debe utilizar para proteger o acceder a datos sensibles.		✓	✓	320
RD31	Se debe seguir la premisa de diseño sin secretos. Se debe diseñar el sistema bajo la premisa de que el público conocerá los detalles de su funcionamiento interno.	✓	✓	✓	N/D

4.C.1.6 Requerimientos arquitecturales de errores, registro de logs y auditoría

ID	Descripción	Nivel			CWE
		1	2	3	
RD32	Se debe utilizar un formato de registro de logs común en todo el sistema.		✓	✓	1009

[8] El concepto de *capa de servicio confiable* se refiere a cualquier punto confiable, independientemente de su ubicación. Por ejemplo, microservicio, *API serverless*, *API* externa, etc.

[9] Por ejemplo, un estándar como NIST, "Recommendation for Key Management", Special Publication 800-57.

RD33	Los logs se deben transmitir de manera segura a un sistema remoto para análisis, detección, alertas y escalamiento.		✓	✓	N/D
RD34	Se debe evitar mostrar información sensible de los usuarios de la plataforma a los administradores.	✓	✓	✓	N/D

4.C.1.7 Requerimientos arquitecturales de protección de datos y privacidad

ID	Descripción	Nivel			CWE
		1	2	3	
RD35	Todos los datos sensibles deben ser identificados y clasificados en niveles de protección.		✓	✓	N/D
RD36	Todos los niveles de protección deben tener un conjunto asociado de requerimientos de protección, como requisitos de cifrado, requisitos de integridad, retención, privacidad y otros requisitos de confidencialidad; y que estos se apliquen en la arquitectura.		✓	✓	N/D

4.C.1.8 Requerimientos arquitecturales de comunicación

ID	Descripción	Nivel			CWE
		1	2	3	
RD37	La aplicación debe cifrar la comunicación entre componentes, particularmente cuando estos componentes están en diferentes contenedores, sistemas, sitios o proveedores de la nube.		✓	✓	319
RD38	Los componentes de la aplicación deben verificar la autenticidad de cada lado en un enlace de comunicación (ej.: validar los certificados TLS), para prevenir ataques man-in-the-middle.		✓	✓	295
RD39	Al diseñar la red, considerar las posibles amenazas a las que se verá expuesta, y las contramedidas a adoptar.				N/D

4.C.1.9 Requerimientos arquitecturales de software malicioso

ID	Descripción	Nivel			CWE
		1	2	3	
RD39	Se debe utilizar un sistema de control de versiones sobre el código fuente con procedimientos para asegurar que los check-ins son acompañados de issues o tickets de cambio. Este sistema de control		✓	✓	284

	del código debe tener controles de acceso y usuarios identificables para permitir trazabilidad de cualquier cambio.				

4.C.1.10 Requerimientos arquitecturales de lógica de negocio

ID	Descripción	Nivel			CWE
		1	2	3	
RD40	Se deben definir y documentar todos los componentes de la aplicación en términos de negocio o funciones de seguridad que proveen.		✓	✓	1059
RD41	Todos los flujos de lógica de negocio de alto valor, incluida la autenticación, la gestión de sesiones y el control de acceso, no deben compartir estados no sincronizados.		✓	✓	362
RD42	Todos los flujos de lógica de negocio de alto valor, incluida la autenticación, la gestión de sesiones y el control de acceso, deben ser resistentes a ataques de condiciones de carrera.			✓	367

4.C.1.11 Requerimientos arquitecturales de subida de archivos

ID	Descripción	Nivel			CWE
		1	2	3	
RD43	Todos los archivos subidos por los usuarios se deben almacenar fuera de la raíz del sitio web.		✓	✓	552
RD44	Los archivos subidos por los usuarios, si se requiere que sean mostrados o descargados desde la aplicación, deben ser servidos ya sea mediante descargas octet-stream o de un dominio aparte (ej.: archivo almacenado en bucket). Se debe implementar una Content Security Policy (CSP) para reducir el riesgo de vectores XSS u otros ataques desde el archivo subido.		✓	✓	646

4.D - Despliegue: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con el despliegue a ser implementados por el proveedor del mismo.

