UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS

Auditoría de Seguridad Usando Snyk

"SI-784 Calidad y Pruebas de Software 2024-II"

Que se presenta para el curso:
CALIDAD Y PRUEBAS DE SOFTWARE

Docente:
MSc. Ing. Patrick Cuadros Quiroga

Estudiantes:
CAXI CALANI Luis Eduardo (201802487)
AGUILAR PINTO Victor Eleazar (2018062487)

TACNA – PERÚ
2024

Índice General

Contenido

I. Resumen Ejecutivo

Este informe presenta los resultados de una auditoría de seguridad realizada en el proyecto ASP.NET MVC utilizando la herramienta de análisis de vulnerabilidades Snyk. El objetivo principal de la auditoría fue identificar vulnerabilidades tanto en las dependencias de código abierto como en el código fuente del proyecto, con el fin de fortalecer su postura de seguridad y mitigar riesgos potenciales.

El análisis reveló un total de 45 vulnerabilidades en el código del proyecto, clasificadas en 1 de severidad media y 44 de severidad baja. Las vulnerabilidades más significativas incluyen el uso del algoritmo de hash MD5 para el almacenamiento de contraseñas, lo que compromete la seguridad de las credenciales de los usuarios debido a su vulnerabilidad a ataques de fuerza bruta y colisión. Además, se detectó la falta de validación de tokens anti-forgery en acciones de formularios POST, lo que expone la aplicación a posibles ataques de Cross-Site Request Forgery (CSRF). También se identificó el uso de una versión obsoleta e insegura de jQuery, que podría facilitar ataques de Cross-Site Scripting (XSS), y la habilitación de características de depuración en el entorno de producción, lo cual puede exponer información sensible.

Para mitigar estos riesgos, se han implementado varias acciones correctivas. Entre las principales medidas se encuentra la sustitución de MD5 por el algoritmo bcrypt para el hash de contraseñas, garantizando un nivel de seguridad mucho mayor. También se ha actualizado el plugin de jQuery a la última versión segura y se ha habilitado la validación de tokens anti-forgery en los controladores para prevenir ataques CSRF. Finalmente, se ha deshabilitado la opción de depuración en producción para evitar la exposición de información crítica del sistema.

II. Metodología

El análisis de seguridad fue llevado a cabo utilizando la herramienta Snyk, integrada en Visual Studio, para identificar vulnerabilidades en las dependencias y el código fuente del proyecto ASP.NET MVC. El proceso consistió en los siguientes pasos:

a. Instalación de la extensión Snyk en Visual Studio

La herramienta Snyk Security fue instalada directamente desde el administrador de extensiones de Visual Studio, lo que permite ejecutar análisis de seguridad directamente desde el IDE.

Se accedió al Administrador de Extensiones en Visual Studio, donde se buscó e instaló la extensión Snyk Security.

