https://doi.org/10.22201/dgtic.ctud.2025.3.1.93

Vol. 3, Núm. 1. enero-marzo 2025, págs. 89 - 96

Análisis estático de código fuente de aplicativos en materia de seguridad informática

Angie Aguilar Domínguez

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

angie.aguilar@comunidad.unam.mx

ORCID: 0009-0007-7590-0678

Información del reporte:

Para citar este reporte técnico:

Aguilar Domínguez, A. (2025). Análisis estático de código fuente de aplicativos en materia de seguridad informática. *Cuadernos Técnicos Universitarios de la DGTIC, 3* (1) páginas(89 - 96).

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93

Resumen

El análisis estático de código fuente se realizó para detectar vulnerabilidades conocidas que pudieran llegar a ser explotadas, afectando tanto a aplicativos en entornos web como en dispositivos móviles. La revisión de seguridad permitió llevar a cabo el descubrimiento de fallas de seguridad, accionando la implementación de las correcciones necesarias previo a la publicación en entornos productivos. La Dirección General de Cómputo y de Tecnologías de Información y Comunicación implementó una metodología que combinó el uso de herramientas automatizadas específicas para este fin; éstas trabajaron en conjunto con la verificación manual de los hallazgos para detectar y descartar posibles falsos positivos, además de considerar elementos correspondientes a la calidad del código que pudieran llegar a interferir en la seguridad de la información. Los hallazgos detectados y verificados fueron clasificados de acuerdo con el nivel de criticidad, siguiendo la metodología de calificación de riesgos propuesta por el *Open Web Application Security Project*. Se emitió un reporte técnico con los hallazgos, dirigido al equipo de desarrollo, para su adecuada atención y así solventar mejoras a los aplicativos. Posteriormente, se verificó la adecuada implementación de correcciones. El análisis estático de código fuente permitió detectar y remediar vulnerabilidades conocidas a nivel de código en los aplicativos que fueron analizados, colaborando a mejorar la seguridad de la información que con ellos se maneja.

Palabras clave:

Análisis estático, código fuente, seguridad informática, vulnerabilidades, SAST.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 90 - 96

1. INTRODUCCIÓN

Debido a la gran cantidad de actividades que se realizan hoy en día mediante el uso de distintas herramientas de tecnologías de información, es necesario tener presente las vulnerabilidades existentes en el software que pueden ser explotadas por distintas amenazas, ya que, de acuerdo con Candau (2021), "las vulnerabilidades son consustanciales al software. Todos los años se publican alrededor de 5000 en diferentes tecnologías" (p.13).

Por lo anterior, es importante considerar la seguridad como un elemento esencial del ciclo de vida del desarrollo de software, independientemente del tipo, tamaño, cantidad de usuarios o finalidad de uso de aplicativo del que se trate, incluyéndola en las etapas de diseño e implementación (desarrollo).

Como indican Assal y Chiasson (2018) en la investigación que realizaron sobre la inclusión de seguridad en los proyectos de software:

> …encontramos una gran brecha en las prácticas de seguridad descritas por nuestros participantes en la etapa de diseño. En esta etapa, había equipos en todos los puntos del espectro de priorización de la seguridad; sin embargo, la mayoría de los participantes indicaron que sus equipos no consideraban la seguridad como parte de esta etapa (p. 284).

Así, durante la etapa de implementación, también es necesario tener presente que "la seguridad se considera una responsabilidad del desarrollador durante la implementación, y los participantes explicaron que son conscientes de las vulnerabilidades introducidas por los errores al escribir el código" (Assal y Chiasson, 2018, p. 184).

Dada la necesidad de la realización de pruebas de seguridad sobre aplicativos web y móviles, la Dirección General de Cómputo y de Tecnologías de Información y Comunicación (DGTIC) puso en marcha la realización de pruebas de revisión de seguridad en código fuente previo a la liberación en entornos productivos de los desarrollos realizados dentro de la Universidad.

El objetivo de realizar pruebas de análisis estático de código fuente en materia de seguridad consiste en encontrar posibles vulnerabilidades y fallos en términos de seguridad dentro de la programación, los cuales pueden ser mitigados siguiendo las recomendaciones propuestas antes de la publicación de los aplicativos para su uso por parte de los usuarios finales.