4.D.1 Requerimientos de Puesta en producción

4.D.1.1 Segregación de ambientes

ID	Descripción	Nivel			CWE
		1	2	3	
D1	Debe existir segregación entre ambientes de desarrollo, pruebas y producción.		✓	✓	N/D
D2	No deben existir accesos irrestrictos a los entornos productivos.		✓	✓	N/D
D3	No deben utilizarse datos productivos en entornos de desarrollo y pruebas. Es recomendable la generación de datos ficticios o aleatorios con estructura equivalente a la productiva.		✓	✓	N/D
D4	Debe existir una serie de controles formales previo a la puesta en producción de nuevas versiones de la aplicación.		✓	✓	N/D
D5	No deben existir backdoors administrativas al producirse la migración de código hacia entornos productivos.		✓	✓	N/D

4.D.1.2 Hardenizado de equipos

ID	Descripción	Nivel			CWE
		1	2	3	
D6	Se deben eliminar los componentes innecesarios, configurar correctamente los necesarios, activar componentes de seguridad y documentar las configuraciones establecidas para cada componente.	✓	✓	✓	N/D
D7	Las contraseñas preestablecidas por defecto deben ser cambiadas, cumpliendo con la política de contraseñas de la organización.	✓	✓	✓	N/D
D8	Se deben seguir las buenas prácticas de seguridad recomendadas por cada fabricante y comunidad de usuarios expertos, en la configuración del sistema operativo y en los servicios utilizados por la aplicación.	✓	✓	✓	N/D
D9	Debe existir una separación entre equipos de servicios web y bases de datos, ubicando el servicio web en la DMZ y el servicio de base de datos en un entorno privado.	✓	✓	✓	N/D
D10	Debe existir una comunicación cifrada entre los equipos de servicios web y de bases de datos.	✓	✓	✓	N/D

D11	Debe existir un esquema de particiones que separe el sistema operativo, servicios, los registros de auditoría y archivos cargados por usuarios.		✓	✓	N/D
D12	Las cuentas de servicio y administrativas deberán contar con los mínimos privilegios necesarios para realizar las acciones previstas.	✓	✓	✓	N/D
D13	Debe existir un servicio de cifrado para datos en tránsito y el cifrado de las unidades de almacenamiento, de manera de proteger el código fuente, las bases de datos y los datos personales.	✓	✓	✓	N/D
D14	Deben existir planes de prueba y sus correspondientes ejecuciones de dichas pruebas y revisiones de la configuración para todos los componentes. Es preferible lograr una configuración base segura que pueda adaptarse para cada caso de uso.	✓	✓	✓	
D15	Deben existir herramientas de seguridad para cada capa. Por citar ejemplos: ● Red: balanceadores de carga, firewall, sistemas de detección y prevención de intrusos. ● Sistema operativo: firewall de host, controles de integridad de archivos, cifrado, scripts de hardening, antimalware. ● Servicios: firewall de aplicaciones, módulos de seguridad, scripts de hardening. ● Aplicación: Correcta configuración, componentes de seguridad, generación de registros de auditoría.	✓	✓	✓	N/D
D16	Deben existir herramientas en el pipeline que realicen los escaneos de seguridad básicos de las imágenes y configuraciones de contenedor, con el fin de encontrar vulnerabilidades durante las integraciones y despliegues.		✓	✓	N/D

4.D.1.3 Hardening en contenedores

ID	Descripción	Nivel			CWE
		1	2	3	
D17	La tecnología que se esté utilizando para contenedores (como Docker) debe estar actualizada, para mitigar las vulnerabilidades del software.	✓	✓	✓	N/D
D18	Solo los usuarios de confianza deben poder controlar los daemons, para minimizar los privilegios administrativos. Solo se deben usar cuentas administrativas cuando sea requerido.		✓	✓	N/D
D19	La auditoría debe estar configurada para los procesos de contenedor, para los archivos y directorios y para las actividades y uso de los procesos daemon que se ejecutan con privilegios de root.		✓	✓	N/D
D20	Los contenedores no deben correr con usuario root.	✓	✓	✓	N/D