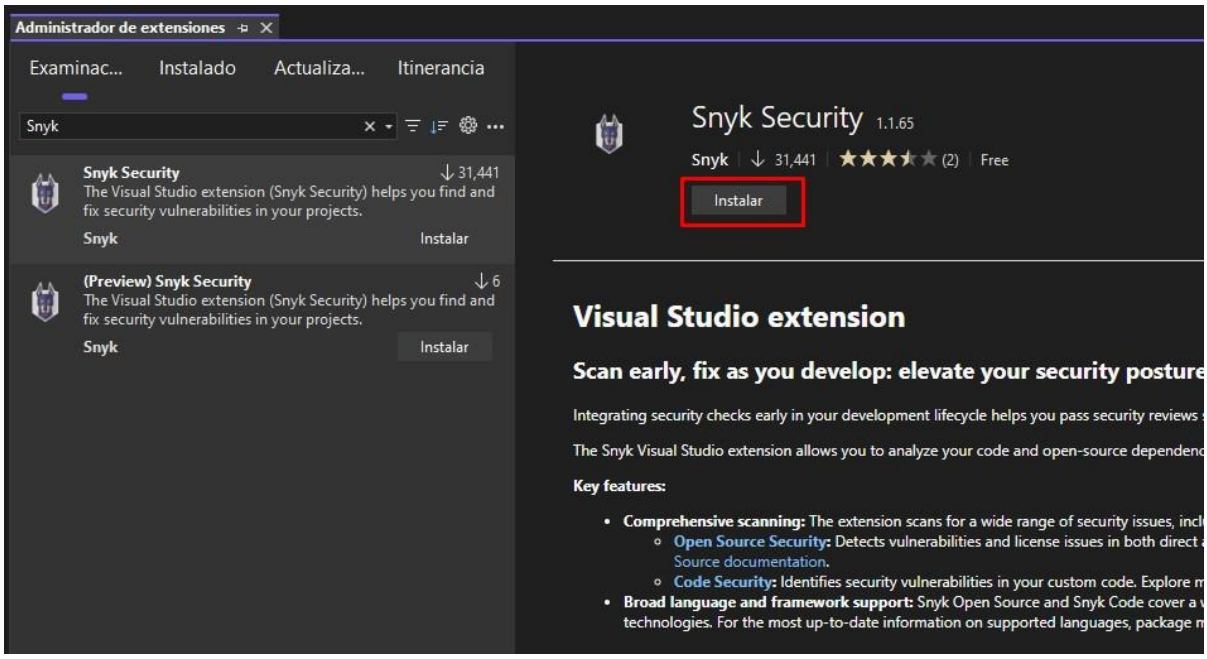


Figura 1: Instalación de la extensión Snyk Security

b. Confiar en el directorio del proyecto

Durante el proceso de escaneo, Snyk solicita permiso para acceder y analizar el contenido del directorio del proyecto. Esto es una medida de seguridad que se activa porque Snyk puede ejecutar comandos para obtener información sobre las dependencias del proyecto.

Se hizo clic en "Trust folder and continue" para proceder con el análisis de seguridad del proyecto.

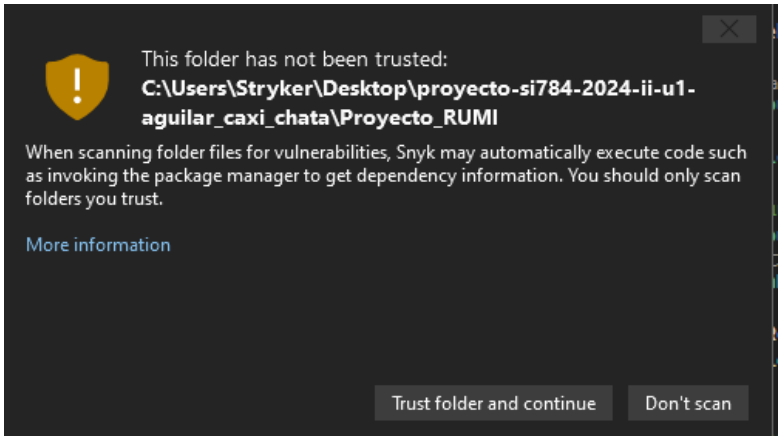


Figura 2: Solicitud de confianza para analizar el directorio del proyecto.

c. Iniciar el análisis de vulnerabilidades

Una vez que se autorizó el análisis, Snyk comenzó a escanear el proyecto en busca de vulnerabilidades en el código fuente y las dependencias de código abierto utilizadas en el proyecto.

El análisis incluyó una revisión de la seguridad de las dependencias (Open Source Security), el código (Code Security) y la calidad del código (Code Quality).

