UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMÁTICA

Resolución: RPC-SO-02-No.053-2021

PROYECTO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGISTER

Título del proyecto:
Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA.
Línea de Investigación:
Ciencias de la ingeniería aplicadas a la producción, sociedad y desarrollo sustentable
Campo amplio de conocimiento:
Tecnologías de la Información y la Comunicación (TIC)
Autor/a:
Sánchez López Leonardo Antonio
Tutor/a:
PhD. Renato Toasa PhD. Maryory Urdaneta

Quito – Ecuador

2025

APROBACIÓN DEL TUTOR

Yo, Renato Mauricio Toasa Guachi con C.I: 1804724167 en mi calidad de Tutor del proyecto de investigación titulado:Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA.

Elaborado por: Leonardo Antonio Sanchez Lopez, de C.I: 1722998109, estudiante de la Maestría en Seguridad Informática, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M., septiembre de 2025

Firmado electrónicamente por:
RENATO MAURICIO
TOASA GUACHI
Validar únicamente con FirmaEC

_____

Firma

APROBACIÓN DEL TUTOR

Yo, Maryory Urdaneta Herrera con C.I: 1759316126 en mi calidad de Tutor del proyecto de investigación titulado: Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA.

Elaborado por: Leonardo Antonio Sanchez Lopez, de C.I: 1722998109, estudiante de la Maestría en Seguridad Informática, de la **UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL)**, como parte de los requisitos sustanciales con fines de obtener el Título de Magister, me permito declarar que luego de haber orientado, analizado y revisado el trabajo de titulación, lo apruebo en todas sus partes.

Quito D.M.,septiembre de 2025

MARYORY URDANETA HERRERA

Firmado digitalmente por MARYORY URDANETA HERRERA
Fecha: 2025.09.04 15:17:27 -05'00'

Firma

DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE

Yo, Leonardo Antonio Sánchez López con C.I: 1722998109, autor/a del proyecto de titulación denominado: Metodología para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA. Previo a la obtención del título de Magister en Seguridad Informática.

1. Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar el respectivo trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Manifiesto mi voluntad de ceder a la Universidad Tecnológica Israel los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor@ del trabajo de titulación, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital como parte del acervo bibliográfico de la Universidad Tecnológica Israel.
3. Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de prosperidad intelectual vigentes.

Quito D.M., septiembre de 2025

Firmado electrónicamente por:
LEONARDO ANTONIO SANCHEZ LOPEZ
Validar únicamente con FirmaEC

_____

Firma

Tabla de contenidos

APROBACIÓN DEL TUTOR 2
APROBACIÓN DEL TUTOR 3
DECLARACIÓN DE AUTORIZACIÓN POR PARTE DEL ESTUDIANTE 4
INFORMACIÓN GENERAL 8
Contextualización del tema 8
Problema de investigación 9
Objetivo general 10
Objetivos específicos 11
Vinculación con la sociedad y beneficiarios directos: 11
CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO 13
1.1. Contextualización general del estado del arte 13
1.2. Proceso investigativo metodológico 14
1.3. Análisis de resultados 15
Análisis cuantitativo 15
Análisis cualitativo. 16
Síntesis de hallazgos 17
CAPÍTULO II: PROPUESTA 18
2.1. Fundamentos teóricos aplicados 18
2.2. Descripción de la propuesta 18
a.Estructura General 19
b.Explicación del aporte 21
c.Estrategias y/o técnicas 21
2.3. Validación de la propuesta 22
2.4. Matriz de articulación de la propuesta 22
CONCLUSIONES 26
RECOMENDACIONES 27
BIBLIOGRAFÍA 28
ANEXOS 29

Índice de Tablas

Tabla 1. Frecuencia de riesgos percibidos por los expertos. 16

Tabla 2. Matriz de articulación. 24

Índice de Figuras

Figura 1. Estructura general de la propuesta. 20

INFORMACIÓN GENERAL

Contextualización del tema

Hoy en día la transformación digital ha cambiado muchas cosas en los campos relacionados con la tecnología, los campos de tecnología se han visto muy afectados. Aquí es donde la Inteligencia Artificial (IA) se ha convertido en una fuerza revolucionaria en el campo de la informática. En ella, los modelos de lenguaje de gran escala (LLMs) se han vuelto más que nunca imprescindibles, ya que no solo entienden el lenguaje natural, sino que también generan respuestas y soluciones que antes exigían conocimientos especializados. Este desarrollo revolucionó la manera en que la gente interactúa con la información y resuelve problemas técnicos. Herramientas como ChatGPT o GitHub Copilot son un ejemplo de ello, ya que permiten generar código y hacer posible que personas no expertas en programación crean software funcional con instrucciones sencillas (Jošt et al., 2023).

Y aunque esta alta demanda tecnológica es una gran oportunidad, también trae grandes retos, especialmente en el ámbito de la seguridad de la información. Ahora cualquiera puede desarrollar una aplicación completa o automatizar procesos sin ser un experto en seguridad informática, lo que muchas veces resulta en sistemas funcionales, pero inseguros. Esto se da porque los modelos de lenguaje no siempre pueden distinguir entre un código funcional y uno seguro. Por ejemplo, es posible que se produzcan scripts con errores clásicos como inyecciones SQL o vulnerabilidades XSS, sin que el usuario tenga las herramientas para detectarlos y corregirlos por su cuenta (Brundage et al., 2023).

Esta situación ha generado un nuevo tipo de riesgo: uno que no proviene necesariamente del exterior, sino del uso mal informado o poco crítico de la inteligencia artificial en entornos de desarrollo internos. Cuando el código generado por IA no es evaluado por profesionales con experiencia en ciberseguridad, se abre la posibilidad de que estos errores sean explotados por atacantes, lo cual amplía significativamente la superficie de ataque de muchas organizaciones.

Al mismo tiempo, los propios modelos de lenguaje también están siendo utilizados por actores maliciosos para llevar a cabo ataques más sofisticados. Ya se han identificado usos de estas tecnologías para crear correos de phishing que son más difíciles de detectar, desarrollar nuevas variantes de malware con rapidez, e incluso automatizar la búsqueda de vulnerabilidades que todavía no han sido divulgadas oficialmente (National Cyber Security Centre, 2023). Esto demuestra que la inteligencia artificial, además de ser una herramienta poderosa para los defensores, también lo es para quienes buscan vulnerar sistemas.

Frente a este panorama, los métodos tradicionales para proteger los sistemas, como las pruebas de penetración manuales, se están quedando atrás. Aunque son métodos detallados y efectivos, no logran cubrir toda la velocidad ni el volumen con el que hoy se crean nuevas aplicaciones y estructuras. Además, requieren tiempo, recursos y personal especializado, lo cual no siempre está al alcance de todas las organizaciones.

Por estas razones, una posible alternativa que surge es automatizar también las defensas. En este contexto se plantea la propuesta del agente autónomo denominado "CAI", pensado para realizar pruebas de pentesting ético con el apoyo de modelos de lenguaje. La finalidad es utilizar esta misma tecnología, pero desde una perspectiva constructiva, donde el objetivo no sea escribir código inseguro, sino todo lo contrario: descubrir posibles fallos antes de que sean aprovechados por otros. Se trataría, entonces, de una solución que permita anticiparse a las amenazas y generar una auditoría continua, inteligente y automatizada sobre los sistemas que están en uso (Pearce et al., 2023).

Finalmente, el auge del uso abierto de modelos de lenguaje en el desarrollo tecnológico plantea serios desafíos para la seguridad de la información. La solución no está en prohibir su uso, sino en entender los riesgos, gestionarlos adecuadamente y, en lo posible, utilizar estas mismas herramientas para proteger lo que hoy se está volviendo tan fácil de vulnerar.

Problema de investigación

El problema que motiva esta investigación surge a partir de una nueva realidad tecnológica: el uso extendido y cada vez más acelerado de LLM tanto en el desarrollo de software como en tareas relacionadas con la ciberseguridad. Aunque estas herramientas han sido diseñadas para facilitar procesos y aumentar la productividad, su adopción sin los conocimientos adecuados ha empezado a generar una brecha de seguridad que es cada vez más difícil de manejar. En la práctica, lo que se observa es una clara desventaja para los equipos de seguridad, que no logran mantenerse al ritmo de los nuevos riesgos que aparecen a diario.