D21	La característica de “content trust” debe estar habilitada. Esto provee la capacidad de utilizar firmas digitales para datos enviados y recibidos entre los registries. Estas firmas permiten la verificación del lado del cliente de la identidad de quien publica la imagen y asegura la proveniencia de las imágenes de contenedor.	✓	✓	✓	N/D
D22	El HEALTHCHECK debe estar habilitado para las imágenes de contenedor. Esto permite controles para asegurar que los contenedores continúan operativos.		✓	✓	N/D
D23	El tráfico de red debe estar restringido entre contenedores.	✓	✓	✓	N/D
D24	No se deben utilizar registries inseguros. Un registry seguro utiliza TLS y un certificado válido.	✓	✓	✓	N/D
D25	Debe existir autenticación TLS para los daemons.	✓	✓	✓	N/D
D26	Las imágenes deben estar firmadas de manera que se pueda controlar el origen y la validez de la imagen del contenedor, previo al despliegue. Se deben prohibir imágenes que no estén firmadas, o que su firma no sea correcta o esté comprometida.	✓	✓	✓	N/D
D27	Se debe habilitar el escaneo de vulnerabilidades en imágenes contra las bases de datos de MITRE, Common Vulnerabilities and Exposures (CVE), y NIST, National Vulnerability Database (NVD).		✓	✓	N/D
D28	De ser aplicable, debe estar habilitado un perfil de AppArmor. Habilitando esta característica, se refuerzan las políticas de seguridad en contenedores.		✓	✓	N/D
D29	De ser aplicable, las opciones de seguridad de SELinux deben estar configuradas, para dar una capa extra de seguridad a los contenedores.		✓	✓	N/D
D30	Las capacidades del kernel de Linux dentro de los contenedores deben estar restringidas. Lo contrario permite a un atacante con acceso al contenedor, crear tráfico de red manipulado.		✓	✓	N/D
D31	No deben correrse contenedores privilegiados.	✓	✓	✓	N/D
D32	Los directorios sensibles (ej.: /, /boot, /etc) no deben ser montados en los contenedores. Si estos directorios se montaran en modo de lectura y escritura, se estaría permitiendo hacer cambios a los archivos que existen allí.	✓	✓	✓	N/D
D33	El daemon de SSH (sshd) no debe ser ejecutado en los contenedores porque incrementa la complejidad para la gestión de la seguridad.		✓	✓	N/D

D34	El namespace de la red del host no debe estar compartido. De lo contrario, podrían permitirse acciones no deseadas (ej.: apagado del host).	✓	✓	✓	N/D
D35	El uso de memoria para contenedores debe estar limitado. Lo contrario puede conducir a que el contenedor haga que el sistema se vuelva inestable e inutilizable.		✓	✓	N/D
D36	El sistema de archivos root del contenedor debe estar montado en modo solo lectura. Esta opción reduce vectores de ataque, ya que el sistema de archivos del contenedor no puede ser manipulado o escrito (a menos que tenga permisos de lectura y escritura en las carpetas del sistema de archivos).	✓	✓	✓	N/D
D37	El tráfico de contenedor entrante debe estar vinculado a una interfaz del host específica.	✓	✓	✓	N/D
D38	Deben existir herramientas de control automatizado que chequean CIS benchmark.		✓	✓	N/D
D39	Deben existir herramientas en el pipeline que realicen los escaneos de seguridad básicos de las imágenes y configuraciones de contenedor, con el fin de encontrar vulnerabilidades durante las integraciones y despliegues.		✓	✓	N/D

4.D.1.4 Hardening en orquestadores

ID	Descripción	Nivel			CWE
		1	2	3	
D40	La autenticación contra el nodo master no debe permitir acceso anónimo.	✓	✓	✓	N/D
D41	La autenticación contra el nodo master no debe permitir acceso Basic.		✓	✓	N/D
D42	La autenticación contra el nodo master no debe permitir cualquier token.	✓	✓	✓	N/D
D43	No se debe permitir acceso no cifrado entre los kubelets (en caso de utilizar Kubernetes como orquestador) y el servidor API y que la versión mínima de TLS es 1.2.	✓	✓	✓	N/D
D44	No debe haber mapeos entre el servidor de API y una IP insegura.	✓	✓	✓	N/D
D45	No debe haber mapeos en el servidor de API hacia puertos inseguros.	✓	✓	✓	N/D
D46	Debe estar habilitada la autorización de tipo RBAC.	✓	✓	✓	N/D

D47	Deben existir herramientas de control automatizado que chequean CIS benchmark.		✓	✓	N/D

4.E - Monitoreo y control: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con monitoreo y control a ser implementados por el proveedor del mismo.

