Revista Científica Multidisciplinar G-ner@ndo ISSN: 2806-5905

Análisis comparativo de técnicas de protección de software para prevenir el acceso no autorizado y la distribución ilegal

Comparative analysis of software protection techniques to prevent unauthorized access and illegal distribution

Jean Carlos Zambrano Navarrete & Miguel Rodríguez Véliz

DIMENSIÓN CIENTÍFICA

Enero - junio, V°7 - N°1; 2026

Recibido: 13-05-2026
Aceptado: 18-05-2026
Publicado: 22-05-2026

PAIS

- Ecuador, Portoviejo
- Ecuador, Portoviejo

INSTITUCION

- Universidad Técnica de Manabí
- Universidad Técnica de Manabí

CORREO:

✉ jzambrano6765@utm.edu.ec
✉ miguel.rodriguez@utm.edu.ec

ORCID:

https://orcid.org/0009-0001-4068-8888
https://orcid.org/0000-0003-4474-3853

FORMATO DE CITA APA.

Zambrano, J. & Rodríguez, M. (2026). Análisis comparativo de técnicas de protección de software para prevenir el acceso no autorizado y la distribución ilegal. *Revista G-ner@ndo, V°7 (N°1). Pág. 5483 – 5514.*

Resumen

Proteger el software hoy en día no fue solo una cuestión técnica, es casi una carrera constante contra nuevas formas de ataque. Bajo esa idea, este estudio revisó 29 trabajos publicados entre 2020 y 2025, seleccionados cuidadosamente mediante Parsifal, con el propósito de entender qué soluciones se están aplicando realmente para enfrentar estos desafíos. Las investigaciones analizadas abordaron técnicas como la ofuscación de código, los mecanismos de licenciamiento y activación, el watermarking y el fingerprinting. Los resultados evidencian una tendencia hacia el uso de estrategias híbridas y complementarias, las cuales ofrecen un mejor equilibrio entre seguridad, rendimiento y compatibilidad frente a soluciones aisladas. Asimismo, se identificaron limitaciones relacionadas con la falta de estandarización de métricas de evaluación y la integración entre técnicas. Este trabajo aportó una visión estructurada al estado del arte y planteó orientaciones para futuras investigaciones en entornos digitales.

Palabras clave: protección de software; acceso no autorizado; distribución ilegal; seguridad del software.

Abstract

Protecting software today is no longer just a technical matter; it's almost a constant race against new forms of attack. With this in mind, this study reviewed 29 papers published between 2020 and 2025, carefully selected using Parsifal, to understand which solutions are actually being implemented to address these challenges. The analyzed research covered techniques such as code obfuscation, licensing and activation mechanisms, watermarking, and fingerprinting. The results show a trend toward the use of hybrid and complementary strategies, which offer a better balance between security, performance, and compatibility compared to isolated solutions. Limitations related to the lack of standardized evaluation metrics and the integration between techniques were also identified. This work provided a structured overview of the state of the art and offered guidance for future research in digital environments.

Keywords: software protection; unauthorized access; illegal distribution; software security.

Introducción

Hoy en día, el desarrollo de software ha crecido de forma acelerada, impulsado por la transformación digital y la incorporación de tecnologías en casi todos los sectores, desde la industria y la salud hasta el transporte y los servicios en la nube. Este avance ha traído importantes beneficios en términos de innovación y automatización, pero también ha expuesto al software a nuevos riesgos. Problemas como el acceso no autorizado, la ingeniería inversa, la manipulación del código y la distribución ilegal se han vuelto cada vez más comunes, afectando directamente la propiedad intelectual y la sostenibilidad económica de quienes desarrollan software (Ahire & Abraham, 2022; Hataba et al., 2022).

Frente a esta perspectiva, la protección del software ha cobrado una relevancia especial dentro de la ingeniería de software. A diferencia de la seguridad del software, que se enfoca en detectar vulnerabilidades y prevenir ataques externos, la protección del software busca resguardar el producto en sí, evitando su copia, análisis o modificación sin autorización (Basile et al., 2023; Broeck et al., 2022). Esta diferencia no es menor, ya que gran parte de las propuestas en la literatura se centran en la seguridad en general, dejando en segundo plano aspectos clave como la piratería o la defensa de la propiedad intelectual.

Las técnicas tradicionales de seguridad se quedan cortas, no basta con proteger desde afuera cuando quien ataca puede observar, modificar e incluso manipular el comportamiento del software desde adentro. Ante esta realidad, han surgido mecanismos más específicos que buscan dificultar el análisis y la alteración del código (Basile et al., 2023). Todo esto ha empujado la investigación hacia estrategias más complejas, orientadas en hacer que el software sea más resistente frente a este tipo de amenazas, donde el control del entorno ya no está del lado del desarrollador. Cuando se habla de proteger software, hay una técnica que aparece casi siempre, la ofuscación de código. La idea es

bastante clara, transformar el código para que sea difícil de entender, pero sin cambiar lo que hace. Funciona igual, pero para quien intenta analizarlo, todo se vuelve mucho más complicado.

Por eso ha sido tan estudiada, sobre todo por su capacidad para frenar la ingeniería inversa. A partir de ahí, han surgido varias propuestas interesantes. (Li et al. 2022) plantean modificar la lógica interna del programa mediante ofuscación semántica. (Xiong et al. 2025) proponen una integración más profunda para resistir ataques avanzados. Y (Al-Hakimi et al. 2020) combinan distintas técnicas en un enfoque híbrido para aumentar la protección. Además, investigaciones recientes han explorado la ofuscación basada en virtualización, la cual transforma el código en una representación ejecutada por una máquina virtual personalizada, aumentando significativamente la dificultad de análisis. En este sentido, (Suk & Lee, 2020) proponen una técnica denominada Virtual Code Folding, que mejora la efectividad de la ofuscación al reducir patrones detectables en el código. De manera complementaria, (Hataba et al., 2022) aplican técnicas de ofuscación en entornos de computación en la nube para sistemas vehiculares autónomos, demostrando la aplicabilidad de estas soluciones en escenarios distribuidos.

Otro caso importante de investigación corresponde a las técnicas de gestión de derechos digitales (DRM), las cuales buscan controlar el acceso y la distribución del software mediante mecanismos de cifrado y autenticación. (Mishra et al., 2021) presentan un marco basado en caos para la distribución segura de contenido digital, destacando la importancia de integrar mecanismos robustos de protección en entornos donde la distribución ilegal es un problema recurrente. De igual manera, el watermarking digital ha sido utilizado como una técnica para garantizar la autenticidad y la propiedad del software o de los datos asociados. (Anand et al., 2024) proponen un enfoque basado en aprendizaje profundo para la protección de registros médicos, demostrando que estas técnicas pueden

aplicarse en contextos donde la integridad de la información es crítica. Sin embargo, algunos estudios han evidenciado que estas técnicas son más comunes en el ámbito multimedia que en la protección directa de software, lo que limita su aplicabilidad en ciertos escenarios.

Por otro lado, el fingerprinting de software ha emergido como una técnica relevante para identificar instancias únicas de programas y rastrear su distribución ilegal. En este sentido, (Alrabaee et al., 2022) realizan una revisión exhaustiva de las técnicas de fingerprinting, proporcionando una taxonomía detallada y destacando su importancia en la protección de la propiedad intelectual. Estas técnicas complementan otras soluciones al permitir la trazabilidad del software incluso después de su distribución. Junto a estas estrategias, también han cobrado fuerza las técnicas enfocadas en dificultar el análisis dinámico del software. Entre ellas destacan el anti-debugging y el anti-tracing, diseñadas para evitar que un programa pueda ejecutarse en entornos controlados por un atacante. En otras palabras, buscan cortar el problema desde el momento en que alguien intenta observar el comportamiento del software en tiempo real.