El conflicto principal está en la diferencia de velocidad: mientras las vulnerabilidades aumentan rápidamente por el uso automatizado e incluso irresponsable de estos modelos, las estrategias tradicionales de protección siguen siendo lentas, manuales o limitadas. Las metodologías convencionales —como el análisis de código o las pruebas de penetración manuales— no fueron diseñadas para trabajar al mismo ritmo al que hoy se crean y despliegan aplicaciones apoyadas por inteligencia artificial.

Este problema se refleja en dos dimensiones. Por un lado, dentro de las organizaciones, cada vez más personas sin formación especializada en seguridad están utilizando estos modelos para generar código. Aunque esto puede parecer una ventaja, lo cierto es que muchas veces ese código presenta errores de seguridad que pasan desapercibidos por falta de conocimientos técnicos. Estas fallas no son necesariamente complejas, pero sí críticas, y pueden generar vulnerabilidades serias que quedan ocultas en el día a día del desarrollo. En este sentido, se genera un riesgo interno, difícil de prever y más difícil aún de auditar.

Por otro lado, los mismos modelos de lenguaje también están siendo utilizados por actores externos para mejorar sus tácticas de ataque. Hoy ya no es necesario ser un hacker experimentado para escribir un código malicioso o para armar una campaña de phishing efectiva: con la ayuda de la IA, muchas barreras técnicas se han reducido, lo que amplía el espectro de quienes pueden lanzar ataques con cierta efectividad. Además, los ciberatacantes más avanzados ahora cuentan con herramientas mucho más sofisticadas para automatizar y escalar sus operaciones.

En consecuencia, los profesionales de la ciberseguridad se encuentran en una posición reactiva. Las herramientas automatizadas que existen actualmente muchas veces no detectan vulnerabilidades complejas o relacionadas con la lógica del negocio, mientras que las pruebas manuales no pueden abarcar la enorme cantidad de sistemas y código que se generan constantemente. Esto no solo incrementa los falsos positivos, sino que también deja sin cubrir muchos puntos vulnerables.

Las prácticas de pentesting ético, tal como se aplican hoy, ya no son suficientes para enfrentar los riesgos emergentes que trae el uso masivo de modelos de lenguaje. Las amenazas evolucionan más rápido que los mecanismos de defensa. Por eso, se vuelve urgente repensar las estrategias de evaluación y protección de sistemas, y desarrollar herramientas que sean capaces de responder con el mismo nivel de autonomía e inteligencia con el que se están creando los nuevos riesgos.

Objetivo general

Proponer un proceso sistemático para el fortalecimiento de la seguridad en el ciclo de vida del desarrollo de software (SDLC) mediante la integración de un agente autónomo de pentesting ético.

Objetivos específicos

- Contextualizar los fundamentos teóricos sobre el desarrollo seguro de software (SDLC), las vulnerabilidades introducidas por los LLMs y las capacidades actuales de los agentes autónomos para pentesting ético.
- Analizar los puntos críticos en el ciclo de vida del desarrollo de software donde el código generado por IA introduce mayores riesgos.
- Diseñar un proceso sistemático de aseguramiento de la calidad del código que integre un agente de pentesting autónomo en las fases de desarrollo y pruebas (CI/CD).
- Validar la propuesta del proceso sistemático a través del criterio de especialistas en ciberseguridad, evaluando su pertinencia, aplicabilidad y potencial de implementación en entornos reales de desarrollo.

Vinculación con la sociedad y beneficiarios directos:

El proyecto de titulación propuesto mantiene una vinculación directa con las necesidades actuales del entorno profesional y tecnológico, en particular con los desafíos que enfrentan las organizaciones en materia de ciberseguridad frente al uso emergente de herramientas basadas en inteligencia artificial, como los LLMs. Al plantear un proceso que nos permita integrar un agente autónomo de pentesting en el SDLC, se busca dar una solución tangible, aplicable y relevante en el mundo real en el que automatización y seguridad se entrelazan.

En esta relación, se planificó realizar una capacitación presencial con el equipo de Tecnologías de la Información de la empresa BestPoint, ubicada en la ciudad de Quito. Esta forma de desarrollar este proceso sistematizado se ajusta a los propósitos de fortalecimiento de capacidades técnicas en sectores productivos y tecnológicos, haciendo contribuciones más allá del ámbito académico. El proyecto apoya la mejora de las prácticas seguras de desarrollo en instituciones educativas, creando las condiciones para promover una cultura de ciberseguridad preventiva.

Esta reunión es para socializar los puntos relevantes del proyecto, pero sobre todo para presentar la propuesta de un proceso sistematizado del agente CAI, el cual se integra a los flujos CI/CD y que sea capaz de prevenir vulnerabilidades en entornos de código potenciados por IA. La presencia de profesionales de la seguridad informática en estos espacios puede validar, retroalimentar y mejorar la aplicabilidad de la propuesta en el mundo real.

Se realizó una reunión presencial en STB-EC SAS , empresa que crea soluciones tecnológicas a medida y software de facturación. Como parte del proceso de socialización con el proyecto, se tuvo una presentación con ocho personas del departamento de TI. La sesión presentó los

principales hallazgos de la investigación y el concepto del agente " CAI ", como se muestra en el Anexo 1.

Los usuarios a quién va dirigido este trabajo son profesionales de TI, desarrolladores, personal de QA y especialistas en ciberseguridad que desean mejorar sus estrategias de protección desde las primeras fases del ciclo de vida del software. Son beneficiarios indirectos, además, los actores institucionales y los usuarios finales que forman parte del ecosistema digital y que se benefician de sistemas más seguros y resilientes.

CAPÍTULO I: DESCRIPCIÓN DEL PROYECTO

1.1. Contextualización general del estado del arte

En los últimos años, la inteligencia artificial, y más concretamente los modelos LLM, han revolucionado el desarrollo de software. Estas herramientas pueden generar código, documentar procesos y realizar pruebas simples, lo que ha llevado a su adopción en equipos técnicos de diversas industrias (Jošt et al., 2022; Brundage et al., 2023); pero también han abierto la puerta a nuevos debates sobre los riesgos que implican, especialmente en lo que concierne a la seguridad del código que producen (NCSC y Turing Institute, 2023).

Las últimas investigaciones en ciberseguridad muestran esta preocupación: mientras que los LLM pueden acelerar el proceso de desarrollo, también pueden introducir errores o vulnerabilidades difíciles de detectar (Pearce et al., 2023). Esto empeora cuando los desarrolladores confían ciegamente en el contenido creado sin revisarlo. Y eso es por no hablar de que muchos IA ni siquiera están entrenadas con las últimas mejores prácticas de seguridad, lo que puede hacer que den respuestas obsoletas o inseguras.

Al mismo tiempo, la automatización del pentesting con agentes inteligentes es una nueva área de innovación. Estas herramientas intentan simular auditorías técnicas de seguridad de manera autónoma, insertándose en pipelines de CI/CD sin impactar los tiempos de entrega y sin necesitar intervención humana (Sparks of AGI, 2023).

El enfoque propuesto de un agente autónomo de pentesting (CAI) integrado en el flujo de trabajo de CI/CD. Su inserción responde a un enfoque proactivo de seguridad alineado con la filosofía DevSecOps, especialmente bajo el principio “shift-left”, el cual prioriza la detección temprana de vulnerabilidades.

A partir de este análisis, se puede afirmar que ya existe un consenso emergente sobre el potencial transformador de los LLMs en la ciberseguridad, tanto en sus riesgos como en sus oportunidades. Se ha explicado con claridad que su uso no supervisado puede introducir fallos críticos y que, al mismo tiempo, su incorporación en defensas automatizadas abre nuevas posibilidades. Sin embargo, los estudios realizados hasta el momento son mayormente exploratorios. Aún no existen metodologías estandarizadas para integrar de manera segura y eficiente estas tecnologías en entornos reales de desarrollo de software, especialmente dentro del ciclo de vida del desarrollo seguro (SDLC) y los pipelines de integración y despliegue continuo (CI/CD).