2. DESARROLLO TÉCNICO

De acuerdo con *Open Web Application Security Project* (OWASP), "El análisis de código estático (también conocido como análisis de código fuente) se realiza generalmente como parte de una revisión de código (también conocida como prueba de caja blanca) y se lleva a cabo en la fase de implementación de un ciclo de vida de desarrollo de seguridad (SDL)". (*Static Code Analysis | OWASP Foundation*, s.f.). Dichas pruebas analizan el código (sin estar en ejecución) durante la fase de desarrollo o, en el caso de seguir alguna metodología ágil, cuando se cuenta con un producto mínimo viable.

Para la realización de estas pruebas, se hizo uso de dos herramientas de análisis estático de código fuente; por una parte, una herramienta comercial *on premise* (*Fortify Static Code Analyzer, Fortify SCA*) y, por otra, una herramienta de código abierto *on premise* (*Mobile Security Framework*, *MobSF*).

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 91 - 96

Se hace uso de herramientas automatizadas, ya que, de acuerdo con (Bermejo Higuera et al., 2020, p. 1559): "una de las ventajas más importantes de las herramientas SAST *(Static Application Security Testing*) es que analizan toda la aplicación cubriendo todas las fuentes de entrada. En combinación con una verificación manual de los resultados".

También se hace uso de la metodología OWASP *Risk Rating (OWASP Risk Rating Methodology | OWASP Foundation, s.f.)* para ponderar los hallazgos. Esta metodología permite realizar una estimación del riesgo asociado a la vulnerabilidad detectada en las etapas tempranas del ciclo de vida. Adicionalmente, ofrece dos ventajas importantes: primero, ésta es sencilla de implementar, mientras que, por otro lado, permite personalizar algunos elementos de la ponderación para adecuarse más a las necesidades particulares de cada proyecto, por ejemplo, la priorización al reportar un tipo de hallazgo en específico.

Así, se revisa la totalidad de los archivos del código fuente desarrollado, lo que permite tener presente la seguridad en la etapa de implementación; esto ayudará a que tanto el aplicativo web como el de dispositivos móviles (disponible para dispositivos Android o bien IOS) tome en cuenta la detección de vulnerabilidades en su implementación.

2.1 METODOLOGÍA

La metodología de análisis estático de seguridad de código fuente, propuesta y desarrollada de acuerdo con las necesidades planteadas para llevar a cabo las revisiones, comprende cuatro actividades base: diseño de las pruebas, ejecución de las pruebas, entrega de resultados y verificación de las correcciones.

Adicionalmente, la metodología empleada para la realización de las pruebas considera la aplicación de dos etapas de éstas, las cuales se estructuraron de la siguiente manera:

- Primera etapa. Se realiza el análisis estático de código fuente en materia de seguridad mediante el uso de herramientas automatizadas y verificación manual de los resultados.
- Segunda etapa. Se aplican pruebas de verificación, nuevamente con las herramientas automatizadas y verificación manual, sobre el código que contiene las correcciones implementadas, para verificar la mitigación de los hallazgos detectados en la primera etapa.

Para el diseño de las pruebas, se consideró realizar el análisis del código fuente en materia de seguridad, tomando como referencia las buenas prácticas de codificación segura de OWASP, siguiendo tanto el Top 10 para aplicativos web (*OWASP Top Ten | OWASP Foundation*, *s.f.*), como el Top 10 Mobile (*OWASP Mobile Top 10 | OWASP Foundation*, *s.f.*), para aplicativos que se despliegan en dispositivos móviles.

El análisis se realiza en la totalidad de archivos de código fuente, los cuales son proporcionados por la entidad o dependencia interesada en revisar su aplicativo. Éstos se entregan en un archivo comprimido, lo que permite corroborar que se pueda llevar a cabo la revisión sobre la última versión estable.

La ejecución de las pruebas comprende la realización de los análisis estáticos de caja blanca sobre los archivos que fueron compartidos, los cuales son examinados por las dos herramientas previamente seleccionadas. Posteriormente, se verifica que la salida proporcionada por las herramientas haya finalizado correctamente su ejecución, para con ello, hacer una revisión y depuración de los resultados.

En las imágenes, de la 1 a la 3, puede observarse un ejemplo del resultado de la ejecución de las pruebas. En la Imagen 1, se cuenta con la salida proporcionada por la ejecución de la herramienta *MobSF* para un

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 Vol. 3, Núm. 1. enero-marzo 2025, págs. 92 - 96

aplicativo móvil en su versión para Android, mostrando los permisos declarados para su ejecución y el estatus asignado a cada uno de ellos, así como una breve descripción del permiso.

