# Informe de Análisis de Seguridad - Aplicación Médica

Fecha de análisis: 2025-12-03

- OWASP Application Security Verification Standard (ASVS) Versión 4.0.3

- OWASP Web Security Testing Guide (WSTG)

Nota: Este informe ha sido generado automáticamente mediante análisis de código e integración con DefectDojo.

## 1. Descripción Breve de la Aplicación Analizada

### *1.1. Propósito y Funcionalidad*

La aplicación es una herramienta web monousuario diseñada para el registro personal de peso, talla y cálculo del Índice de Masa Corporal (IMC). Su objetivo principal es permitir a un único usuario realizar un seguimiento de su peso corporal y obtener información sobre su estado nutricional mediante el cálculo automático del IMC.

### *1.2. Características Principales*

• Registro de datos personales: Nombre, apellidos, fecha de nacimiento y talla (en metros)
• Registro de peso: Permite registrar el peso actual con fecha y hora automáticas
• Cálculo automático de IMC: Calcula y muestra el Índice de Masa Corporal basado en el último peso registrado
• Estadísticas históricas: Muestra número de pesajes, peso máximo y peso mínimo registrados
• Sincronización bidireccional: Entre frontend (localStorage) y backend (memoria)
• Modo offline: Funciona sin conexión al servidor utilizando almacenamiento local

### *1.3. Arquitectura Técnica*

• Backend: Flask (Python) con API REST
• Frontend: JavaScript vanilla con almacenamiento en localStorage
• Almacenamiento: Memoria en backend + localStorage en frontend
• Tests: 86 tests backend (pytest) + ~66 tests frontend (Jest)
• Gestión de vulnerabilidades: DefectDojo integrado para seguimiento de debilidades de seguridad

### *1.4. Análisis de Datos que Maneja la Aplicación*

La aplicación maneja diferentes tipos de datos que requieren diferentes niveles de protección:

#### *1.4.1. Datos Sensibles (Personales de Salud)*

• Peso corporal (kg): Dato biométrico personal
• Talla/Altura (m): Dato biométrico personal
• Fecha de nacimiento: Permite inferir edad y otros datos demográficos
• Nombre completo: Identificador personal
• Apellidos: Identificador personal
Clasificación según RGPD: Estos datos están categorizados como datos personales sensibles según el Reglamento General de Protección de Datos, ya que los datos de salud (peso, altura, IMC) están incluidos en la categoría de datos especiales.

Almacenamiento actual:

- Frontend: localStorage del navegador (cliente)

- Backend: Memoria (volátil, se pierde al reiniciar)

## 2. Análisis de Seguridad Realizado

### *2.1. Metodología*

El análisis de seguridad se ha realizado mediante:

- Integración con DefectDojo: Obtención de benchmarks ASVS y findings reales

- Análisis estático de código: Revisión del código fuente Python y JavaScript

- Verificación de cumplimiento ASVS 4.0.3: Comparación con requisitos del estándar

- Mapeo de findings: Relación de vulnerabilidades con requisitos ASVS

- Análisis de arquitectura: Revisión de la estructura y diseño de la aplicación

### *2.2. Herramientas Utilizadas*

• DefectDojo: Benchmarks ASVS y gestión de findings
• Análisis automático de código fuente
• Verificación de patrones de seguridad
• Comparación con estándares OWASP ASVS 4.0.3

## 3. Debilidades de Seguridad Identificadas

### *3.1. Resumen*

El análisis automático ha identificado áreas de mejora en la aplicación. Las debilidades principales están relacionadas con:

- Validación de tipos numéricos (NaN/Infinity)

- Configuración de CORS para producción

- Mejora del logging de errores

## 4. Nivel ASVS Seleccionado, con Justificación

### *4.1. OWASP Application Security Verification Standard (ASVS)*

Versión utilizada en este informe: ASVS 4.0.3 (versión estable, lanzada el 28 de octubre de 2021)

Fuente oficial: OWASP ASVS v4.0.3 en GitHub

El OWASP Application Security Verification Standard (ASVS) versión 4.0.3 es un estándar de seguridad para aplicaciones web que define tres niveles de verificación. Esta versión se centra en corregir errores ortográficos y clarificar requisitos sin introducir cambios significativos ni romper compatibilidad con versiones anteriores.