Por tanto, queda por estudiar cómo operacionalizar esta tecnología en contextos profesionales. Es decir, cómo diseñar, implementar y validar un agente autónomo de pentesting apoyado por LLMs que sea capaz de adaptarse a las exigencias reales de los equipos de desarrollo. Esta investigación se plantea precisamente en ese vacío de conocimiento: construir una metodología clara y aplicable que permita evaluar los riesgos generados por la IA y, al mismo tiempo, utilizar esa misma tecnología como aliada para prevenir fallos antes de que estos puedan ser explotados. El objetivo es pasar de una ciberseguridad reactiva a una más proactiva, continua e inteligente.

1.2. Proceso investigativo metodológico

La presente investigación adoptó un enfoque metodológico mixto ya que tiene carácter cualitativo y cuantitativo, orientado a comprender en profundidad un fenómeno emergente en el ámbito de la ciberseguridad: las amenazas asociadas al uso de LLMs. Esta elección metodológica se sustenta en la naturaleza exploratoria y contextual del objeto de estudio, cuyo propósito no es medir variables numéricas, sino analizar situaciones complejas, interpretar significados y plantear soluciones fundamentadas en la realidad del entorno profesional. Tal como señalan Hernández et al. (2014), el enfoque cualitativo es especialmente pertinente cuando se abordan problemáticas poco exploradas o en proceso de transformación, ya que permite generar conocimiento a partir del análisis profundo del contexto y de las percepciones de los actores implicados.

El desarrollo del estudio se organizó en dos fases complementarias:

- **Fase exploratoria-descriptiva:** Se llevó a cabo una revisión bibliográfica exhaustiva con el objetivo de identificar los conceptos clave y delimitar el estado actual del conocimiento. El análisis de contenido, apoyado en fichas de lectura, permitió clasificar y sintetizar la información en torno a ejes como el Secure SDLC, las vulnerabilidades derivadas del uso de IA generativa y las propuestas emergentes sobre agentes autónomos de auditoría (Pearce et al., 2023; NCSC, 2023).

- **Fase de diseño y validación de la propuesta:** En esta etapa se elaboró una metodología para la integración de un agente autónomo de *pentesting* (denominado “CAI”) en entornos de CI/CD. Con el fin de evaluar su viabilidad y pertinencia, se aplicó la técnica de juicio de expertos. La población estuvo conformada por profesionales en ciberseguridad y desarrollo seguro de software en América Latina, de la cual se seleccionó una muestra intencional integrada por nueve especialistas con amplia trayectoria en el sector.

Ante la dificultad de sincronizar agendas con los encuestados, se optó por utilizar un único instrumento de recolección de datos de manera asíncrona: un cuestionario online (ver en Anexo 2). Este formulario se estructuró de forma mixta:

1. **Preguntas cerradas:** Para obtener valoraciones estructuradas sobre la propuesta.
2. **Preguntas abiertas:** Para recoger percepciones y sugerencias detalladas por escrito.

Para este proceso sistemático se encuestó a nueve expertos de Ecuador, de esta manera se pudo obtener información de diferentes perspectivas dentro del campo técnico, garantizando que la propuesta final esté fundamentada tanto en la teoría como en la práctica de las personas que lo experimentan a diario.

1.3. Análisis de resultados

En este apartado se presenta el análisis de los datos obtenidos mediante el cuestionario de validación aplicado a una muestra intencional de nueve expertos en ciberseguridad y desarrollo seguro de software de la región latinoamericana. La información recolectada fue procesada con el propósito de identificar tendencias en las respuestas cerradas y, de forma complementaria, realizar un análisis temático de las respuestas abiertas, a fin de obtener una validación integral de la propuesta metodológica.

Análisis cuantitativo

Para el análisis de los datos cerrados, se calcularon frecuencias y porcentajes a fin de identificar patrones en la percepción de los participantes, como podemos observar en la Tabla 1. Así, uno de los principales focos fue determinar los riesgos más relevantes que, a criterio de los expertos, introduce el uso de IA generativa en entornos de desarrollo.

Tabla 1

Frecuencia de riesgos percibidos por los expertos

Riesgo identificado	Frecuencia (N=9)	Porcentaje
Excesiva dependencia y erosión de habilidades de seguridad	8	88,9%
Fuga de información sensible y propiedad intelectual	7	77,8%
Inyección de vulnerabilidades en el código fuente	6	66,7%
Degradación de la calidad y mantenibilidad del código	6	66,7%

Como se observa en la Tabla 1, el riesgo con mayor frecuencia señalado (88,9%) fue la excesiva dependencia de la IA y la consecuente disminución de las competencias de seguridad en los equipos de desarrollo. Este hallazgo refleja que la principal inquietud no recae únicamente en la tecnología, sino en su efecto sobre el factor humano. En segundo lugar, se encuentra la preocupación por la fuga de información sensible (77,8%), seguida por la inyección de vulnerabilidades y la degradación de la calidad del código (66,7% en ambos casos).

Análisis cualitativo.

El examen cualitativo de las respuestas abiertas permitió identificar cuatro ejes centrales que sintetizan la percepción de los especialistas:

1. **Confirmación de la problemática**

   El consenso fue absoluto: el código generado por IA, aunque funcional, con frecuencia contiene vulnerabilidades. Se atribuye esta situación a que la herramienta carece del contexto completo del proyecto y a la tendencia de los desarrolladores menos experimentados a confiar ciegamente en los resultados. Un participante señaló que este riesgo se agrava “cuando se delega todo a la IA”, mientras que otro destacó que la calidad del código depende en gran medida de la precisión del *prompt*.

2. **Pertinencia y viabilidad de la propuesta**

   La incorporación de un agente autónomo de *pentesting* fue valorada como una solución “necesaria” y “funcional” para fortalecer los procesos de QA y seguridad. Su principal

aporte radica en la automatización y detección temprana de fallos. Sin embargo, se mencionaron desafíos como la compatibilidad con diversos lenguajes y *frameworks*, así como el impacto potencial en el rendimiento de los entornos CI/CD.

3. **Barreras para la adopción**

   Las principales limitaciones identificadas son de tipo cultural y económico. La resistencia de algunos equipos de desarrollo ante herramientas que podrían percibirse como un obstáculo —especialmente en casos de falsos positivos— representa un reto significativo. Asimismo, se señaló que su implementación implicaría recursos adicionales, por lo que su adopción sería más factible en sectores que manejan datos altamente sensibles, como el bancario.

4. **Recomendaciones de mejora**

   Las sugerencias se centraron en tres aspectos clave:

   - **Precisión:** Reducir al mínimo falsos positivos y falsos negativos, identificando con exactitud la ubicación y naturaleza de la vulnerabilidad.
   - **Integración y reportes:** Conectar el agente con herramientas de gestión de proyectos para la generación automática de tickets y elaboración de reportes claros.
   - **Adaptabilidad:** Permitir la configuración y ajuste (*fine-tuning*) del agente según las particularidades de cada entorno y proyecto.

Síntesis de hallazgos

Los resultados cuantitativos y cualitativos convergen en tres conclusiones principales:

- Existe consenso sobre la relevancia del problema y la necesidad de soluciones automatizadas para mitigar los riesgos introducidos por los LLMs.
- La propuesta del agente "CAI" se percibe como técnicamente viable y útil para fortalecer la seguridad en fases tempranas del ciclo de desarrollo.
- Su adopción exitosa dependerá de superar barreras técnicas (rendimiento y compatibilidad) y culturales (aceptación por parte de los desarrolladores).

En consecuencia, las recomendaciones recogidas serán incorporadas al diseño final con el fin de asegurar que la herramienta no solo sea eficaz en la detección de vulnerabilidades, sino también precisa, adaptable y alineada con las dinámicas de trabajo de los equipos de desarrollo.