4.E.1 Gestión de disponibilidad

4.E.1.1 Monitoreo de software

ID	Descripción	Nivel			CWE
		1	2	3	
MC1	Cada módulo del sistema debe estar monitoreado por agentes de monitoreo para controlar si está en servicio o fuera de servicio.	✓	✓	✓	N/D
MC2	Cada agente de monitoreo debe estar replicado para evitar falsos positivos.	✓	✓	✓	N/D
MC3	Cada error generado en el sistema debe ser registrado y etiquetado en niveles de criticidad.	✓	✓		N/D

4.E.1.2 Respuestas ante emergencias

ID	Descripción	Nivel			CWE
		1	2	3	
MC4	Cada módulo del sistema debe ser controlado de tal manera que ante su salida imprevista de servicio se lo debe restablecer automáticamente o se deben generar las alertas suficientes para hacerlo lo antes posible.	✓	✓	✓	N/D
MC5	Se deben enumerar los diferentes tipos de error posibles, identificar aquellos que podrían ocurrir como consecuencia de un ataque y definir las acciones que se ejecutarán ante la ocurrencia de cada uno de ellos.	✓	✓	✓	N/D

4.E.2 Gestión de fuga de información

ID	Descripción	Nivel			CWE
		1	2	3	
MC6	Cada activo del sistema debe monitorearse para evitar fuga de información.	✓	✓	✓	N/D
MC7	Cada módulo del sistema debe monitorearse para evitar que exponga información sensible en forma pública.	✓	✓	✓	N/D

4.E.3 Monitoreo de infraestructura

ID	Descripción	Nivel			CWE
		1	2	3	
MC8	Se deben monitorear frecuentemente puertos abiertos y servicios en la infraestructura asociada al sistema.	✓	✓	✓	N/D
MC9	Se deben controlar los accesos a la infraestructura y tener un seguimiento de los mismos.	✓	✓	✓	N/D
MC10	Los aspectos de hardware de la infraestructura deben estar monitoreados para verificar que se encuentren entre los niveles aceptables.	✓	✓	✓	N/D

4.F Pruebas: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con Pruebas a ser implementados por el proveedor del mismo.

ID	Descripción	Nivel			CWE
		1	2	3	
P1	Verificar que existan pruebas unitarias en el desarrollo, abarcando mínimamente las vulnerabilidades del top 10 de OWASP.[10]	✓	✓	✓	N/D
P2	Validar que se lleven a cabo revisiones de código seguras entre pares, para los componentes más críticos de la aplicación		✓	✓	N/D

[10] Disponible en https://owasp.org/Top10/.

P3	Verificar que exista una instancia de escaneo estático de código previa a la subida del código por parte del desarrollador a los repositorios (integraciones). Por ejemplo: herramientas / plugins en los entornos de desarrollo utilizados.	✓	✓	✓	N/D
P4	Verificar que se realicen pruebas de penetración internas de caja negra, para detectar vulnerabilidades que hayan escapado a los controles mencionados anteriormente.	✓	✓	✓	N/D
P5	Verificar que se realicen pruebas de caja blanca sobre el código, de manera manual.	✓	✓	✓	N/D

4.G Gestión de datos: requerimientos de seguridad por nivel adoptado

A continuación, se definen los requerimientos de seguridad relacionados con gestión de datos a ser implementados por el proveedor del mismo.