(Norby et al. 2025), comparan distintas técnicas de anti-debugging en varios sistemas operativos, mientras que (Yue et al. 2024) presentan mecanismos más avanzados orientados a frenar técnicas de trazado incluso a nivel de hardware. Todo esto deja ver una dirección bastante clara, las estrategias de protección están adaptándose a ataques cada vez más complejos.

A su vez, ha aumentado el número de investigaciones que no buscan proteger directamente el software, sino analizar o incluso romper los mecanismos de protección. Aquí entran en juego los planteamientos basados en inteligencia artificial para detectar código ofuscado (Greco et al., 2024) o técnicas de análisis de similitud (Zhang et al., 2022).

Junto a estas propuestas, también aparecen las técnicas de tipo distribuido o contractual, como el DRM, el watermarking y el fingerprinting, que centran su atención en la protección del software durante su distribución y uso. A diferencia de otros métodos, aquí no solo se trata de evitar accesos no autorizados, sino de mantener cierto control incluso después de que el software ya ha sido entregado.

Métodos y Materiales

La presente investigación se desarrolló mediante una Revisión Sistemática de la Literatura (SLR), con el objetivo de identificar, analizar y sintetizar las principales técnicas de protección de software orientadas a prevenir el acceso no autorizado y la distribución ilegal. Este estudio metodológico permite garantizar un proceso riguroso, transparente y reproducible en la selección y evaluación de estudios relevantes dentro del área de investigación.

Para la ejecución de la revisión sistemática, se siguieron lineamientos ampliamente aceptados en la literatura de ingeniería de software, adaptando las fases clásicas del proceso: planificación, búsqueda de literatura, selección de estudios, evaluación de calidad y extracción de datos. Asimismo, se utilizó la herramienta Parsifal como apoyo para la gestión del proceso, facilitando la organización de los estudios, la aplicación de criterios de inclusión y exclusión, así como la evaluación sistemática de la calidad de los artículos seleccionados.

Preguntas de investigación

Con el fin de orientar el proceso de revisión y delimitar el alcance del estudio, se definieron las siguientes preguntas de investigación:

P1: ¿Cuáles son las técnicas de protección de software que han demostrado mayor efectividad para prevenir el acceso no autorizado y la distribución ilegal entre 2020 y 2025?

P2: ¿Qué tipos de técnicas han sido más abordadas en la literatura científica reciente y cómo se clasifican según su enfoque (estático, dinámico, contractual o distribuido)?

P3: ¿Qué ventajas y limitaciones presentan las técnicas de protección de software más estudiadas, considerando criterios como rendimiento, seguridad, sostenibilidad y facilidad de integración?

P4: ¿Cómo se han aplicado estas técnicas en dominios específicos como la computación en la nube, inteligencia artificial, sistemas embebidos y contratos inteligentes?

Criterios de Inclusión y Exclusión.

Tabla 1. *Con el objetivo de garantizar la calidad y pertinencia de los estudios analizados, se establecieron los siguientes criterios:*

Criterios de inclusión	**Criterios de exclusión**
Artículos científicos publicados en idioma inglés o español.	Artículos duplicados o con datos insuficientes.
Artículos revisados por pares.	Documentos publicados antes del año 2020.
Estudios que analicen técnicas específicas de protección de software.	Estudios que no traten de manera específica técnicas de protección de software.
Estudios que incluyan evaluación o comparación de técnicas.	Trabajos que no presentan evidencia académica verificable.
Publicaciones científicas comprendidas en el periodo 2020–2025.	

Estrategia de Búsqueda de Literatura

Para la identificación de estudios relevantes, se definió una estrategia de búsqueda basada en el uso de cadenas estructuradas aplicadas en diversas bases de datos científicas. La cadena principal utilizada fue la siguiente:

Tabla 2. *Estrategia de búsqueda y fuentes de información*.

Elemento	Descripción
Estrategia de búsqueda	Se utilizó una cadena de búsqueda estructurada para identificar estudios relevantes relacionados con técnicas de protección de software.
Cadena de búsqueda	("illegal distribution" OR "software piracy" OR "software protection" OR "software security" OR "unauthorized access") AND ("obfuscation" OR "code obfuscation" OR "software watermarking" OR "watermarking" OR "digital rights management" OR "DRM" OR "software fingerprinting" OR "fingerprinting" OR "anti reverse engineering" OR "anti tamper") AND ("techniques" OR "methods" OR "approaches" OR "review" OR "comparison")
Objetivo de la búsqueda	Identificar estudios científicos sobre técnicas de protección de software como ofuscación, DRM, watermarking y fingerprinting.
Bases de datos	IEEE Xplore; Scopus; ScienceDirect; SciELO
Ajustes realizados	Se adaptó la cadena de búsqueda según los requisitos de cada base de datos.
Periodo de búsqueda	2020 – 2025
Tipo de documentos	Artículos científicos revisados por pares

Proceso de selección de estudios

La selección de artículos se realizó en varias etapas, y cada una fue afinando un poco más el resultado. Al inicio, se obtuvo un conjunto amplio aplicando la cadena de

búsqueda en las bases de datos seleccionadas. Luego vino un primer filtro: revisar títulos y resúmenes para descartar lo que no cumplía con los criterios.

Después, se pasó a una fase más detallada: la lectura completa de los artículos preseleccionados. Este paso permitió identificar cuáles realmente eran relevantes, reduciendo bastante el número de estudios hasta quedarse con un grupo final sólido.

Finalmente, todos los artículos seleccionados se organizaron en Parsifal, lo que facilitó tanto su clasificación como el análisis posterior.

Evaluación de calidad

Para asegurar la validez del análisis, los artículos seleccionados fueron evaluados en función de varios criterios:

- Claridad en la descripción de la técnica propuesta.
- Definición de la metodología empleada.
- Presentación de resultados claros y medibles.
- Comparación con otras técnicas existentes.
- Publicación en fuentes científicas confiables.

Cada estudio fue evaluado utilizando una escala cualitativa, lo que permitió identificar aquellos trabajos con mayor rigor científico para el análisis.

Extracción y análisis de datos

Una vez completadas las etapas de selección y evaluación, se pasó a la extracción de la información más relevante de cada estudio. Para mantener consistencia, se definieron previamente varias variables de análisis:

- Tipo de técnica de protección.
- Clasificación (estática, dinámica, contractual o distribuida).
- Objetivo de la técnica.
- Contexto de aplicación.
- Ventajas y limitaciones.

La información recopilada fue organizada en tablas comparativas, lo que permitió identificar patrones, tendencias y relaciones entre las diferentes técnicas. Asimismo, se realizó un análisis cualitativo con el fin de responder las preguntas de investigación planteadas.