CAPÍTULO II: PROPUESTA

2.1. Fundamentos teóricos aplicados

La metodología que propongo para el agente "CAI" se construye sobre marcos conceptuales y tecnológicos ya establecidos en el desarrollo de software y la ciberseguridad. La idea es que esto nos permita obtener una respuesta justificada a los nuevos riesgos que los LLMs están introduciendo.

- **Secure SDLC y el enfoque *shift-left***: El Secure SDLC busca integrar la seguridad en todo el ciclo de vida del software. Según el NIST (2022), hacerlo desde el inicio reduce costos y vulnerabilidades. El principio de *shift-left* lleva la idea a la práctica, adelantando los controles de seguridad. Mi agente "CAI" aplica este principio al activarse justo después del *commit* del desarrollador.
- **Cultura DevSecOps:** Mientras el Secure SDLC dice "qué" hacer, DevSecOps explica el "cómo". Esta cultura promueve la colaboración y la automatización para integrar la seguridad en el día a día. Un estudio de la Universidad Israel (Santacruz Egas, 2024) muestra cómo integrar la seguridad en los *pipelines* mejora la eficiencia, y mi propuesta se alinea con esa filosofía.
- **Riesgos de los LLMs:** La existencia de "CAI" se justifica por las nuevas amenazas que traen los LLMs. OWASP ya ha comenzado a catalogarlas en su proyecto Top 10 for LLM Applications (OWASP, 2023), y mi agente está diseñado para actuar como una contramedida a estos riesgos.
- **Agentes autónomos de *pentesting*:** El aspecto más innovador de mi tesis es el uso de agentes autónomos. Como sugieren Pearce et al. (2023), estos sistemas pueden planificar y razonar para realizar pruebas de seguridad de una manera más contextual, casi como lo haría un humano. El diseño de "CAI" sigue este enfoque.
- **Gestión responsable de IA (ISO/IEC 42001):** Finalmente, toda la propuesta se enmarca en los principios de la norma ISO/IEC 42001:2023. Este estándar guía a las organizaciones para que usen la IA de manera ética y responsable. Al actuar como un control de calidad y seguridad, "CAI" se alinea con los objetivos de esta norma en cuanto a gestión de riesgos y transparencia (ISO/IEC, 2023).

2.2. Descripción de la propuesta

La idea principal de este trabajo es plantear un proceso estructurado que utiliza el CAI, un agente autónomo de pentesting ético diseñado para vivir dentro del SDLC, específicamente en los canales de CI/CD. El objetivo de este agente es anticiparse a los riesgos de seguridad

emergentes, en particular los relacionados con el uso de código producido o potenciado por LLMs.

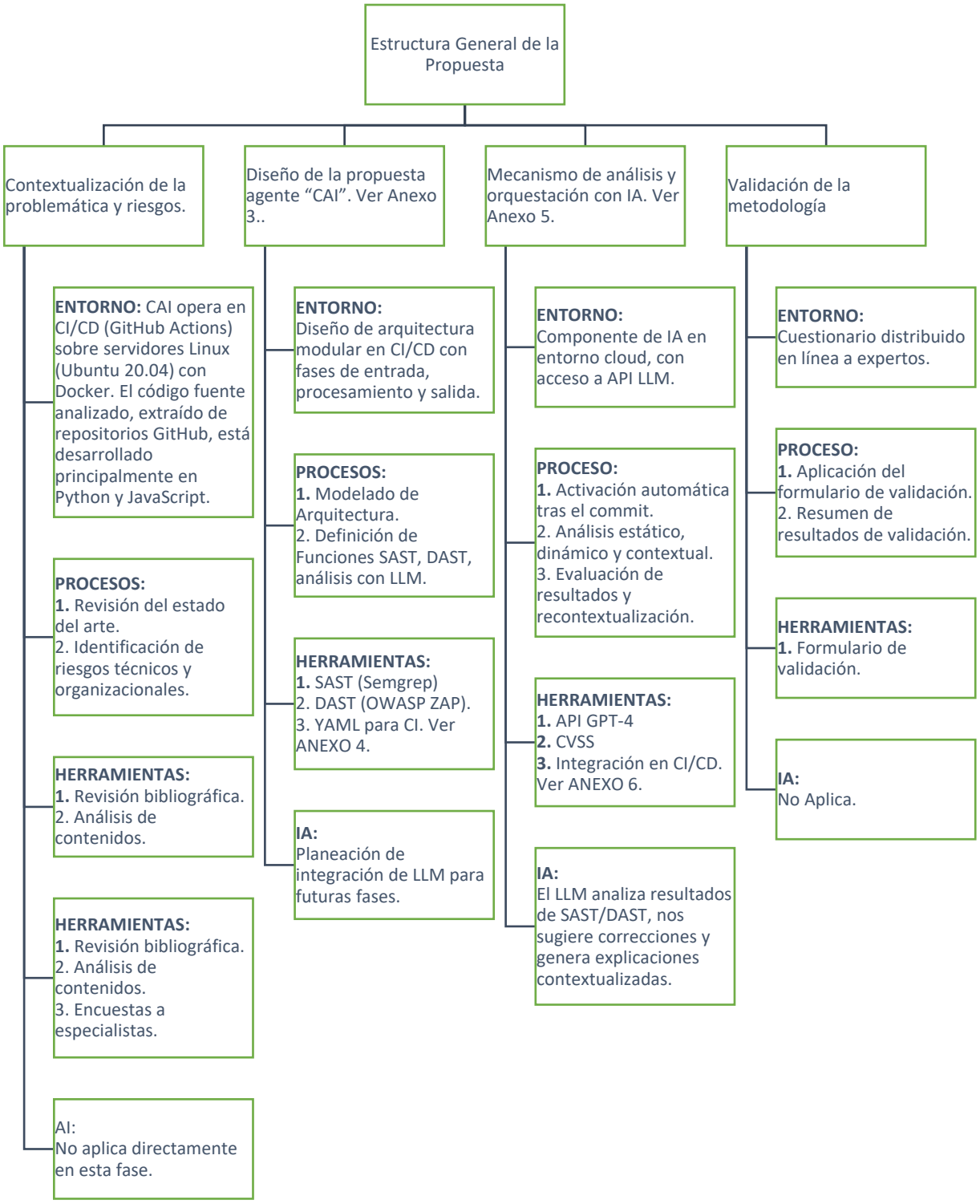
A. Estructura General

La arquitectura de CAI se plantea como un sistema de procesamiento con tres componentes principales: entradas, núcleo y salidas. Este flujo actúa como una puerta de control de calidad y seguridad en la tubería de CI/CD. Buscan disminuir los riesgos de seguridad que surgen al usar código producido por LLMs.

El proceso sistemático se centra en un agente que refuerza la seguridad desde el inicio del desarrollo, aplicando el enfoque *shift-left* de la cultura DevSecOps. A continuación, detallo las fases que componen su funcionamiento en la figura 1.

Figura 1

Estructura general de la propuesta.



B. Explicación del aporte

Como se puede observar en la figura 1, las entradas incluyen elementos como el código fuente en forma de pull requests, las dependencias externas, los artefactos de contenedores, los archivos de IaC, la definición del pipeline y las políticas de seguridad preestablecidas. A partir de esta información, el núcleo de CAI ejecuta un proceso sistemático que contempla la ingesta y contextualización de los artefactos, el análisis estático (SAST, secretos y dependencias), un análisis dinámico opcional en entornos de staging (DAST), la orquestación de hallazgos con un LLM que prioriza y propone soluciones, la correlación y deduplicación de resultados, y la toma de decisiones conforme a las políticas definidas. Las salidas generadas comprenden reportes en formatos estructurados (JSON/XML), la priorización de vulnerabilidades según CVSS, recomendaciones contextualizadas, la creación automática de tickets en sistemas de gestión y la provisión de métricas clave como MTTR o cobertura de análisis. En función de la criticidad de los hallazgos, el pipeline puede ser bloqueado, advertido o simplemente notificado.