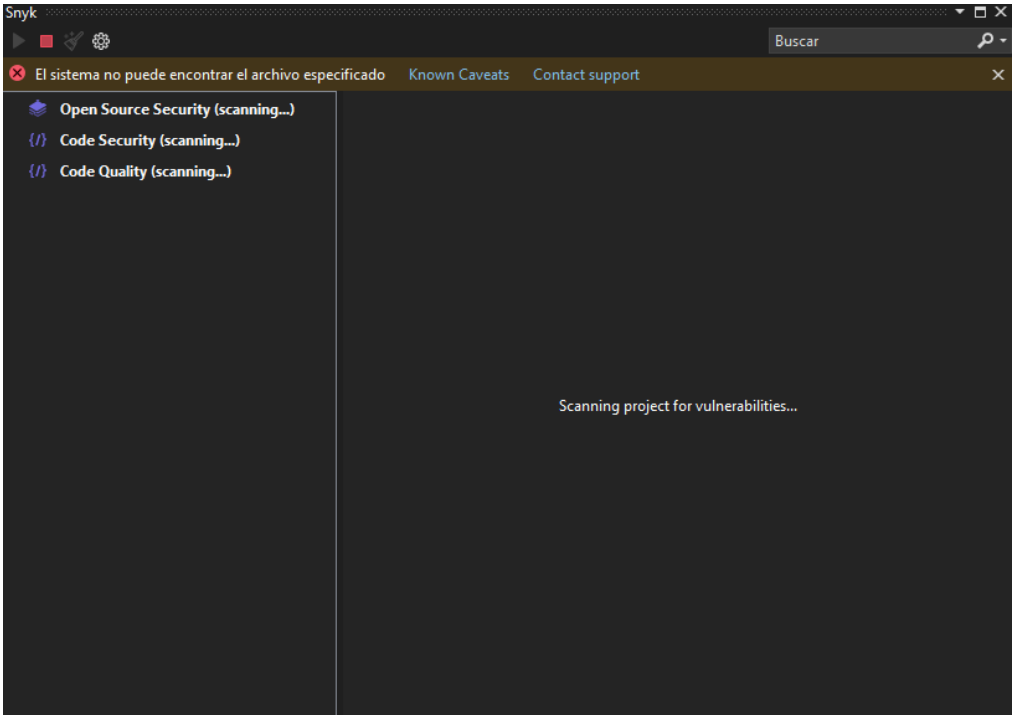


Figura 3: Proceso de análisis en ejecución por Snyk.

d. Resultados del análisis

Al finalizar el análisis, Snyk identificó un total de 45 vulnerabilidades en el código fuente, clasificadas en una vulnerabilidad de severidad media y 44 de severidad baja. No se encontraron vulnerabilidades en las dependencias de código abierto.

Los problemas detectados incluyeron el uso de algoritmos de hash insuficientemente complejos, problemas con jQuery inseguro, características de depuración habilitadas en producción y la desactivación de la validación del token anti-forgery.