Estructura de categorías ASVS 4.0.3: 14 categorías de verificación (V1-V14):

- V1: Architecture, Design and Threat Modeling

- V2: Authentication

- V3: Session Management

- V4: Access Control

- V5: Validation, Sanitization and Encoding

- V6: Stored Cryptographically Sensitive Data
- V7: Error Handling and Logging
- V8: Data Protection
- V9: Communications
- V10: Malicious Code
- V11: Business Logic
- V12: Files and Resources
- V13: API
- V14: Configuration

## *4.2. Nivel Seleccionado: NIVEL 2 (ESTÁNDAR)*

## *4.3. Justificación de la Selección*

La aplicación maneja datos de salud personales (peso, altura, fecha de nacimiento), aunque no sean datos médicos críticos ni información de identificación sensible. Según el Reglamento General de Protección de Datos (RGPD), los datos de salud están categorizados como datos personales sensibles, lo que requiere un nivel de protección superior al básico.

## *4.4. Enumeración de Requerimientos ASVS Nivel 2*

Basado en el análisis automático realizado, se enumeran los requerimientos ASVS Nivel 2 relevantes para esta aplicación. El estándar ASVS 4.0.3 define 14 categorías de verificación (V1-V14). Se detallan las aplicables:

### *4.4.1. V1: Architecture, Design and Threat Modeling*

Estado de cumplimiento: ■ CUMPLE

Explicación del cumplimiento:

■ V1.1: La aplicación cuenta con documentación de arquitectura (README.md)

■ V1.2: Existe documentación de análisis de amenazas en la carpeta docs/

■ V1.3: Se implementan principios de seguridad mediante validaciones y manejo de errores

■ V1.4: La aplicación utiliza una arquitectura REST con separación clara de responsabilidades (frontend/backend)

### *4.4.2. V2: Authentication*

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable

Justificación: Aplicación monousuario sin autenticación compleja

### *4.4.3. V3: Session Management*

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable

Justificación: No utiliza sesiones en el servidor

### *4.4.4. V4: Access Control*

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable

Justificación: Aplicación monousuario sin control de acceso entre usuarios

#### *4.4.5. V5: Validation, Sanitization and Encoding*

Estado de cumplimiento: ■ CUMPLE

Explicación del cumplimiento:

■ V5.1: Se implementa validación de entrada tanto en frontend (JavaScript) como en backend (Python)

■ V5.2: Se valida el tipo de datos en todas las entradas (nombres, números, fechas)

■ V5.3: Se implementa sanitización de entrada mediante la función validate_and_sanitize_name()

■ V5.4: Se valida que los valores numéricos sean válidos (no NaN ni Infinity)

■ V5.5: Se validan límites de rango para altura (0.4-2.72m) y peso (2-650kg) mediante configuración centralizada

■ V5.6: Se valida el formato de fechas (formato ISO)

#### *4.4.6. V6: Stored Cryptographically Sensitive Data*

Requerimientos ASVS Nivel 2 aplicables:

- ■■ No aplicable

Justificación: Datos almacenados localmente en cliente, no en servidor

#### *4.4.7. V7: Error Handling and Logging*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■■ V7.1: Se implementa manejo de errores con bloques try/except, pero las excepciones podrían ser más específicas. El manejo genérico (except Exception as e:) dificulta la depuración y el manejo específico de diferentes tipos de errores. Se recomienda usar excepciones específicas como ValueError, TypeError, etc.

■ V7.2: La API REST devuelve códigos HTTP apropiados: 400 para errores de validación, 404 para recursos no encontrados, 500 para errores del servidor. Los códigos son apropiados pero podrían ser más específicos (ej. 422 para errores de validación de formato).

■■ V7.3: El logging de errores está implementado pero es mejorable. Se recomienda logging estructurado con más contexto (módulo, función, parámetros, etc.). Esto está relacionado con el finding CWE-703 (Improper Check or Handling of Exceptional Conditions) en DefectDojo.

■ V7.4: No se exponen detalles técnicos de errores al usuario final. Los mensajes de error son genéricos y no revelan información sensible sobre la implementación interna.