El valor de este proceso sistemático nos demuestra cómo transformar los resultados de herramientas automatizadas en seguridad. También nos dice que el CAI no solo detecta amenazas, sino que presenta sus resultados de una manera fácil de entender para los equipos de desarrollo, integrándose sin problemas en los flujos CI/CD existentes. La gran innovación es incorporar un LLM capaz de transformar informes técnicos en mensajes priorizados, fáciles de entender y recomendaciones con acciones. Habilitando una retroalimentación rápida y precisa, reduciendo la necesidad de revisión manual y permitiendo un enfoque de seguridad temprana (shift-left) , todo este proceso se puede observar detalladamente en el ANEXO 7.

C. Estrategias y/o técnicas

Este proceso se realizó en forma iterativa y escalonada, que permite ir implementando en el tiempo y controlando los riesgos. En las primeras etapas se definen políticas estrictas y se configuran los entornos CI/CD. Luego, se agregaron herramientas SAST para revisar código y dependencias en cada PR.

Más adelante, se incorpora un LLM como motor de análisis contextual, se hace DAST en ambientes de puesta en escena y se definen puertas para automatizar decisiones en función del riesgo. Entre las técnicas para operacionalizar CAI se encuentran: SAST y DAST combinados, motores de inferencia LLM, gobernanza mediante políticas como código, clasificación CVSS de vulnerabilidades. Finalmente, se activan cuadros con clave métrica para permitir ajustes continuos.

Con esta integración, CAI se transforma en una solución poderosa y flexible que combina prácticas establecidas de seguridad de aplicaciones con lo último en IA generativa. Como resultado, la solución no solo ofrece una solución tecnológica, sino también un modelo que promueve la cultura DevSecOps y refuerza la seguridad desde el principio del ciclo de vida del software.

2.3. Validación de la propuesta

La validación de la propuesta se llevó a cabo mediante el método de juicio de especialistas, en el cual un grupo de expertos con experiencia en el campo de la seguridad de la información y el desarrollo de software juzgó la propuesta. A continuación, se describen los perfiles de los expertos que realizaron la evaluación:

- **Mgst. Andrés F. Loja Morocho**: profesional con estudios de cuarto nivel en Seguridad de la Información y años de experiencia en auditoría tecnológica; Actualmente trabaja para la firma BestPoint. Su aporte como revisor permitió comparar la propuesta desde la perspectiva de auditoría de controles y riesgos tecnológicos empresariales. Observar en el Anexo 8.

- **Mgst. Liliana M. Cajas Chuqui**: es una especialista en Seguridad de la Información con trayectoria en auditoría informática, desempeñándose en la consultora Golden Auditores. Su aporte se centró en el análisis crítico de la viabilidad del proceso, considerando criterios de aplicabilidad en entornos reales de evaluación. Observar en el Anexo 9.

- **Ing. Mauricio Aguilera**: es Ingeniero en Tecnología de la Información con cinco años de experiencia. Actualmente se desempeña como Desarrollador de Software en la empresa Familify, contribuyendo con una visión práctica vinculada al ciclo de vida del desarrollo y ofreciendo una evaluación de la propuesta desde la perspectiva del equipo implementador y usuario directo de la herramienta (Anexo 10).

La selección de estos perfiles, que conjugan la experiencia en auditoría de seguridad con la práctica en desarrollo de software, permitió obtener una validación integral y consistente de la propuesta, garantizando que su análisis contempla tanto la solidez técnica como la aplicabilidad práctica.

2.4. Matriz de articulación de la propuesta

En la Tabla 2 se sintetiza la articulación del producto realizado con los sustentos teóricos, metodológicos, estratégicos-técnicos y tecnológicos empleados.

Tabla 2

Matriz de articulación

EJES O PARTES PRINCIPALES	SUSTENTO TEÓRICO	SUSTENTO METODOLÓGICO	ESTRATEGIAS / TÉCNICAS	DESCRIPCIÓN DE RESULTADOS	INSTRUMENTOS APLICADOS
Contextualización de la problemática y riesgos	Identificación de riesgos emergentes en aplicaciones basadas en LLMs, considerando el OWASP Top 10 for LLMs y vulnerabilidades señaladas en el código asistido por IA (NCSC, 2023).	Enfoque cualitativo con carácter exploratorio–descriptivo, fundamentado en revisión bibliográfica y análisis de contenido.	Estudio de la brecha existente entre la aceleración del desarrollo soportado por IA y las limitaciones de los enfoques de seguridad tradicionales.	La totalidad de expertos consultados confirmó que el código generado mediante IA introduce vulnerabilidades. La principal inquietud fue la erosión de competencias en seguridad (88,9 %).	Fichas de revisión bibliográfica y cuestionario en línea con bloque de percepción de la problemática.
Diseño de la propuesta (agente “CAI”)	Fundamentos en el Secure SDLC y el enfoque shift-left (NIST, 2022). Cultura DevSecOps y automatización. Referencias a agentes autónomos de pentesting (Pearce et al., 2023).	Investigación de tipo propositiva, con diseño metodológico derivado de la síntesis de los fundamentos teóricos.	Definición de una arquitectura en tres fases (Entradas–Procesamiento–Salidas). Implementación iterativa bajo sprints. Integración de técnicas SAST, DAST y razonamiento con LLM.	Los expertos valoraron la propuesta como “necesaria”, “funcional” y “útil”, destacando su aporte a la identificación temprana de fallos.	Organizador gráfico de la propuesta y descripción detallada de componentes y fases.

Mecanismo de análisis y orquestación con IA	Fundamentos en razonamiento y planificación de agentes autónomos. Capacidades de los LLMs para contextualizar riesgos y generar explicaciones (Brundage et al., 2023).	Diseño del componente innovador de orquestación como parte integral de la propuesta.	Uso de LLM para explicar riesgos, priorizar hallazgos con CVSS, sugerir correcciones contextualizadas y realizar correlación/deduplicación de alertas.	Se destacó el valor de la automatización "inteligente". El reto más crítico identificado fue la precisión en la detección y la gestión de falsos positivos para garantizar confianza.	Cuestionario en línea en la sección de evaluación de la propuesta.
Validación de la metodología	Fundamentos de investigación cualitativa según Hernández Sampieri et al. (2014).	Aplicación de juicio de expertos mediante muestreo intencional (n = 9 profesionales).	Implementación asincrónica de un instrumento estructurado que favoreció la participación y la obtención de respuestas elaboradas.	La metodología fue considerada pertinente y con alto potencial. Se advirtieron posibles barreras culturales (resistencia al cambio) y económicas (costo de adopción).	Formulario de validación de especialistas.

CONCLUSIONES

Como resultado del proceso investigativo y del análisis de los datos recopilados, se establecen las siguientes conclusiones, las mismas que corresponden a los objetivos definidos al inicio del estudio:

Se contextualizó la base teórica de mi propuesta, identificando que conceptos como el Secure SDLC, la cultura DevSecOps y los agentes autónomos de *pentesting* se conectan de forma coherente. Mostrándonos que aunque se conocen los riesgos de la IA generativa, todavía faltan soluciones prácticas para mitigarlos, lo que justifica la relevancia de este estudio.

El análisis sobre los puntos críticos, reforzado por la opinión de los especialistas, ayudó a entender que el riesgo que más señalan los expertos es la dependencia excesiva en la IA y la pérdida de habilidades de seguridad en los equipos, además de la fuga de información sensible. Concluyendo que el impacto de la IA en el desarrollo de software es tanto humano como tecnológico.

Se diseñó el proceso estandarizado enfocado en el agente "CAI", con arquitectura modular para integrarse en flujos CI/CD, demostrando que su mayor contribución es el módulo orquestador con LLMs, el cual transforma descubrimientos técnicos en información accionable y ayuda en la seguridad desde el comienzo del desarrollo.

Por último, en cuanto a la validación, un panel de expertos revisó mi propuesta y la aprobada como relevante, aplicable y necesaria. Pero también se identifican elementos que podrían limitar su adopción, como la resistencia al cambio y la exigencia de que la herramienta sea muy certera para evitar falsos positivos.

RECOMENDACIONES

Basado en la finalización y lo aprendido en el desarrollo de este proyecto, propongo las siguientes recomendaciones para futuros trabajos en el área:

Se sugiere que el siguiente paso sea desarrollar un prototipo funcional del agente CAI. Siendo este útil para crear un Producto Mínimo Viable (MVP) para medir su rendimiento y eficacia en un entorno de CI/CD real.