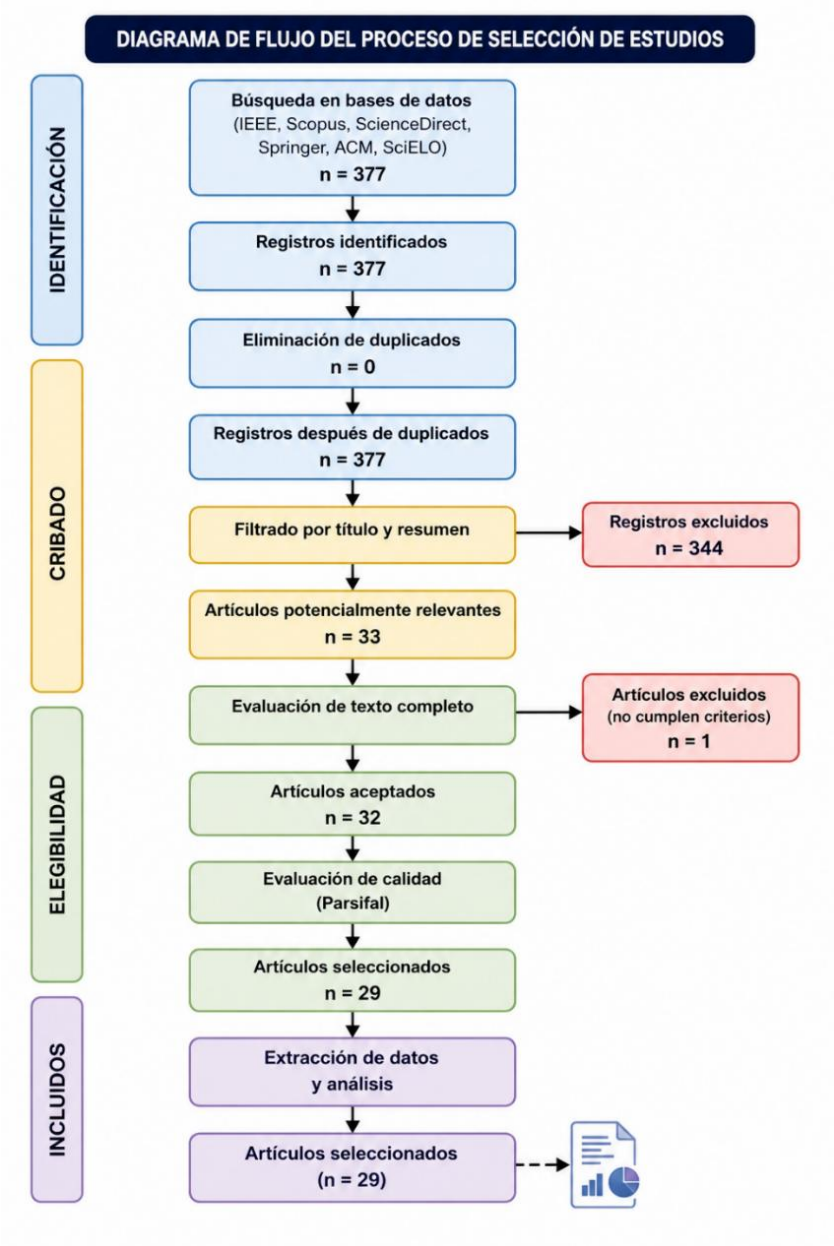
Diagrama de flujo del proceso

Para representar de forma clara y ordenada el proceso de selección de estudios, se elaboró un diagrama de flujo basado en el modelo PRISMA, en total, se identificaron 377 artículos en distintas bases de datos. Al revisar duplicados, no se encontró ninguno. Luego, se aplicó un primer filtro revisando títulos y resúmenes, lo que permitió descartar 344 estudios que no cumplían con los criterios.

Después, se pasó a la lectura completa de los artículos restantes. En esta etapa se seleccionaron 32 estudios. Finalmente, tras aplicar la evaluación de calidad con Parsifal, se definieron 29 artículos como base para el análisis final. Todo este recorrido se presenta en

la Figura 1, donde se pueden ver claramente las fases de identificación, selección, elegibilidad e inclusión.

Figura 1. *Diagrama de flujo del proceso de selección de estudios basado en el modelo PRISMA.*



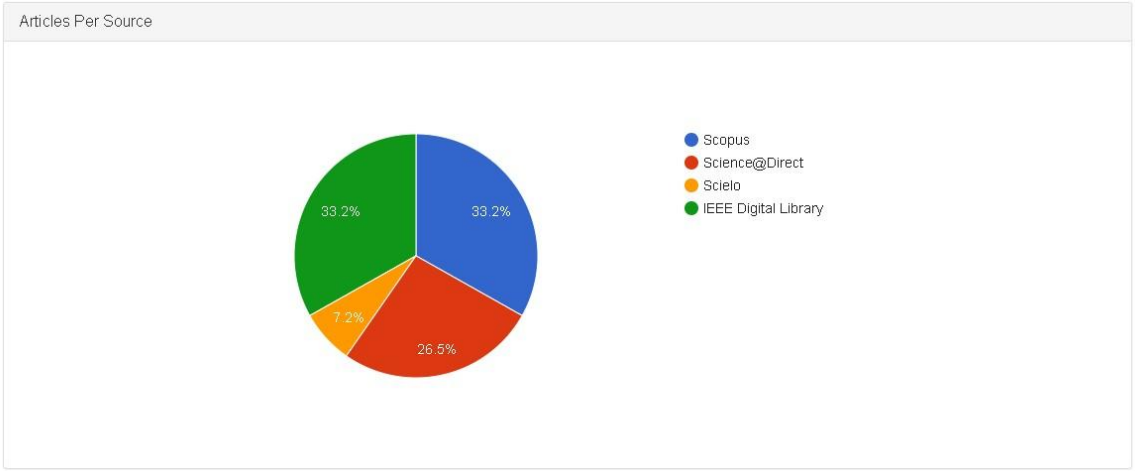
Análisis de resultados

En este apartado se analizan los 29 artículos seleccionados tras el proceso de revisión sistemática. Más que verlos de forma individual, la idea es entender qué patrones aparecen cuando se observan en conjunto.

El análisis se centra en responder las preguntas de investigación, considerando la distribución de los estudios, su origen, su evolución en el tiempo y la forma en que se clasifican las distintas técnicas de protección de software.

Distribución de artículos por fuente

Figura 2. *Distribución de artículos por fuente*

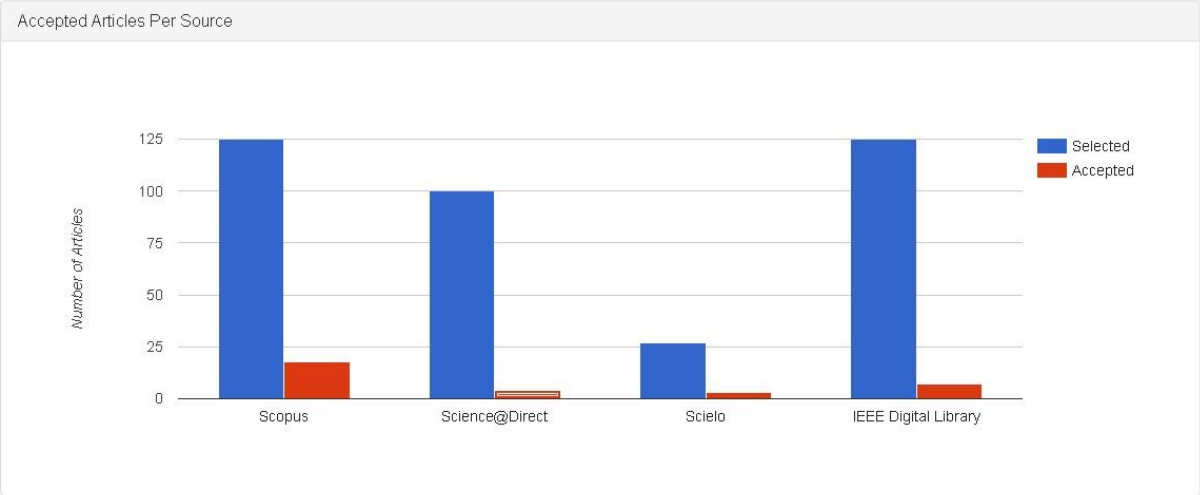


La Figura 2 muestra la distribución porcentual de los artículos recopilados en función de las bases de datos utilizadas. Se observa que las fuentes Scopus e IEEE Digital Library representan la mayor proporción de estudios, cada una con aproximadamente el 33.2% del total. Por su parte, ScienceDirect aporta el 26.5%, mientras que SciELO presenta una menor participación con el 7.2%.