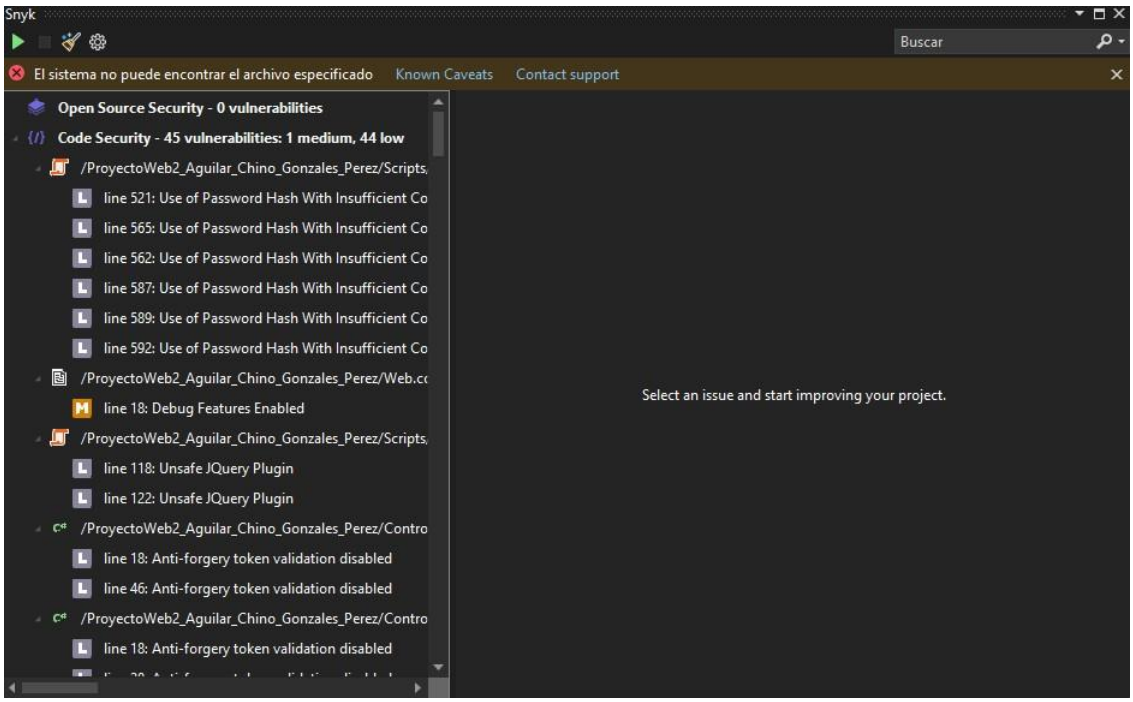
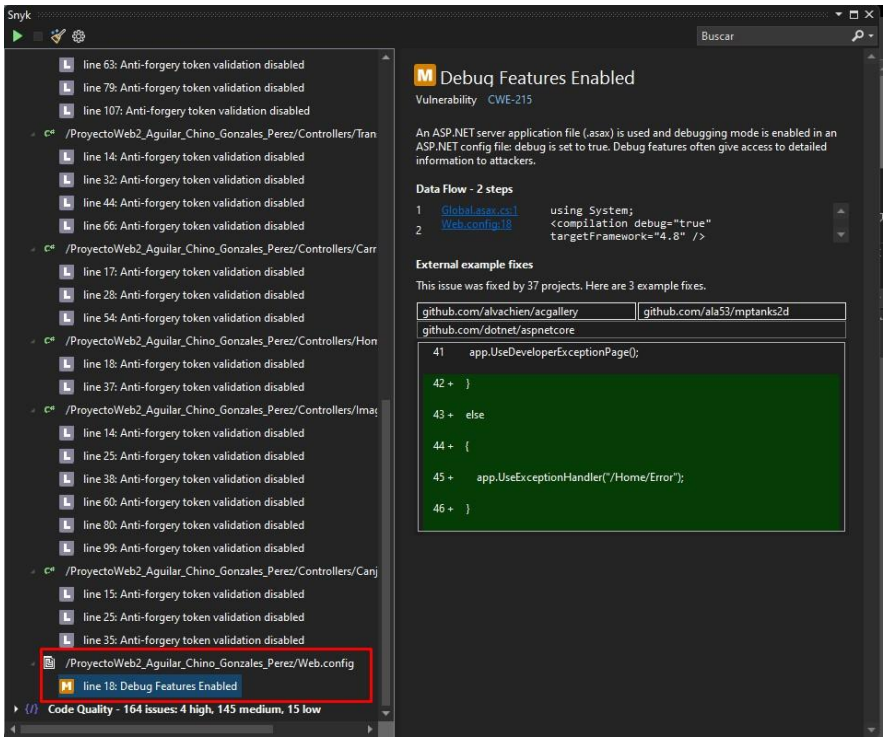