Imagen 1

Salida proporcionada por la herramienta MobSF al verificar permisos de ejecución

APPLICATION PERMISSIONS

Search:

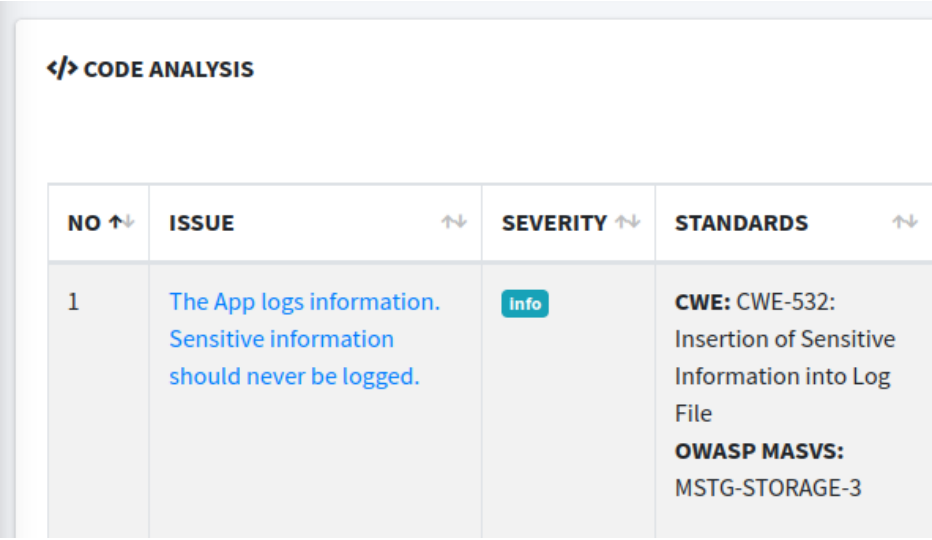
PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.ACCESS_COARSE_LOCATION	dangerous	coarse (network-based) location	Access coarse location sources, such as the mobile network database, to determine an approximate phone location, where available. Malicious applications can use this to determine approximately where you are.
android.permission.ACCESS_FINE_LOCATION	dangerous	fine (GPS) location	Access fine location sources, such as the Global Positioning System on the phone, where available. Malicious applications can use this to determine where you are and may consume additional battery power.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.INTERNET	normal	full Internet	Allows an application to create

En la Imagen 2, se observa la salida proporcionada por la herramienta *MobSF* en la sección donde se analiza el código fuente y el primer hallazgo detectado, el cual proporciona una breve descripción y las referencias a las vulnerabilidades asociadas que se encuentran reportadas en distintas bases de datos. En este ejemplo, se encontró que el aplicativo guarda en bitácora información sensible que no debe ser almacenada ahí.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 Vol. 3, Núm. 1. enero-marzo 2025, págs. 93 - 96

Imagen 2

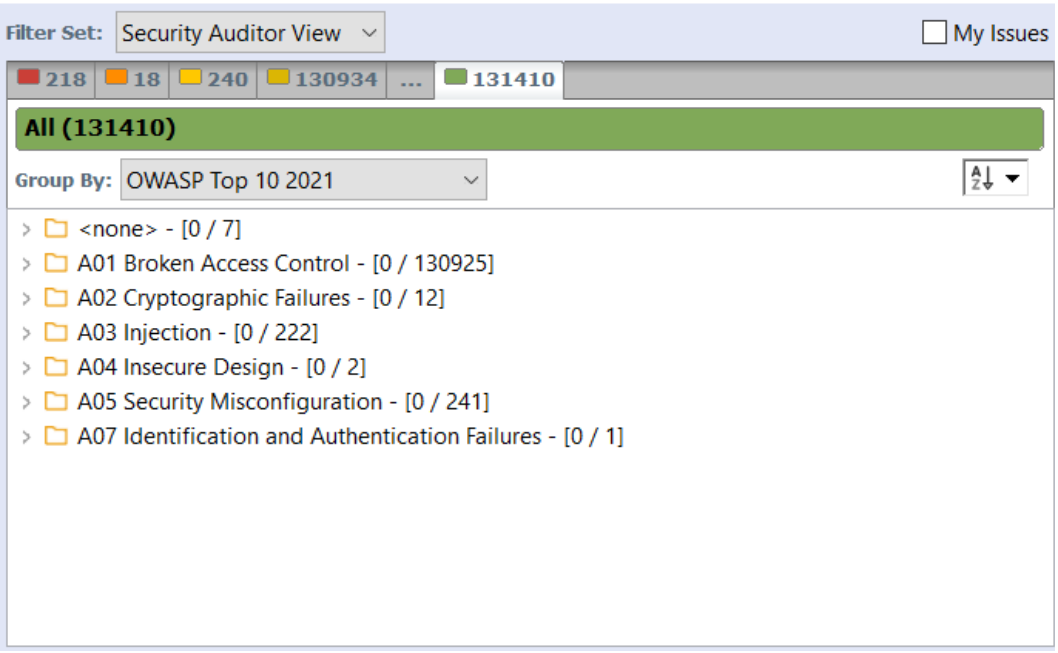
Salida proporcionada por la herramienta MobSF al analizar el código fuente