Estos resultados evidencian que la mayor parte de la producción científica en el área de protección de software se concentra en bases de datos de alto impacto, lo cual respalda la calidad y relevancia de los estudios incluidos en el análisis.

Artículos seleccionados y aceptados por fuente

Figura 3. *Artículos seleccionados y aceptados por fuente*.



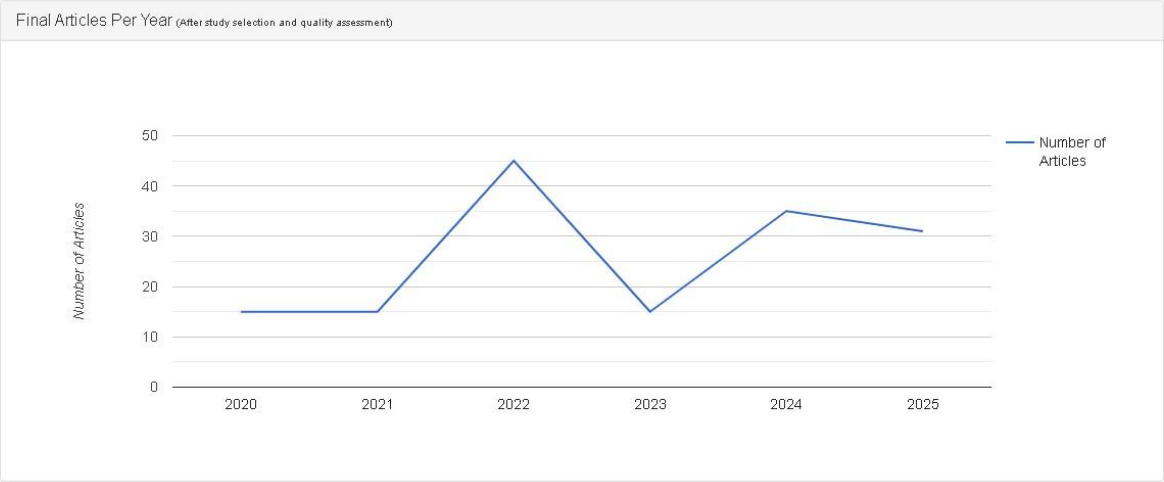
En la Figura 3 se presenta la relación entre el número de artículos identificados inicialmente y aquellos que fueron finalmente aceptados tras la aplicación de los criterios de inclusión, exclusión y evaluación de calidad.

Se observa que bases de datos como Scopus e IEEE Digital Library concentran una mayor cantidad de estudios en la fase inicial. Aun así, el número de artículos aceptados se reduce de forma considerable. Esta diferencia responde al proceso de filtrado, que permitió descartar trabajos que no cumplían con los requisitos metodológicos o que no estaban directamente enfocados en la protección de software.

Por otro lado, bases como SciELO muestran un volumen menor tanto de artículos seleccionados como aceptados, lo que sugiere una menor producción científica en este campo dentro de dicha fuente.

Distribución de artículos por año

Figura 4. *Distribución de artículos por año*.



La Figura 4 muestra cómo ha cambiado el número de artículos seleccionados entre 2020 y 2025. En 2020 y 2021, la cantidad se mantiene bastante estable. Luego, en 2022, se observa un aumento claro, alcanzando el punto más alto.

Después de ese pico, en 2023 hay una caída en el número de publicaciones. Sin embargo, en 2024 se recupera y en 2025 se mantiene en un nivel más estable. Esta evolución deja ver que el interés por la protección de software ha ido creciendo, sobre todo a partir de 2022.

Si se analizan las figuras en conjunto, aparecen algunas ideas clave. Por un lado, la mayoría de los estudios provienen de bases de datos reconocidas, lo que aporta

confianza en la calidad de la información. Por otro, el número de artículos se reduce bastante durante la selección, lo que confirma que el proceso fue exigente.

Respuestas a las preguntas de investigación

Tabla 3. *Técnicas de protección de software más efectivas*

Técnica	Descripción	Nivel de efectividad	Evidencia
Ofuscación de código	Transforma el código para dificultar su comprensión	Alto	Amplio uso en la literatura
Virtualización	Ejecuta código en máquina virtual personalizada	Muy alto	Alta resistencia a ingeniería inversa
Anti-debugging	Bloquea herramientas de depuración	Alto	Protege en ejecución
Anti-fuzzing	Evita pruebas automatizadas	Alto	Protege en ejecución
DRM	Controla acceso y uso	Medio–Alto	Reduce vulnerabilidades
Watermarking	Inserta marcas digitales	Medio	Protección de propiedad
Fingerprinting	Identifica copias únicas	Medio	Trazabilidad

RQ1: ¿Cuáles son las técnicas de protección de software que han demostrado mayor efectividad para prevenir el acceso no autorizado y la distribución ilegal entre 2020 y 2025?

Los resultados del análisis evidencian que las propuestas de protección de software más efectivas se enfocaron en incrementar la dificultad del análisis y manipulación del software, restringir el uso no autorizado y fortalecer la protección de la propiedad intelectual. La literatura científica reciente muestra una tendencia hacia mecanismos cada vez más

robustos, orientados a reducir riesgos relacionados con la ingeniería inversa, la copia no autorizada y la distribución ilegal del software.

La ofuscación de código se presenta como uno de los mecanismos más importantes dentro de la literatura científica reciente. Su principal objetivo consiste en transformar la estructura interna del software para dificultar su comprensión, análisis y modificación sin alterar su comportamiento funcional. Este tipo de protección complica los procesos de ingeniería inversa mediante cambios en el flujo del programa, modificación de instrucciones y ocultamiento de la lógica interna del código. (Rodríguez Véliz et al. 2024) validaron experimentalmente un modelo de protección basado en la ofuscación del código, demostrando que este mecanismo incrementa la resistencia frente al análisis no autorizado y fortalece la protección de la propiedad intelectual.

Por otra parte, los mecanismos DRM (Digital Rights Management) demostraron ser soluciones relevantes para controlar el acceso y la distribución del software mediante sistemas de autenticación, cifrado y licenciamiento digital. Estas propuestas buscan restringir la reproducción, copia y uso indebido del software una vez distribuido. (Mishra et al. 2021) propusieron un marco de distribución segura basado en DRM y algoritmos caóticos, evidenciando que este tipo de mecanismos mejora el control sobre el contenido digital y reduce los riesgos asociados a la distribución ilegal. Aunque estas soluciones no impiden completamente la ingeniería inversa, sí permiten mantener un mayor control sobre la utilización del software en entornos comerciales y plataformas distribuidas.