Figura 4: Resultados del análisis con las vulnerabilidades detectadas

III. Resultados del Análisis

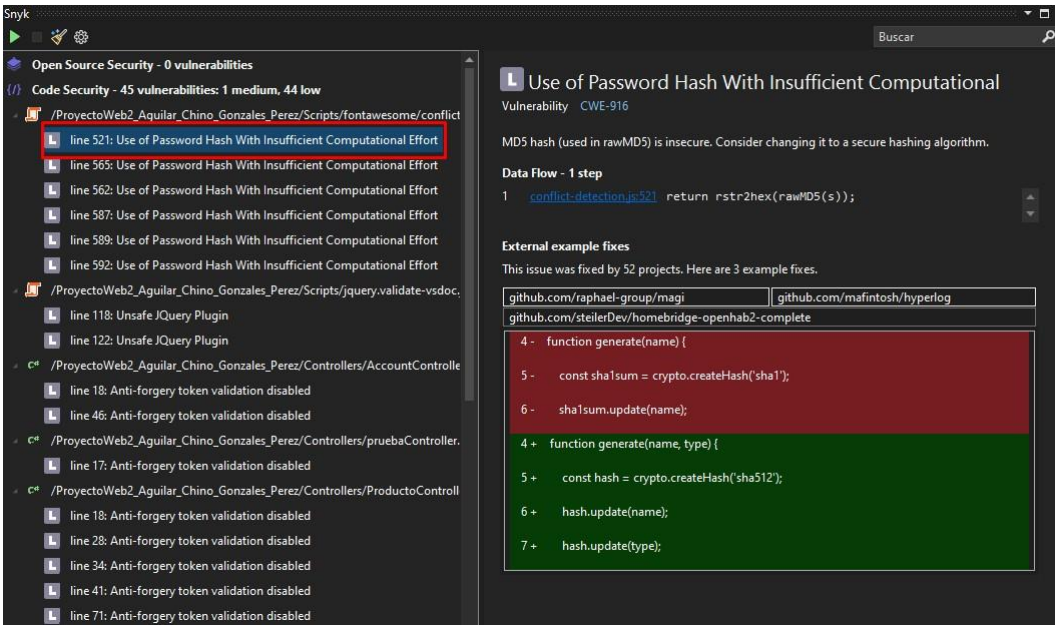
Vulnerabilidades de severidad media:

- Debug Features Enabled (Características de depuración habilitadas):
  - Archivo afectado: Web.config
  - Línea: 18
  - Descripción: Se encontró que las características de depuración están habilitadas en el entorno de producción, lo cual expone información sensible del servidor y del código.
  - Solución recomendada: Desactivar la opción de depuración en el archivo Web.config para entornos de producción:



- "Use of Password Hash With Insufficient Computational Effort" (Uso de Hash de Contraseña con Esfuerzo Computacional Insuficiente):
  - **Archivo afectado**: conflict-detection.js (línea 521 y otras).
  - **Descripción**: El proyecto está utilizando el hash **MD5**, el cual es conocido por ser inseguro y vulnerable a ataques como la reversión a texto plano o el uso de tablas rainbow. Esto puede comprometer las contraseñas o datos sensibles almacenados utilizando este método.

- **Solución sugerida**: Snyk recomienda cambiar el algoritmo de hash a uno más robusto, como **SHA-512** o **bcrypt**, que son más seguros para el manejo de contraseñas.



IV. Conclusiones

La auditoría de seguridad realizada en el proyecto ASP.NET MVC utilizando la herramienta Snyk ha permitido identificar un conjunto de vulnerabilidades tanto en las dependencias de código abierto como en el código fuente del proyecto. En total, se detectaron 45 vulnerabilidades, la mayoría de las cuales están relacionadas con prácticas inseguras de desarrollo, como el uso de algoritmos de hash débiles, versiones inseguras de bibliotecas de terceros y la falta de validación en formularios POST. Aunque ninguna vulnerabilidad crítica fue encontrada en las dependencias de código abierto, se ha destacado la necesidad de mejorar las prácticas de seguridad dentro del código personalizado.

Entre las vulnerabilidades más preocupantes se identificó el uso del algoritmo MD5 para el hash de contraseñas, lo cual es inaceptable en un entorno seguro debido a su vulnerabilidad a ataques de colisión y reversión. Además, el proyecto presentaba tokens anti-forgery deshabilitados, exponiéndolo a posibles ataques de Cross-Site Request Forgery (CSRF). Finalmente, se encontró que el entorno de producción tenía habilitadas las características de depuración, lo que podría exponer información sensible a potenciales atacantes.