En la Imagen 3, puede observarse la clasificación de hallazgos realizada por la herramienta *Fortify SCA*, tomando como referencia a OWASP Top 10; los presenta agrupados en carpetas después de haber realizado el análisis del código fuente. En este ejemplo, para el caso de Inyección, existen 222 hallazgos.

Imagen 3

Salida proporcionada por la herramienta Fortify SCA de acuerdo con el Top 10 de OWASP



https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 94 - 96

Con los resultados previamente obtenidos, es necesario que se lleve a cabo la verificación manual de la salida proporcionada por las herramientas para descartar falsos positivos, ya que, como indica de Barros (2022), "se sabe que las herramientas SAST producen falsos positivos (encuentran una vulnerabilidad donde no existe y, por lo tanto, no requieren el esfuerzo de los desarrolladores para solucionarla)" (p. 2).

Al descartar falsos positivos, se corrobora que los resultados informados al equipo de desarrollo no incluyan hallazgos erróneos y contengan sólo aquellos que representan un problema de seguridad si no son mitigados, evitando compartir información errónea.

Para la entrega de resultados, se redacta un reporte técnico con la finalidad de que el equipo responsable del aplicativo lo analice y considere la implementación de las medidas recomendadas o bien, realice una justificación que sustente la razón que impide la implementación de la corrección.

Así, el reporte entregable contiene los hallazgos, agrupados, clasificados y priorizados, acompañados de un conjunto de recomendaciones para solucionarlos, las referencias a las fuentes de información para profundizar la recomendación emitida y la calificación asignada al hallazgo de acuerdo con la metodología para ponderar conocida como OWASP *Risk Rating (OWASP Risk Rating Methodology | OWASP Foundation, s.f.).*

Las recomendaciones emitidas para solventar los hallazgos toman como referencia guías y mejores prácticas de la industria. Un ejemplo de esto son aquellas que han sido emitidas por el *National Institute of Standards and Technology* (NIST), por *MITRE Corporation*, o bien, por las distintas bases de datos de vulnerabilidades existentes.

Finalmente, en caso de que sea solicitado, se revisa la atención de las observaciones durante la segunda etapa de pruebas y se emite un segundo reporte actualizado, el cual contiene los hallazgos persistentes (aquellos que no fueron solventados), así como otros nuevos que hayan surgido durante los últimos cambios que requieran ser corregidos.

3. RESULTADOS

Como resultado del diseño y puesta en marcha de estas pruebas, se cuenta con un catálogo de los hallazgos y recomendaciones de seguridad asociadas con información unificada, el cual permitió estandarizar la información de los hallazgos que se incluye en la emisión de los reportes.

Durante el periodo de realización de las pruebas, se emitieron un total de 12 reportes, los cuales, en su mayoría, fueron revisiones de primera etapa, lo que indica que los equipos de desarrollo que solicitaron la revisión de su aplicativo aún presentan dificultades para incluir prácticas de seguridad embebidas en el ciclo de vida de sus proyectos de software.

Los hallazgos más frecuentes hacen referencia a los siguientes elementos:

- Permisos de ubicación en aplicativos para dispositivos móviles cuando no es necesario hacer uso de ellos, ya que comparten la ubicación en tiempo real del dispositivo.
- Ausencia de canales de comunicación cifrados para la transmisión de datos, lo que permitiría a un usuario malintencionado intentar interceptar datos enviados en texto claro.

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 95 - 96

- Uso de bibliotecas y software de terceros que se encuentran desactualizados, o bien, que cuentan con vulnerabilidades conocidas, lo que se podría emplear como vector de ataque contra el aplicativo.