Asimismo, el watermarking es una técnica que inserta marcas digitales invisibles dentro del software o de los datos relacionados, permitiendo demostrar la propiedad y detectar modificaciones no autorizadas. (Anand et al. 2024) desarrollaron un enfoque basado en aprendizaje profundo para proteger registros médicos mediante watermarking,

demostrando que este tipo de solución puede aplicarse en escenarios donde la integridad de la información es fundamental. Los estudios revisados muestran que el watermarking resulta especialmente útil en contextos donde se necesita garantizar autenticidad y trazabilidad del contenido digital.

Por último, el fingerprinting emergió como un mecanismo efectivo para rastrear y diferenciar copias específicas de software distribuidas a distintos usuarios. Su funcionamiento consiste en insertar identificadores únicos dentro de cada instancia del programa, permitiendo identificar el origen de copias filtradas o distribuidas ilegalmente. (Alrabaee et al. 2022), en su revisión sobre fingerprinting de código binario, destacaron que estas soluciones representan una herramienta importante para la protección de la propiedad intelectual y la detección de distribución no autorizada, especialmente en entornos donde el software puede ser replicado fácilmente.

Tabla 4. *Clasificación de técnicas de protección*

Categoría	Técnicas	Momento de acción	Objetivo principal
Estáticas	Ofuscación (clásica, semántica, híbrida)	Antes de ejecución	Dificultar comprensión
Dinámicas	Anti-debugging, anti-tracing, anti-fuzzing, virtualización	Durante ejecución	Bloquear análisis
Contractuales	DRM, licencias	Uso/consumo	Control de acceso
Distribuidas	Watermarking, fingerprinting	Post-distribución	Trazabilidad/propiedad

RQ2: ¿Qué tipos de técnicas han sido más abordadas en la literatura científica reciente, y cómo se clasifican según su enfoque?

El análisis de los estudios seleccionados muestra que la literatura científica reciente ha centrado sus investigaciones en el desarrollo de mecanismos de protección capaces de reducir riesgos asociados al acceso no autorizado y la distribución ilegal del software.

Dentro de estas propuestas, las técnicas estáticas se caracterizan por aplicar transformaciones directamente sobre el código antes de que el software sea ejecutado. Su principal finalidad consiste en dificultar la comprensión de la lógica interna del programa y aumentar la complejidad del análisis estático sin modificar el comportamiento funcional del software. Dentro de esta categoría, la literatura revisada evidencia una alta presencia de investigaciones relacionadas con la ofuscación clásica, la ofuscación semántica y la ofuscación híbrida, mecanismos ampliamente utilizados para complicar procesos de ingeniería inversa y ocultar la estructura real del código fuente.

Las técnicas dinámicas se enfocan en proteger el software durante su ejecución, actuando directamente sobre el comportamiento del programa en tiempo real. Estas soluciones buscan detectar, bloquear o dificultar herramientas de análisis utilizadas por atacantes, evitando procesos de depuración, monitoreo o análisis automatizado. Entre las técnicas más abordadas dentro de esta categoría destacan el anti-debugging, anti-tracing, anti-fuzzing y los mecanismos de virtualización, los cuales incrementan significativamente la dificultad del análisis dinámico y fortalecen la resistencia frente a ataques avanzados.

Las técnicas contractuales se caracterizan por regular el acceso y uso del software mediante mecanismos de autenticación, licenciamiento y control de permisos. Su enfoque principal no se centra en ocultar el código, sino en restringir la utilización no autorizada del software y mantener control sobre su distribución. En este contexto, la literatura científica reciente destaca principalmente los sistemas DRM y los mecanismos de licenciamiento digital, ampliamente utilizados en entornos comerciales para proteger la propiedad intelectual y limitar la reproducción ilegal del software.

Las técnicas distribuidas se enfocan en mantener mecanismos de protección una vez que el software ha sido distribuido. Estas propuestas buscan garantizar autenticidad,

trazabilidad y protección de la propiedad intelectual mediante la identificación y seguimiento de copias específicas del software. Dentro de esta categoría, los estudios revisados resaltan especialmente el watermarking y el fingerprinting, técnicas que permiten insertar identificadores digitales invisibles para rastrear distribuciones no autorizadas y validar la autenticidad del contenido protegido.

Tabla 5. *Evaluación de técnicas de protección de software según rendimiento, seguridad, sostenibilidad, integración, ventajas y limitaciones*

Articulo	**Técnica**	**Criterios de Evaluación Funcional**	**Ventajas/ Limitaciones**
Secure cloud model for intellectual privacy protection of arithmetic expressions in source codes using data obfuscation techniques(Ahire & Abraham, 2022)	Ofuscación	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Alta protección Limitación: Impacto en rendimiento
Role based access control using identity and broadcast based encryption for securing cloud data(Saxena & Alam, 2022)	Control de acceso	Rendimiento: Alto Seguridad: Medio Integración: Alto	Ventaja: Fácil implementación Limitación: Protección limitada
Authenticating and securing healthcare records: A deep learning-based zero watermarking approach(Anand et al., 2024)	Watermarking	Rendimiento:Alto Seguridad: Medio Integración: Alto	Ventaja: Protección de datos Limitación: No evita acceso directo
Chaos-based content distribution framework for digital rights management system(Mishra et al., 2021)	DRM	Rendimiento: Medio Seguridad: Medio Integración: Medio	Ventaja: Control de distribución Limitación: Dependencia de infraestructura
Enhancing resilience in IoT cybersecurity: the roles of	Ofuscación IoT	Rendimiento: Medio Seguridad: Alto	Ventaja: Protección en entornos

Articulo	Técnica	Criterios de Evaluación Funcional	Ventajas/ Limitaciones
obfuscation and diversification techniques(Rauti & Laato, 2024)		Integración: Medio	Limitación: Complejidad
Enhanced obfuscation for software protection in autonomous vehicular cloud computing platforms(Hataba et al., 2022)	Ofuscación vehicular	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Seguridad en sistemas críticos Limitación: Difícil integración
COOPS: A code obfuscation method based on obscuring program semantics(Li et al., 2022)	Ofuscación semántica	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Alta resistencia Limitación: Complejidad de mantenimiento
Robust 3D mesh zero-watermarking based on spherical coordinate and skewness measurement(Lee et al., 2021)	Watermarking 3D	Rendimiento: Alto Seguridad: Medio Integración: Medio	Ventaja: Integridad de datos Limitación: Alcance limitado
Obfuscated code is identifiable by a token-based code clone detection technique(Akram et al., 2022)	Detección de ofuscación	Rendimiento: Alto Seguridad: Bajo Integración: Alto	Ventaja: Mejor analisis Limitación: No protege
A survey of binary code fingerprinting approaches: taxonomy, methodologies, and features(Alrabaee et al., 2022)	Fingerprinting	Rendimiento: Alto Seguridad: Medio Integración: Alto	Ventaja: Trazabilidad Limitación: No bloquea ataques
Code obfuscation based on deep integration(Xiong et al., 2025)	Ofuscación avanzada	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Alta seguridad Limitación: Complejidad
Unleashing the power of pseudo-code for binary code similarity analysis(Zhang et al., 2022)	Análisis de código	Rendimiento: Alto Seguridad: Bajo Integración: Alto	Ventaja: Eficencia Limitación: No protege