Dado que el principal riesgo identificado es el factor humano, recomiendo realizar estudios socio-técnicos sobre cómo impactaría la integración de agentes autónomos en los equipos de desarrollo. Gracias a esto, se podrían crear guías y programas de capacitación para fomentar un uso crítico de estas herramientas.

Al ser el módulo de orquestación con LLM la parte más innovadora, aconsejo que futuras investigaciones se centren en su optimización. Explorar técnicas de *prompt engineering* o *fine-tuning* podría mejorar mucho la precisión y reducir los falsos positivos, algo que los expertos señalan como fundamental.

Finalmente, frente a la resistencia cultural que se observó durante el proyecto, se recomienda diseñar un modelo de adopción gradual para el CAI, empezando con un modo de “advertencia" antes de activar los bloqueos automáticos, para que los equipos se familiaricen y confíen en la herramienta.

BIBLIOGRAFÍA

Brundage, M., Clark, J., Askell, A., Krueger, G., Henighan, T., Perez, E., ... & Brockman, G. (2023). *Sparks of artificial general intelligence: Early experiments with GPT-4* (arXiv:2303.12712). arXiv. https://arxiv.org/abs/2303.12712

Flick, U. (2015). *Introducción a la investigación cualitativa* (5.ª ed.). Ediciones Morata.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación* (6.ª ed.). McGraw-Hill Education.

International Organization for Standardization. (2023). *ISO/IEC 42001:2023 — Information technology — Artificial intelligence — Management system*. ISO. https://www.iso.org/standard/81228.html

Jošt, G., Taneski, V., & Karakatič, S. (2022). The impact of large language models on programming education and student learning outcomes. *Applied Sciences, 14*(10). https://doi.org/10.3390/app14104115

National Cyber Security Centre, & The Alan Turing Institute. (2023). *The near-term impact of AI on the cyber threat*. https://www.ncsc.gov.uk/files/The-near-term-impact-of-AI-on-the-cyber-threat.pdf

National Institute of Standards and Technology. (2022). *Secure Software Development Framework (SSDF), version 1.1*. U.S. Department of Commerce. https://csrc.nist.gov/publications/detail/white-paper/2022/03/23/secure-software-development-framework-ssdf/final

Pearce, H., George, Z., Tan, B., Bud, V.-M., He, Y., Pardo, F. R., Teo, P. M. L., & Stumm, M. F. P. (2023). Rethinking penetration testing with large language models (arXiv:2311.02773). arXiv. https://arxiv.org/abs/2311.02773

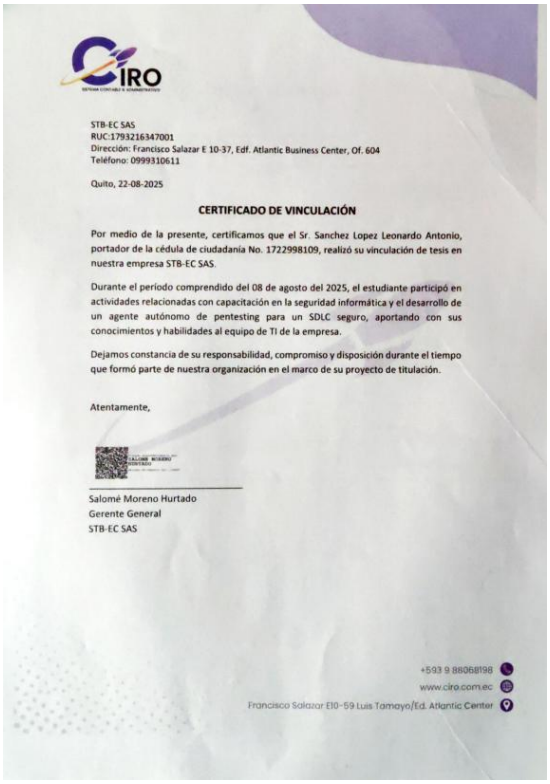
Santacruz Egas, J. P. (2024). *Métricas de seguridad para evaluar los procesos en el Sistema Integrado de Gestión Estratégica de la Universidad Israel aplicando los principios de DevSecOps* [Tesis de maestría, Universidad Israel]. Universidad Israel.

Zaruma, E. (2024). *Informe sobre la seguridad y vulnerabilidad de las aplicaciones bancarias móviles en los sistemas operativos Android e iOS mediante técnicas de Ethical Hacking para mitigar amenazas* [Tesis de maestría, Universidad Tecnológica Israel]. Repositorio Institucional Digital. http://repositorio.uisrael.edu.ec/handle/47000/4142

ANEXOS

ANEXO 1

EVIDENCIA CHARLA PRESENCIAL



CIRO

STB-EC SAS
RUC:1793216347001
Dirección: Francisco Salazar E 10-37, Edf. Atlantic Business Center, Of. 604
Teléfono: 0999310611

Quito, 22-08-2025

CERTIFICADO DE VINCULACIÓN

Por medio de la presente, certificamos que el Sr. Sanchez Lopez Leonardo Antonio, portador de la cédula de ciudadanía No. 1722998109, realizó su vinculación de tesis en nuestra empresa STB-EC SAS.

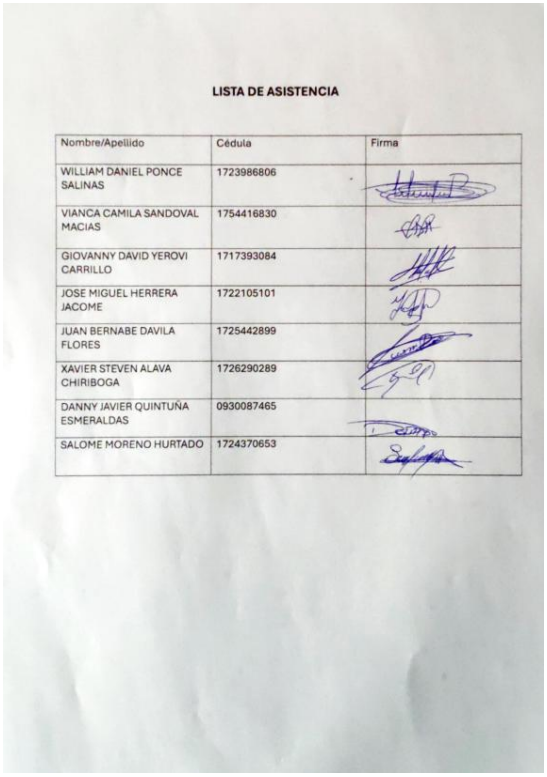
Durante el período comprendido del 08 de agosto del 2025, el estudiante participó en actividades relacionadas con capacitación en la seguridad informática y el desarrollo de un agente autónomo de pentesting para un SDLC seguro, aportando con sus conocimientos y habilidades al equipo de TI de la empresa.

Dejamos constancia de su responsabilidad, compromiso y disposición durante el tiempo que formó parte de nuestra organización en el marco de su proyecto de titulación.