4.G.1 Backups

ID	Descripción	Nivel			CWE
		1	2	3	
GD1	Realizar backups de los datos generados por la aplicación. Se deberá definir la frecuencia de acuerdo a la importancia y a la tasa de cambio de los datos.	✓	✓	✓	N/D
GD2	Realizar backup de la instancia virtual (si correspondiera).		✓	✓	N/D
GD3	Realizar backup de la configuración del despliegue (si correspondiera).		✓	✓	N/D
GD4	Diseñar procedimiento paso a paso para la restauración de cada backup.	✓	✓	✓	N/D
GD5	Demostrar que la restauración de backups funciona.	✓	✓	✓	N/D

4.G.2 Integridad

ID	Descripción	Nivel			CWE
		1	2	3	
GD6	Desarrollar procesos que auditen la integridad de los datos para detectar manipulaciones indebidas.			✓	N/D
GD7	Utilizar funciones hash para verificar integridad en transferencias de grandes volúmenes de datos.			✓	N/D

4.G.3 Análisis

ID	Descripción	Nivel			CWE
		1	2	3	
GD8	Analizar y priorizar los datos más sensibles en cuanto a confidencialidad, integridad y disponibilidad.			✓	N/D
GD9	Analizar los permisos de acceso y modificación sobre los datos para roles externos e internos al equipo de desarrollo y/o mantenimiento.			✓	N/D
GD10	Analizar si para el acceso a determinados datos es necesario MFA.		✓	✓	N/D

5 Glosario

2FA: Abreviatura de Segundo Factor de Autenticación.

Activo del sistema: Un activo del sistema es un recurso con valor, por ejemplo, puede ser IPs, dominios, repositorios, endpoints de callback, APIs, etc.

Agente de monitoreo: Es un software que monitorea a otro software y que presenta los resultados de forma amigable para que un humano interpretelo que está ocurriendo.

API key: Una clave para autenticarse ante un servicio externo o interno.

AppArmor: Módulo de seguridad del kernel Linux que permite al administrador del sistema restringir las capacidades de un programa, de manera tal de proteger al sistema operativo y las aplicaciones de varias amenazas, reforzando la política de seguridad. Para definir las restricciones se asocia a cada programa un perfil de seguridad**.**

Ataque man-in-the-middle: Es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas y procurar que ninguna de las víctimas conozca que el enlace entre ellos ha sido violado.

CSRF: *Cross-Site Request Forgery* es un ataque que falsifica una petición a un servidor web haciéndose pasar por un usuario de confianza.

CWE: *Common Weakness Enumeration* es un listado de tipos de vulnerabilidades mantenido por la comunidad dedicada a la ciberseguridad. Sirve como un lenguaje común y como una línea de base para los esfuerzos de identificación, mitigación y prevención de debilidades.

Hardening: Proceso de asegurar un sistema mediante la reducción de vulnerabilidades en el mismo.

MFA: Multi-Factor Authentication.

N/D: No disponible.

OWASP: *Open Web Application Security Project* es una fundación sin fines de lucro que trabaja para mejorar la seguridad del software.

Paquete de código externo: Cuando en este documento se hace referencia a paquete de código externo, se refiere a código desarrollado por terceras partes que se añade al proyecto de software. En general, estos paquetes se agregan a través de gestores de paquetes o gestores de dependencias como npm, pip, NuGet, entre otros.

Path traversal: Es un ataque dirigido a obtener acceso a archivos y directorios ubicados fuera del directorio raíz de un sitio web.

RFD: *Reflective File Download* es un ataque que permite ganar control completo de la máquina de la víctima mediante la descarga virtual de un archivo desde dominios confiables.

RFI: *Remote File Inclusion* es un ataque que permite incluir un archivo, usualmente explotando mecanismos de inclusión dinámica de archivos.

SSRF: *Server-side request forgery* es una vulnerabilidad asociada a aplicaciones web que permite a un atacante inducir a la aplicación del lado del servidor a realizar solicitudes HTTP a un dominio arbitrario de la elección del atacante.

Template injection: Es un ataque que permite ejecutar comandos en un servidor a partir de la inyección de una entrada maliciosa en una plantilla usada para generar dinámicamente código HTML.

XSS: Cross-Site Scripting es un ataque de inyección que permite utilizar una aplicación web para enviar código malicioso para ser ejecutado por otro usuario en la forma de un script ejecutado por el navegador web.

XXE: *XML External Entity* es un ataque a una aplicación que procesa entradas XML. Ocurre cuando una entrada XML que contiene una referencia a una entidad externa es procesada por un *parser XML* configurado de forma deficiente.

Zip bomb: Archivo de tamaño pequeño que al descomprimirse supera los límites de almacenamiento.

.

6 Anexos

6.A - Codificación

6.A.1 Validación de entradas

Las validaciones de entradas deben aplicarse a todas las entradas.