#### *4.4.8. V8: Data Protection*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■ V8.1: Se implementan validaciones defensivas antes de operaciones críticas. Por ejemplo, antes de calcular el IMC se valida que los datos almacenados estén dentro de los límites válidos, protegiendo contra datos corruptos o antiguos.

■■ V8.2: Falta configuración completa de headers de seguridad. Se recomienda implementar todos los headers de seguridad recomendados por OWASP

■■ V8.3: Falta protección completa contra clickjacking. El finding CWE-1021 está pendiente en DefectDojo. Se recomienda implementar X-Frame-Options: DENY y CSP frame-ancestors 'none'

■■ V8.4: Falta configuración adecuada de CORS. Se recomienda configurar CORS con orígenes específicos en lugar de permitir todos los orígenes

#### *4.4.9. V9: Communications*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■■ V9.1: La aplicación actualmente utiliza HTTP. Se recomienda HTTPS para producción para proteger las comunicaciones.

■ V9.2: CORS está configurado correctamente

■ V9.3: Se valida el origen de las peticiones CORS

#### *4.4.10. V10: Malicious Code*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■ V10.1: Se valida toda la entrada de datos para prevenir código malicioso

■ V10.2: Se sanitiza la entrada en el frontend

■■ V10.3: Falta configuración completa de Content-Security-Policy

■ V10.4: La aplicación no permite la carga de archivos desde usuarios

#### *4.4.11. V11: Business Logic*

Estado de cumplimiento: ■ CUMPLE

Explicación del cumplimiento:

■ V11.1: La lógica de negocio está centralizada y bien estructurada en app/config.py

■ V11.2: Se implementan controles de negocio consistentes (validaciones de límites, cálculos)

#### *4.4.12. V12: Files and Resources*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■■ V12.1: Parcialmente aplicable - Solo los endpoints de importación/exportación de DefectDojo manejan archivos

■ V12.2: Se utiliza secure_filename() para nombres de archivo en los endpoints de DefectDojo

#### *4.4.13. V13: API*

Estado de cumplimiento: ■ CUMPLE

Explicación del cumplimiento:

■ V13.1: La API REST está documentada mediante código y sigue estándares RESTful

■ V13.2: Todos los endpoints de la API validan la entrada de datos

■ V13.4: Se manejan errores con códigos HTTP apropiados (400, 404, 500)

#### *4.4.14. V14: Configuration*

Estado de cumplimiento: ■■ PARCIAL

Explicación detallada:

■ V14.1: La configuración está centralizada en app/config.py

■■ V14.2: Falta configuración centralizada de headers de seguridad

■■ V14.3: La configuración de CORS podría estar mejor gestionada

## 5. Análisis WSTG (OWASP Web Security Testing Guide)

### *5.1. Introducción al WSTG*

El OWASP Web Security Testing Guide (WSTG) es una guía completa para probar la seguridad de aplicaciones web y servicios web. Proporciona un marco de mejores prácticas utilizado por profesionales de seguridad y organizaciones en todo el mundo.

Este análisis se basa en los findings de WSTG almacenados en DefectDojo, obtenidos mediante la sincronización bidireccional con el WSTG Tracker.

### *5.2. Estado de los Tests WSTG*

■■ No se encontraron findings de WSTG en DefectDojo.

Esto puede deberse a:

- No se ha ejecutado la sincronización WSTG aún

- No hay tests WSTG configurados en el WSTG Tracker

- Los findings WSTG no se han sincronizado con DefectDojo

## 6. Recomendaciones y Próximos Pasos

### *6.1. Resumen del Estado Actual*

La aplicación implementa buenas prácticas de seguridad en validación de entrada, headers de seguridad y manejo defensivo de datos.

### *6.2. Mejoras Recomendadas*

Para alcanzar el cumplimiento completo del ASVS Nivel 2, se recomienda implementar las siguientes mejoras:

• Validación de tipos numéricos: Implementar validación explícita de NaN e Infinity
• CORS en producción: Restringir CORS a orígenes específicos en producción
• Logging mejorado: Implementar logging estructurado de errores

## 7. Referencias

• OWASP ASVS (página principal)
• OWASP ASVS 4.0.3 en GitHub (versión utilizada en este informe)
• OWASP ASVS 4.0.3 - Categorías de verificación
• OWASP ASVS 5.0 (versión actual)
• OWASP WSTG (Web Security Testing Guide)