Atentamente,

Salomé Moreno Hurtado
Gerente General
STB-EC SAS

+593 9 88068198
www.ciro.com.ec
Francisco Salazar E10-59 Luis Tamayo/Ed. Atlantic Center

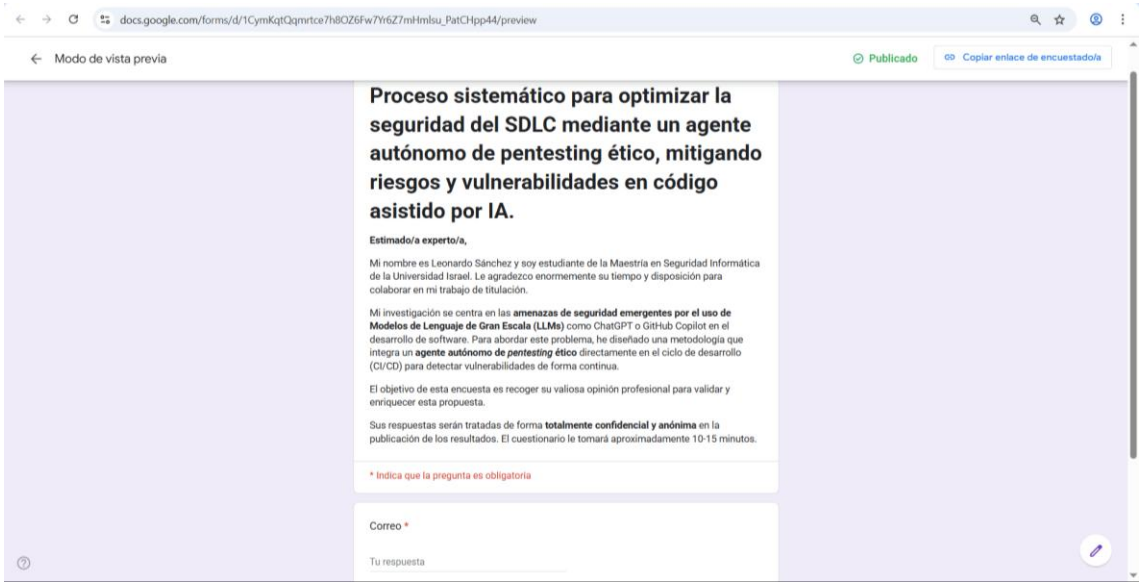


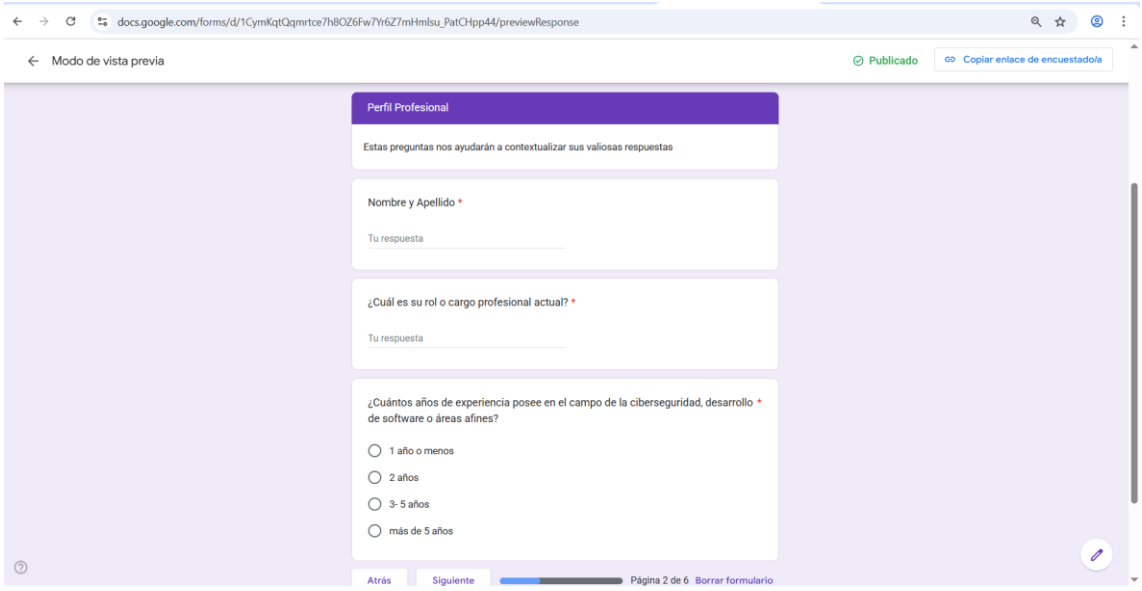
LISTA DE ASISTENCIA

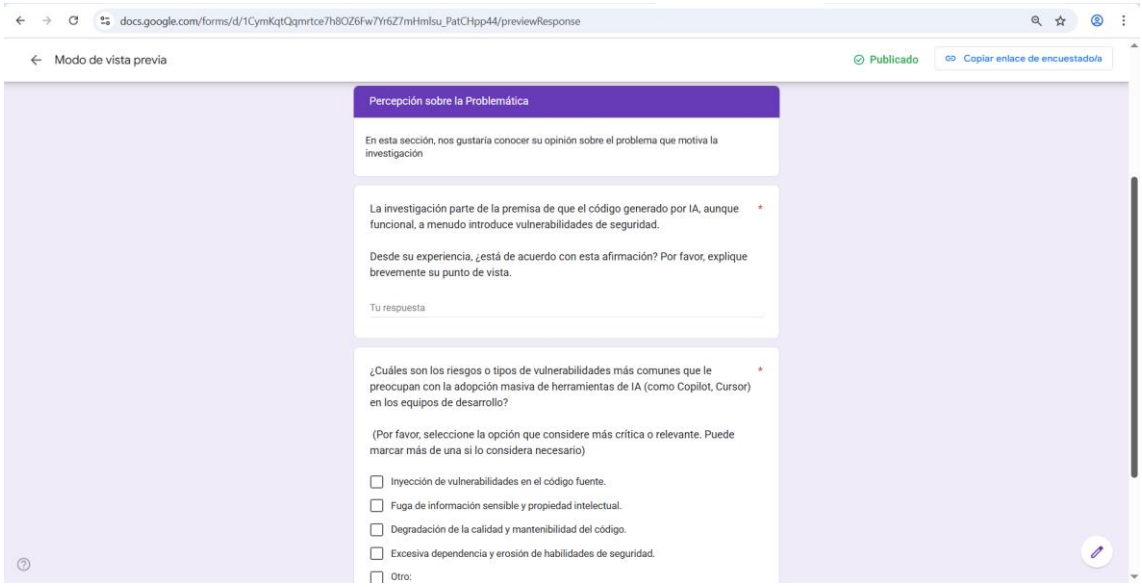
Nombre/Apellido	Cédula	Firma
WILLIAM DANIEL PONCE SALINAS	1723986806	
VIANCA CAMILA SANDOVAL MACIAS	1754416830	
GIOVANNY DAVID YEROVI CARRILLO	1717393084	
JOSE MIGUEL HERRERA JACOME	1722105101	
JUAN BERNABE DAVILA FLORES	1725442899	
XAVIER STEVEN ALAVA CHIRIBOGA	1726290289	
DANNY JAVIER QUINTUÑA ESMERALDAS	0930087465	
SALOME MORENO HURTADO	1724370653	