Articulo	Técnica	Criterios de Evaluación Funcional	Ventajas/ Limitaciones
ImageShield: a responsibility-to-person blind watermarking mechanism for image datasets protection(Tang et al., 2025)	Watermarking IA	Rendimiento: Medio Seguridad: Medio Integración: Medio	Ventaja: Innovación Limitación: Aplicación limitada
Automated Intel SGX integration for enhanced application security(Regano & Canavese, 2024)	SGX	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Seguridad Limitación: Complejo
CatchFuzz: Reliable active anti-fuzzing techniques against coverage-guided fuzzer(Kim & Lee, 2024)	Anti-fuzzing	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Reduce ataques automatizados Limitación: Afecta pruebas
Internal interface diversification as a method against malware(Rauti et al., 2021)	Diversificación	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Resistencia a ataques Limitación: Complejidad
Question–Answer Methodology for Vulnerable Source Code Review via Prototype-Based Model-Agnostic Meta-Learning(Corona-Fraga et al., 2025)	ML seguridad	Rendimiento: Alto Seguridad: Bajo Integración: Alto	Ventaja: Automatización Limitación: No protege
Enhancing Analog IC Security Using Randomized Obfuscation Circuits(Chaudhuri et al., 2025)	obfuscation	Rendimiento: Bajo Seguridad: Alto Integración: Bajo	Ventaja: Alta seguridad Limitación: Difícil implementación
Flexible software protection(Broeck et al., 2022)	Protección flexible	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Adaptabilidad Limitación: Complejidad
Design, implementation, and automation of a risk management	Gestión riesgo MATE	Rendimiento: Medio Seguridad: Alto	Ventaja: Enfoque estratégico

Articulo	Técnica	Criterios de Evaluación Funcional	Ventajas/ Limitaciones
approach for man-at-the-End software protection(Basile et al., 2023)		Integración: Medio	Limitación: Dependencia del contexto
M-PIVAD - Virtual memory based approach against non-control data attacks(Dileesh & Shanthi, 2020)	Seguridad memoria	Rendimiento: Medio Seguridad: Medio Integración: Medio	Ventaja: Protección interna Limitación: Limitado alcance
Hybrid Obfuscation Technique to Protect Source Code From Prohibited Software Reverse Engineering(Al-Hakimi et al., 2020)	Ofuscación híbrida	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Mayor robustez Limitación: Overhead
TOP: A Combined Logical and Physical Obfuscation Method for Securing Networks-on-Chip Against Reverse Engineering Attacks(Hashemi et al., 2025)	Ofuscación NoC	Rendimiento: Bajo Seguridad: Alto Integración: Bajo	Ventaja: Seguridad alta Limitación: No software puro
VCF: Virtual Code Folding to Enhance Virtualization Obfuscation(Suk & Lee, 2020)	Virtualización	Rendimiento: Bajo Seguridad: Muy alto Integración: Bajo	Ventaja: Maxima protección Limitación: Alto costo
Enabling Obfuscation Detection in Binary Software through eXplainable AI(Greco et al., 2024)	Ofuscación detección IA	Rendimiento: Alto Seguridad: Bajo Integración: Alto	Ventaja: Precisión Limitación: No protege
Validación de un modelo de protección basado en la ofuscación del código(Rodríguez Véliz, 2024)	Ofuscación validada	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Validación experimental Limitación: limitado
Armor: Protecting Software Against Hardware Tracing Techniques(Yue et al., 2024)	Anti-tracing	Rendimiento: Bajo Seguridad: Muy alto Integración: Bajo	Ventaja: Protección avanzada Limitación: Complejidad

Articulo	Técnica	Criterios de Evaluación Funcional	Ventajas/ Limitaciones
Practical Anti-Fuzzing Techniques With Performance Optimization(Zhou & Wang, 2023)	Anti-fuzzing	Rendimiento: Medio Seguridad: Alto Integración: Bajo	Ventaja: Protección dinámica Limitación: Impacto rendimiento
Measurement of Anti-Debugging Techniques on the Windows and Linux Operating Systems for the Intel x86_64 Architecture(Norby et al., 2025)	Anti-debugging	Rendimiento: Medio Seguridad: Alto Integración: Medio	Ventaja: Protección en ejecución Limitación: Puede evadirse

RQ3. ¿Qué ventajas y limitaciones presentan las técnicas de protección de software más estudiadas, considerando criterios como rendimiento, sostenibilidad y facilidad de integración?

A partir del análisis detallado de la Tabla 5, se observa que las técnicas de protección de software más estudiadas presentan un comportamiento diferenciado en función de los criterios de rendimiento, sostenibilidad y facilidad de integración, lo que evidencia la existencia de compromisos importantes entre estos factores. Por un lado, técnicas como la ofuscación de código, la virtualización y los mecanismos dinámicos de anti-debugging, anti-fuzzing y anti-tracing destacan por ofrecer niveles altos e incluso muy altos de seguridad, ya que incrementan significativamente la dificultad del análisis estático y dinámico, dificultando procesos como la ingeniería inversa o la manipulación del software. Sin embargo, estas ventajas vienen acompañadas de limitaciones relevantes: suelen generar un impacto moderado o alto en el rendimiento debido al procesamiento adicional que introducen, presentan una baja o media facilidad de integración al requerir modificaciones profundas en la arquitectura del sistema, y además implican mayores

desafíos de sostenibilidad, ya que su mantenimiento, actualización y adaptación a nuevos entornos resulta complejo y costoso en el tiempo.

Mientras que, otras técnicas como los mecanismos de control de acceso, el DRM, el watermarking y el fingerprinting muestran un comportamiento más equilibrado desde el punto de vista operativo. Estas soluciones destacan por su alto o medio rendimiento, así como por su mayor facilidad de integración en sistemas existentes, lo que las hace más viables en contextos reales de implementación. Asimismo, presentan una mejor sostenibilidad, dado que su mantenimiento es menos complejo y su aplicación no requiere transformaciones profundas del código. No obstante, su principal limitación radica en el nivel de protección que ofrecen, el cual suele ser medio o limitado, ya que no actúan directamente sobre la estructura interna del software ni bloquean de forma efectiva el acceso no autorizado, sino que se orientan principalmente al control de uso, la gestión de licencias o la trazabilidad del software una vez distribuido.

Además, se identifican técnicas emergentes y complementarias, como aquellas basadas en inteligencia artificial o en análisis de código, que aportan ventajas en términos de automatización, eficiencia y facilidad de integración. Estas soluciones suelen presentar un alto rendimiento y una implementación relativamente sencilla, contribuyendo a mejorar procesos de detección, análisis o evaluación de seguridad. Sin embargo, su papel dentro de la protección del software es limitado, ya que no constituyen mecanismos de defensa directa, sino herramientas de apoyo que complementan otras técnicas más robustas.

RQ4. ¿Cómo se han aplicado estas técnicas en dominios específicos como la computación en la nube, inteligencia artificial, sistemas embebidos y contratos inteligentes?

Los resultados del análisis de los estudios incluidos en la revisión, se observa que la aplicación de las técnicas de protección de software varía significativamente según el dominio, adaptándose a las necesidades y riesgos específicos de cada entorno.

En el caso de la computación en la nube, las estrategias se centran principalmente en el control de acceso y la protección de datos en entornos compartidos, donde múltiples usuarios interactúan con los mismos recursos. Aquí predominan técnicas como la ofuscación de código y el DRM, que permiten restringir el uso del software y proteger la información frente a accesos no autorizados.