La realización de análisis estático de código fuente sirvió como un apoyo a las entidades y dependencias de la Universidad para mejorar el aseguramiento de datos sensibles empleados por los distintos aplicativos, por ejemplo: datos de colaboradores de una dependencia, o bien, datos de estudiantes de alguna facultad.

La entrega de los resultados de la revisión permitió a los responsables de los aplicativos tomar las medidas necesarias para mejorar o incluir la seguridad (en caso de no haberla considerado) en sus proyectos de software, en la etapa de implementación y, sobre todo, a tomar en cuenta y atender las vulnerabilidades detectadas, así como a reducir el riesgo de que sean explotadas.

4. CONCLUSIONES

Como indican Assal y Chiasson (2018): "Las mejores prácticas a menudo se ignoran, simplemente porque el cumplimiento aumentaría la carga sobre el equipo; en su opinión, los equipos están haciendo un equilibrio razonable entre costo y beneficio" (p. 292).

Por lo anterior, la seguridad en el desarrollo de software debe considerarse a lo largo de las diferentes etapas, buscando aligerar la carga de su inclusión, y considerarse desde las primeras etapas de diseño de la solución que se implementará, con lo que se busca proteger no sólo la información que se maneja, sino a los usuarios y la infraestructura de TI involucrada.

La realización de las pruebas de seguridad en el código fuente de los aplicativos permitió encontrar vulnerabilidades, las cuales, al corregirse, mejoraron la seguridad en el manejo de la información utilizada, mitigando riesgos que pueden ser detectados por este tipo de pruebas antes de ser explotados.

De acuerdo con De Barros Reigada (2021): "el análisis estático tiene algunas limitaciones debido a la falta de información sobre el tiempo de ejecución. En las herramientas SAST, esta falta de información puede afectar gravemente a la calidad de los hallazgos, es decir, a su precisión" (p. 2)

Por lo anterior, es importante tomar en cuenta que el análisis estático de código fuente es sólo uno de los elementos que pueden ayudar a mejorar la seguridad de los aplicativos, ya que debe acompañarse de una revisión dinámica del aplicativo, así como la realización de análisis más completos como la aplicación de pruebas de penetración o *pentest* y la implementación de mejores prácticas sobre configuraciones para el robustecimiento de la infraestructura de tecnologías de información involucrada en su despliegue.

Finalmente, la revisión de seguridad también fungió como un facilitador para la publicación en entornos productivos de los aplicativos. Para el caso de los aplicativos web, disminuyendo la cantidad de hallazgos realizados en el análisis dinámico de seguridad y, en el caso de las aplicaciones para dispositivos móviles, facilitando su publicación en la tienda de aplicaciones correspondiente (App Store para el caso de Apple y Google Play Store para el caso de Android).

https://doi.org/10.22201/dgtic.ctud.2025.3.1.93 **Vol. 3, Núm. 1.** enero-marzo 2025, págs. 96 - 96

REFERENCIAS

Assal, H., & Chiasson, S. (2018). *Security in the Software Development Lifecycle*. USENIX Association. https://www.usenix.org/system/files/conference/soups2018/soups2018-assal.pdf

Bermejo Higuera, J. R., et. al (2020). Benchmarking Approach to Compare Web Applications Static Analysis Tools Detecting OWASP Top Ten Security Vulnerabilities. *Computers, Materials & Continua*, *64*(3), 1555–1577. https://doi.org/10.32604/cmc.2020.010885

Candau, J. (2021). Ciberseguridad: Evolución y tendencias. Bie3: *Boletín IEEE*, 23, 460–494. https://dialnet.unirioja.es/servlet/articulo?codigo=8175398

De Barros Reigada, A. (2021). Generic SAST tool Comparer. *Universidade Do Minho*. https://repositorium.sdum.uminho.pt/bitstream/1822/84173/1/Alexandra%20de%20Barros%20Reigada.pdf

OWASP Mobile Top 10 | OWASP Foundation. (s.f.). https://owasp.org/www-project-mobile-top-10/

OWASP Risk Rating Methodology | *OWASP Foundation*. (s.f.). https://owasp.org/www-community/OWASP_Risk_Rating_Methodology

OWASP Top Ten | OWASP Foundation. (s.f.). https://owasp.org/www-project-top-ten/

Static Code Analysis | *OWASP Foundation*. (s.f.). https://owasp.org/www-community/controls/Static_Code_Analysis