Se debe definir el conjunto de caracteres permitido, así como también el mínimo y máximo de longitud de los datos.

6.A.1.1 Estrategias de validación de entradas

Se debería aplicar de forma sintáctica y semántica.

6.A.1.1.1 Validación sintáctica

Se debe reforzar la sintaxis correcta de campos estructurados (ej.: fecha, moneda, etc.).

6.A.1.1.2 Validación semántica

La validación debe reforzar la exactitud de los valores en un contexto de negocio específico (ej.: fechas de inicio son menores a fechas de fin, precios dentro del rango esperado).

6.A.1.2 Requerimientos de seguridad

Se debe tener en cuenta que las validaciones implementadas mediante JavaScript pueden saltarse fácilmente por el atacante que deshabilita JavaScript o utiliza un proxy web. Toda validación que se ejecute en el cliente, también debe ejecutarse en el servidor.

6.A.1.2.1 Validadores y conversión de tipo de datos

Es preferible la utilización de validadores y conversores de datos que estén disponibles de forma nativa en frameworks de aplicaciones web (ej.: validadores de Django, Apache Common. validaciones contra esquemas JSON y XML para entradas en esos formatos, en Java, Integer.parseInt(), int en Python) con manejo estricto de excepciones.

6.A.1.2.2 Control de rangos

Se deben controlar rango de valores mínimos y máximos para números y fechas; y controlar la longitud mínima y máxima para cadenas.

6.A.1.2.3 Campos requeridos

En campos requeridos, validar que la entrada no sea nula o una cadena vacía.

6.A.1.2.4 Lista blanca vs. lista negra

Es recomendable utilizar lista blanca en lugar de lista negra, ya que esto permite definir exactamente qué está permitido, y por definición, cualquier otra cosa no estará autorizada.

Si se trata de datos estructurados (ej.: fechas, códigos postales, direcciones de correo, etc.), se deberá definir un patrón robusto de validación basado en expresiones regulares.

Es recomendable utilizar vectores con valores permitidos para conjuntos de parámetros cadena (ej.: días de la semana).

6.A.1.2.5 Uso de expresiones regulares

Se deben usar expresiones regulares para el procesamiento de cualquier dato estructurado, cubriendo toda la entrada de la cadena (^...$), evitando utilizar wildcards que coincidan con cualquier caracter (o clases de caracteres, ej.: \S).[11]

El diseño de una expresión regular tiene que tener en cuenta posibles ataques de denegación de servicio mediante su explotación.[12]

6.A.2 Prevención de XSS y Content Security Policy

Cuando se retornan datos mediante HTML, la información deberá codificarse para prevenir la ejecución de datos maliciosos. Por ejemplo, la etiqueta <script> sería retornada como <script>.[13]

6.A.3 Utilización de manera correcta de los verbos HTTP

Se deben utilizar los verbos de HTTP de acuerdo a la semántica de cada uno de ellos. Por ejemplo, utilizar GET solo para consulta y POST y PUT para registrar o modificar información.

6.A.4 Utilizar token CSRF

Se deben utilizar token anti-CSRF, para evitar ataques en formularios remitidos a través del método POST.

---

[11] Para más información, se pueden consultar https://www.regular-expressions.info/ y https://owasp.org/www-community/OWASP_Validation_Regex_Repository.

[12] Para más información, se puede consultar https://owasp.org/www-community/attacks/Regular_expression_Denial_of_Service_-_ReDoS.

[13] Para recomendaciones adicionales acerca de prevención de XSS, se puede consultar https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html,

6.A.5 Gestión de archivos subidos por el usuario

6.A.5.1 Validaciones al momento de la subida de archivos[14]

Cuando se reciben archivos se deben hacer las siguientes validaciones:

- Los nombres de los archivos deben contener el tipo de extensión esperada.
- El archivo subido no debe superar el tamaño máximo definido por el análisis funcional.

Si se aceptan archivos ZIP, se deben realizar las siguientes validaciones, antes de descomprimir el archivo:

- ruta objetivo,
- nivel de compresión,
- tamaño estimado del archivo descomprimido.