En la inteligencia artificial, la protección no solo se enfoca en el código, sino también en los modelos y los datos, que representan activos críticos. En este contexto, técnicas como el watermarking basado en aprendizaje profundo han ganado relevancia, ya que permiten insertar marcas invisibles en modelos o datasets para garantizar su autenticidad y detectar usos indebidos. Además, se utilizan herramientas basadas en inteligencia artificial para analizar código, detectar patrones de ofuscación o identificar vulnerabilidades, aunque estas actúan más como mecanismos de apoyo que como protección directa.

En los sistemas embebidos, la protección del software está fuertemente condicionada por las capacidades limitadas del hardware, lo que obliga a tomar decisiones más cuidadosas. A diferencia de otros entornos, aquí no es viable implementar soluciones complejas o pesadas, ya que podrían afectar el funcionamiento del dispositivo. Por ello, se recurre a técnicas más eficientes en consumo de recursos, como formas simplificadas de ofuscación, mecanismos ligeros de virtualización o soluciones apoyadas en el propio hardware. Estas estrategias buscan dificultar el acceso o la manipulación del software sin comprometer el rendimiento. El desafío principal en este tipo de sistemas consiste en

encontrar un balance adecuado entre proteger el software y mantener la eficiencia operativa, especialmente en dispositivos donde cada recurso cuenta.

En el caso de los contratos inteligentes y las tecnologías basadas en blockchain, la protección del software adapta a una solución distinta al tradicional. En lugar de centrarse en ocultar o dificultar el acceso al código, se prioriza la confiabilidad del sistema mediante mecanismos que aseguran que cada operación sea transparente y verificable. La blockchain permite registrar todas las transacciones de forma permanente e inalterable, lo que facilita no solo el seguimiento del uso del software, sino también la validación de su integridad. De esta manera, la seguridad se apoya en la imposibilidad de modificar la información registrada y en la capacidad de auditar cada acción, generando confianza entre las partes. Este enfoque representa una forma diferente de proteger el software, donde el énfasis no está en esconder, sino en garantizar que cualquier intento de manipulación sea detectable y prácticamente inviable.

Discusión

La protección de software ha mostrado una evolución hacia propuestas más elaboradas, en las que ya no basta con aplicar una única técnica. El análisis de los estudios seleccionados evidencia que las soluciones actuales tienden a combinar distintos mecanismos, buscando responder de forma más efectiva a amenazas cada vez más complejas, como la ingeniería inversa, el acceso no autorizado y la distribución ilegal.

En este escenario, la ofuscación de código continúa ocupando un lugar relevante dentro de la literatura. A pesar del desarrollo de nuevas alternativas, sigue siendo ampliamente utilizada debido a que permite dificultar la comprensión del software sin afectar su comportamiento. Esta característica la convierte en una opción práctica y adaptable, lo

que explica su uso frecuente como parte de estrategias de protección más amplias (Ahire & Abraham, 2022; Al-Hakimi et al., 2020).

Aunque la literatura también evidencia que las técnicas tradicionales de ofuscación presentan limitaciones frente a herramientas modernas de análisis. Esto ha impulsado el desarrollo de enfoques más avanzados, como la ofuscación basada en virtualización, que transforma la lógica del programa en representaciones más complejas, aumentando la resistencia frente a ataques (Suk & Lee, 2020). De forma complementaria, técnicas como la ofuscación semántica y los enfoques híbridos han demostrado mejorar la protección al combinar diferentes transformaciones, haciendo más difícil la reconstrucción del código original (Li et al., 2022).

En cuanto a las técnicas dinámicas, los mecanismos de anti-debugging y anti-fuzzing han ganado relevancia al dificultar el análisis del software durante su ejecución. Estas soluciones reducen la capacidad de los atacantes para detectar vulnerabilidades, aunque no eliminan completamente el riesgo, ya que pueden ser superadas mediante técnicas más avanzadas (Zhou & Wang, 2023). Este comportamiento pone de manifiesto que ninguna técnica, por sí sola, resulta completamente efectiva, lo que refuerza la necesidad de adoptar enfoques combinados y adaptativos (Broeck et al., 2022).

Por otra parte, hay técnicas que no buscan bloquear el acceso directamente, sino mantener el control sobre el software después de que ya ha sido distribuido. Aquí entran el watermarking y el fingerprinting, que permiten identificar y rastrear el software en caso de uso indebido (Alrabaee et al., 2022; Anand et al., 2024).

Algo parecido ocurre con los sistemas DRM. Se usan bastante para controlar cómo se utiliza y se distribuye el software, sobre todo en entornos comerciales. Eso sí, su

efectividad no es igual en todos los casos; depende mucho de cómo y dónde se implementen (Mishra et al., 2021).

También se nota algo interesante, cada vez se están incorporando más tecnologías emergentes, como la inteligencia artificial y blockchain. En el caso de la inteligencia artificial, su papel se centra en analizar código y detectar patrones, lo que ayuda a identificar técnicas de protección y posibles fallos (Greco et al., 2024). Eso sí, no protege directamente el software, sino que funciona más como una herramienta de apoyo.

Conclusiones

A partir del análisis sistemático de la literatura científica comprendida entre 2020 y 2025, se deduce que la protección de software ha evolucionado de ser una implementación técnica aislada a consolidarse como una disciplina de defensa adaptativa. La investigación evidencia que, en el escenario tecnológico actual, la eficacia real no reside en la adopción de una tecnología única, sino en la implementación de estrategias híbridas y complementarias que logran mitigar el compromiso crítico entre la seguridad, el rendimiento y la facilidad de integración.

Por un lado, las técnicas de alta complejidad técnica, como la virtualización de código, ofrece una resistencia muy alta a la ingeniería inversa y el anti-debugging, actúan como barreras críticas durante la ejecución para bloquear el análisis dinámico. Sin embargo, su implementación conlleva un alto costo computacional y una integración compleja que puede comprometer el rendimiento en sistemas con recursos limitados, como los dispositivos embebidos.

Por otro lado, los mecanismos de control post distribución, como el DRM, el watermarking y el fingerprinting, se consolidan como herramientas esenciales para la

sostenibilidad económica, permitiendo la trazabilidad y la gestión de licencias en entornos comerciales. Aunque su capacidad para impedir el acceso directo al código es menor, su facilidad de integración y bajo impacto en el rendimiento los hacen indispensables para proteger la propiedad intelectual una vez que el software sale del control directo del desarrollador.

En conclusión, uno de los principales desafíos de esta disciplina continúa siendo la ausencia de métricas estandarizadas que permitan evaluar y comparar objetivamente la efectividad de las diferentes soluciones propuestas en la literatura. Esta limitación dificulta establecer criterios comunes para medir seguridad, rendimiento y resistencia frente a amenazas cada vez más automatizadas y sofisticadas. Por ello, futuras investigaciones deberían enfocarse en la creación de modelos de evaluación más uniformes, así como en el desarrollo de técnicas de protección adaptativas que respondan de manera eficiente a la evolución constante de los entornos digitales.