6.A.5.2 Almacenamiento de archivos subidos

Antes de almacenar un archivo subido se deben realizar las siguientes acciones:

- Utilizar un nuevo nombre de archivo al almacenarlo en el sistema operativo. No utilizar ningún texto controlado por el usuario o utilizado para el almacenamiento temporal, sino un valor aleatorio (ej.: renombrar el archivo test.pdf a JAI1287uaisdjhf.pdf), esto previene el riesgo de que se acceda al archivo directamente y las evasiones de filtro (ej.: test.jpg; .asp o /../../../../../test.jpg).
- Decidir la ruta del archivo en el servidor (nunca debe especificarse del lado del cliente).
- Analizar el archivo por contenido malicioso (anti-malware, análisis estático, etc).

6.A.5.3 Publicación del contenido subido

Las imágenes deben ser publicadas con el encabezado Content-Type correcto (ejemplo: image/jpeg, application/x-xpinstall).

6.A.5.4 Archivos especiales

Se debe utilizar un enfoque de allow-list (lista blanca) para permitir sólo subida de tipos de archivo y extensiones específicos.

Permitir la subida de los siguientes tipos de archivos es problemático porque puede conducir a la explotación de vulnerabilidades:

- Archivos con nombre crossdomain.xml o clientaccesspolicy.xml deben ser prohibidos porque pueden dar lugar al robo de datos y a ataques CSRF.
- Archivos con nombre .htaccess y .htpasswd deben ser prohibidos porque poseen configuraciones del servidor.

[14] Se pueden consultar recomendaciones adicionales en https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html.

- Archivos ejecutables como aspx, asp, css, swf, xhtml, rhtml, shtml, jsp, js, pl, php, cgi deben ser prohibidos.

6.A.5.5 Verificación de subida de imágenes

Antes de almacenar un archivo de imagen subido se deben realizar las siguientes acciones:

- Utilizar librerías de reescritura de imágenes para verificar que la imagen es válida y para quitar contenido extraño.
- Configurar la extensión de la imagen almacenada para que sea una extensión válida basada en el tipo de contenido detectado en el procesamiento de la imagen (no confiar únicamente en la cabecera Content-Type de la petición con la imagen subida).
- Asegurarse que el tipo de contenido esté en una lista de tipos de imágenes definidos.

6.A.6 Validación de direcciones de correo

La mejor manera de validar direcciones de correo es ejecutar algunas validaciones iniciales y luego pasar la dirección al servidor de correo y capturar la excepción en caso de que la rechace. Estas validaciones iniciales deben ser:

- Que existan dos partes, separadas por @.
- Que la dirección no contenga caracteres peligrosos (backticks, comillas simples o dobles, null bytes). Estos caracteres dependerán de cómo se tratará la dirección (ej.: si se insertará en una página o si quedará registrada en la base de datos).
- Que la parte de dominio contenga sólo letras, números, guiones y puntos.
- Que la longitud sea razonable: la parte local no debe superar los 63 caracteres. La longitud total no debe superar los 254 caracteres.

6.A.7 Manejo de errores[15]

La aplicación debe cubrir toda posible falla y retornar errores 5xx sólo para indicar respuestas a peticiones que no puede resolver y errores del lado del backend.

No se debe incorporar en la respuesta hacia el cliente ningún contenido que revele detalles de implementación. Se debe retornar un error genérico independientemente de los errores internos que hayan ocurrido.

Considerar el retorno de errores 4xx para casos en donde hay errores en el cliente HTTP (ej.: accesos no autorizados, cuerpo de peticiones muy largo, etc.).

Monitorear las aplicaciones cuando ocurren errores 5xx. Son indicadores de que hay fallas en ciertos conjuntos de entradas.

[15] Para recomendaciones adicionales, consultar https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html.

Respetar la referencia de errores RFC 2616[16].

7 Historial de cambios

Rev	Vigencia	Descripción	Revisión	Aprobación
00	22/12/2021	Primera versión consolidada.		
01	22/05/2021	Elimina referencia a la existencia de múltiples documentos. Corrige texto del requerimiento C14. Incorpora las secciones 4.F (Pruebas) y 4.G (Gestión de datos).		

[16] Disponible en https://datatracker.ietf.org/doc/html/rfc2616/.