Referencias bibliográficas

Ahire, P., & Abraham, J. (2022). Secure cloud model for intellectual privacy protection of arithmetic expressions in source codes using data obfuscation techniques. En Theoretical Computer Science (Vol. 922, pp. 131-149). https://doi.org/10.1016/j.tcs.2022.04.018

Akram, J., Vasan, D., & Luo, P. (2022). Obfuscated code is identifiable by a token-based code clone detection technique. En International Journal of Information and Computer Security (Vol. 19, Números 3-4, pp. 254-273). https://doi.org/10.1504/ijics.2022.127132

Al-Hakimi, A. M. H., Sultan, A. B. M., Abdul Ghani, A. A., Ali, N. M., & Admodisastro, N. I. (2020). Hybrid Obfuscation Technique to Protect Source Code From Prohibited Software Reverse Engineering. IEEE Access, 8, 187326-187342. https://doi.org/10.1109/ACCESS.2020.3028428

Alrabaee, S., Debbabi, M., & Wang, L. (2022). A Survey of Binary Code Fingerprinting Approaches: Taxonomy, Methodologies, and Features. En ACM Computing Surveys (Vol. 55, Número 1). https://doi.org/10.1145/3486860

Anand, A., Bedi, J., Aggarwal, A., Khan, M. A., & Rida, I. (2024). Authenticating and securing healthcare records: A deep learning-based zero watermarking approach. En Image and Vision Computing (Vol. 145). https://doi.org/10.1016/j.imavis.2024.104975

Basile, C., Sutter, B. D., Canavese, D., Regano, L., & Coppens, B. (2023). Design, implementation, and automation of a risk management approach for man-at-the-End software protection. Computers & Security, 132, 103321. https://doi.org/10.1016/j.cose.2023.103321

Broeck, J. V. den, Coppens, B., & Sutter, B. D. (2022). Flexible software protection. Computers & Security, 116, 102636. https://doi.org/10.1016/j.cose.2022.102636

Chaudhuri, J., Bhattacharya, M., & Chakrabarty, K. (2025). Enhancing Analog IC Security Using Randomized Obfuscation Circuits. En IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Vol. 44, Número 3, pp. 867-881). https://doi.org/10.1109/TCAD.2024.3466810

Corona-Fraga, P., Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, L. K., Perez-Meana, H., Portillo-Portillo, J., Olivares-Mercado, J., & García Villalba, L. J. (2025). Question–Answer Methodology for Vulnerable Source Code Review via Prototype-Based Model-Agnostic Meta-Learning. En Future Internet (Vol. 17, Número 1). https://doi.org/10.3390/fi17010033

Dileesh, E. D., & Shanthi, A. P. (2020). M-PIVAD - Virtual memory based approach against non-control data attacks. Computers & Security, 95, 101834. https://doi.org/10.1016/j.cose.2020.101834

Greco, C., Ianni, M., Guzzo, A., & Fortino, G. (2024). Enabling Obfuscation Detection in Binary Software through eXplainable AI. IEEE Transactions on Emerging Topics

in Computing, 1-12. https://doi.org/10.1109/TETC.2024.3439884

Hashemi, M., Mohammadi, S., & Carlson, T. E. (2025). TOP: A Combined Logical and Physical Obfuscation Method for Securing Networks-on-Chip Against Reverse Engineering Attacks. IEEE Access, 13, 133438-133456. https://doi.org/10.1109/ACCESS.2025.3590967

Hataba, M., Sherif, A., & Elkhouly, R. (2022). Enhanced Obfuscation for Software Protection in Autonomous Vehicular Cloud Computing Platforms. En IEEE Access (Vol. 10, pp. 33943-33953). https://doi.org/10.1109/ACCESS.2022.3159249

Kim, H. Y., & Lee, D. H. (2024). CatchFuzz: Reliable active anti-fuzzing techniques against coverage-guided fuzzer. En Computers and Security (Vol. 143). https://doi.org/10.1016/j.cose.2024.103904

Lee, J.-S., Liu, C., Chen, Y.-C., Hung, W.-C., & Li, B. (2021). Robust 3D mesh zero-watermarking based on spherical coordinate and Skewness measurement. En Multimedia Tools and Applications (Vol. 80, Número 17, pp. 25757-25772). https://doi.org/10.1007/s11042-021-10878-0

Li, Y., Kang, F., Shu, H., Xiong, X., Sha, Z., & Sui, Z. (2022). COOPS: A Code Obfuscation Method Based on Obscuring Program Semantics. En Security and Communication Networks (Vol. 2022). https://doi.org/10.1155/2022/6903370

Mishra, D., Obaidat, M. S., Rana, S., Dharminder, D., Mishra, A., & Sadoun, B. (2021). Chaos-Based Content Distribution Framework for Digital Rights Management System. En IEEE Systems Journal (Vol. 15, Número 1, pp. 570-576). https://doi.org/10.1109/JSYST.2020.2977929

Norby, A., Rimal, B. P., & Brizendine, B. (2025). Measurement of Anti-Debugging Techniques on the Windows and Linux Operating Systems for the Intel x86_64 Architecture. IEEE Access, 13, 46568-46583. https://doi.org/10.1109/ACCESS.2025.3550009

Rauti, S., & Laato, S. (2024). Enhancing resilience in IoT cybersecurity: The roles of obfuscation and diversification techniques for improving the multilayered cybersecurity of IoT systems. En Data and Policy (Vol. 6). https://doi.org/10.1017/dap.2024.84

Rauti, S., Laurén, S., Mäki, P., Uitto, J., Laato, S., & Leppänen, V. (2021). Internal interface diversification as a method against malware. En Journal of Cyber Security Technology (Vol. 5, Número 1, pp. 15-40). https://doi.org/10.1080/23742917.2020.1813397

Regano, L., & Canavese, D. (2024). Automated Intel SGX Integration for Enhanced Application Security. En IEEE Access (Vol. 12, pp. 110312-110321). https://doi.org/10.1109/ACCESS.2024.3441240

Rodríguez Véliz, Y. N., Miguel, Hernández González, Anaisa, Lima, Roberto Sepúlveda, Musa. (2024). Validación de un modelo de protección basado en la ofuscación

del código. Ingeniería Industrial, 45, 91-108.

Saxena, U. R., & Alam, T. (2022). Role based access control using identity and broadcast based encryption for securing cloud data. En Journal of Computer Virology and Hacking Techniques (Vol. 18, Número 3, pp. 171-182). https://doi.org/10.1007/s11416-021-00402-1

Suk, J. H., & Lee, D. H. (2020). VCF: Virtual Code Folding to Enhance Virtualization Obfuscation. IEEE Access, 8, 139161-139175. https://doi.org/10.1109/ACCESS.2020.3012684

Tang, Z., Yu, J., Chai, X., Ma, T., Gan, Z., & Wang, B. (2025). ImageShield: A responsibility-to-person blind watermarking mechanism for image datasets protection. En Applied Intelligence (Vol. 55, Número 1). https://doi.org/10.1007/s10489-024-06093-7

Xiong, X., Sha, Z., Shu, H., & Wei, R. (2025). Code obfuscation based on deep integration. En Computer Journal (Vol. 68, Número 12, pp. 1980-1995). https://doi.org/10.1093/comjnl/bxaf089

Yue, T., Zhang, F., Ning, Z., Wang, P., Zhou, X., Lu, K., & Zhou, L. (2024). Armor: Protecting Software Against Hardware Tracing Techniques. IEEE Transactions on Information Forensics and Security, 19, 4247-4262. https://doi.org/10.1109/TIFS.2024.3372816

Zhang, W., Xu, Z., Xiao, Y., & Xue, Y. (2022). Unleashing the power of pseudo-code for binary code similarity analysis. En Cybersecurity (Vol. 5, Número 1). https://doi.org/10.1186/s42400-022-00121-0

Zhou, Z., & Wang, C. (2023). Practical Anti-Fuzzing Techniques With Performance Optimization. IEEE Open Journal of the Computer Society, 4, 206-217. https://doi.org/10.1109/OJCS.2023.3301883