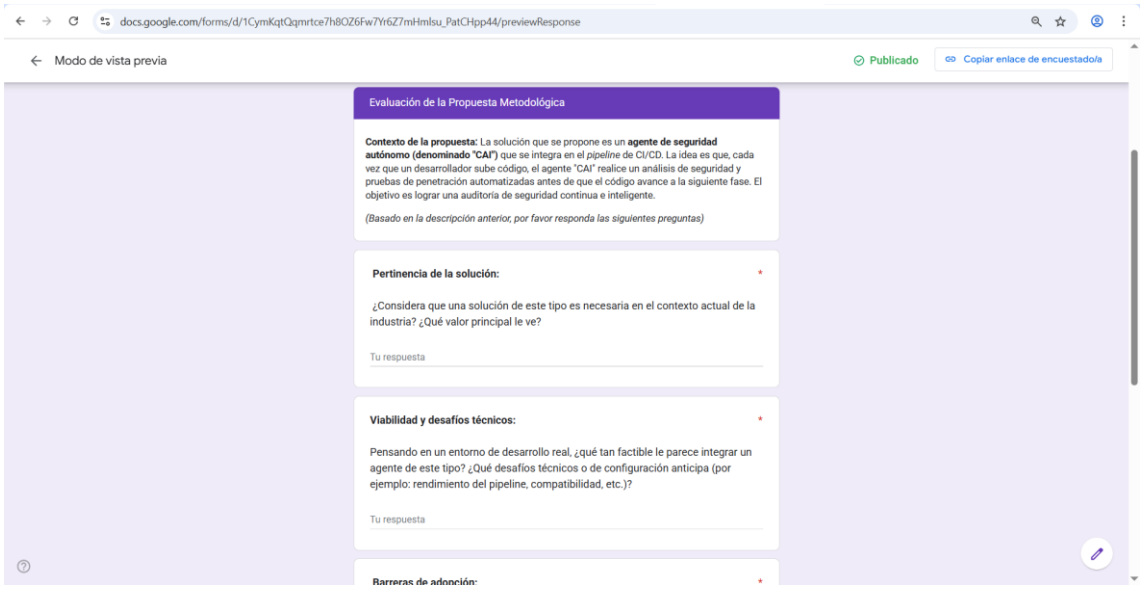
ANEXO 2

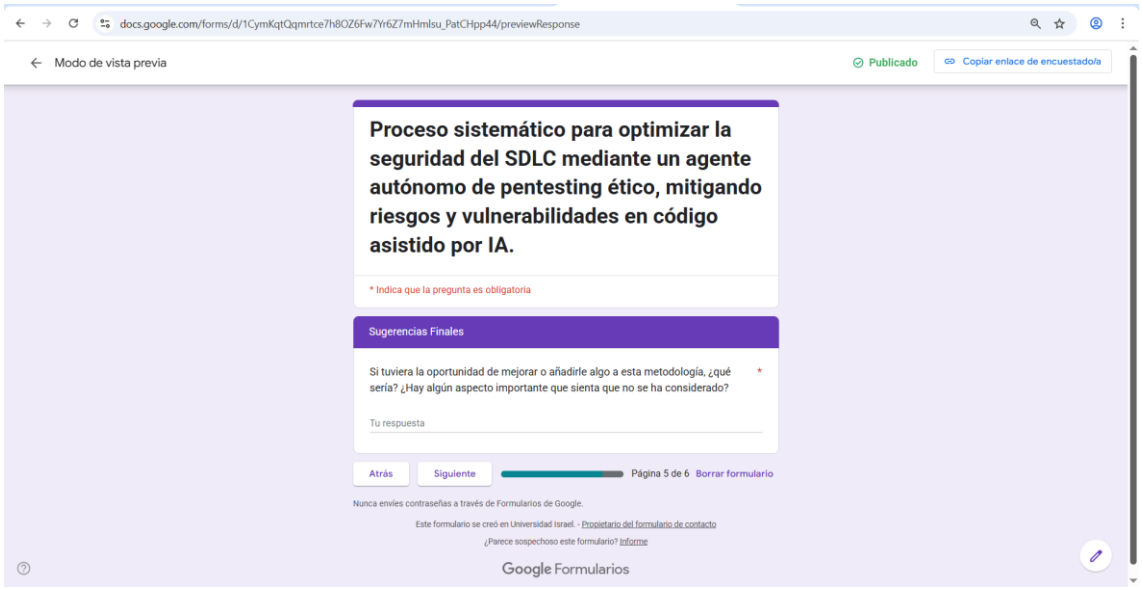
FORMATO DE LA ENCUESTA






docs.google.com/forms/d/1CymKqtQqmrtce7h8OZ6Fw7Yr6Z7mHmlsu_PatCHpp44/previewResponse
Modo de vista previa
Publicado
Copiar enlace de encuestado/a
Percepción sobre la Problemática
En esta sección, nos gustaría conocer su opinión sobre el problema que motiva la investigación
La investigación parte de la premisa de que el código generado por IA, aunque funcional, a menudo introduce vulnerabilidades de seguridad.
Desde su experiencia, ¿está de acuerdo con esta afirmación? Por favor, explique brevemente su punto de vista.
Tu respuesta
¿Cuáles son los riesgos o tipos de vulnerabilidades más comunes que le preocupan con la adopción masiva de herramientas de IA (como Copilot, Cursor) en los equipos de desarrollo?
(Por favor, seleccione la opción que considere más crítica o relevante. Puede marcar más de una si lo considera necesario)
Inyección de vulnerabilidades en el código fuente.
Fuga de información sensible y propiedad intelectual.
Degradación de la calidad y mantenibilidad del código.
Excesiva dependencia y erosión de habilidades de seguridad.
Otro:


docs.google.com/forms/d/1CymKqtQqmrtce7h8OZ6Fw7Yr6Z7mHmlsu_PatCHpp44/previewResponse
Modo de vista previa
Publicado
Copiar enlace de encuestado/a
Evaluación de la Propuesta Metodológica
Contexto de la propuesta: La solución que se propone es un agente de seguridad autónomo (denominado "CAI") que se integra en el pipeline de CI/CD. La idea es que, cada vez que un desarrollador sube código, el agente "CAI" realice un análisis de seguridad y pruebas de penetración automatizadas antes de que el código avance a la siguiente fase. El objetivo es lograr una auditoría de seguridad continua e inteligente.
(Basado en la descripción anterior, por favor responda las siguientes preguntas)
Pertinencia de la solución:
¿Considera que una solución de este tipo es necesaria en el contexto actual de la industria? ¿Qué valor principal le ve?
Tu respuesta
Viabilidad y desafíos técnicos:
Pensando en un entorno de desarrollo real, ¿qué tan factible le parece integrar un agente de este tipo? ¿Qué desafíos técnicos o de configuración anticipa (por ejemplo: rendimiento del pipeline, compatibilidad, etc.)?
Tu respuesta
Barreras de adopción:


docs.google.com/forms/d/1CymKqtQqmrtce7h8OZ6Fw7Yr6Z7mHmlsu_PatCHpp44/previewResponse
Modo de vista previa
Publicado
Copiar enlace de encuestado/a
Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA.
* Indica que la pregunta es obligatoria
Sugerencias Finales
Si tuviera la oportunidad de mejorar o añadirle algo a esta metodología, ¿qué sería? ¿Hay algún aspecto importante que sienta que no se ha considerado?
Tu respuesta
Atrás
Siguiente
Página 5 de 6
Borrar formulario
Nunca envíes contraseñas a través de Formularios de Google.
Este formulario se creó en Universidad Israel. - Propietario del formulario de contacto
¿Parece sospechoso este formulario? Informe
Google Formularios

ANEXO 3

DISEÑO DE LA PROPUESTA (AGENTE “CAI”)

Procedimiento CAI en CI/CD

Área de Seguridad de la Información

Documento de Proceso y Procedimiento — Integración del agente CAI en el ciclo de vida del desarrollo de software (SDLC) y su uso en CI/CD.

Nombre del proceso	Integración de agente CAI en el ciclo de vida del desarrollo de software (SDLC)
Código	PR-CAI-001
Versión	1.0
Fecha	28/08/2025
Área responsable	Área de Seguridad de la Información
Aprobado por	CTO — David Cingolani

1. Objetivo

Establecer un marco formal para integrar el agente autónomo de pentesting ético (CAI) en los flujos de CI/CD de proyectos con frontend en **React**, backend en **Node.js** y base de datos **PostgreSQL**, garantizando detección temprana de vulnerabilidades, trazabilidad de evidencias y mejora continua. Alineado con **ISO/IEC 42001**, **NIST SSDF** y **OWASP**.

La orquestación del CAI combina análisis estático (SAST), dinámico (DAST), auditoría de dependencias (SBOM) y escaneo de secretos, priorizando riesgos con criterios CVSS y aplicando puertas de calidad (bloqueo, advertencia o notificación) dentro del pipeline CI/CD.

2. Alcance

Aplica a los desarrollos bajo supervisión de la Dirección de Tecnología, involucrando a Desarrollo, QA de Seguridad, Equipo DevOps y Seguridad de la Información. Se excluyen sistemas *legacy* no soportados y pilotos fuera del pipeline oficial.

3. Entradas

- Código fuente en repositorios GitHub Enterprise (PRs).
- Dependencias npm desde registries autorizados y SBOM en CycloneDX.
- Artefactos Docker validados en Harbor.
- Definición de pipeline CI/CD en GitHub Actions.
- Políticas del DevSecOps Handbook 2024.

4. Actividades

Nº	Actividad	Responsable	Herramientas / Evidencias	Frecuencia
1	**Revisión de PR y activación del pipeline.** El Frontend Engineer valida cambios, genera el PR y activa automáticamente el pipeline CI/CD.	David Sandoval — Frontend Engineer	GitHub Actions; repositorio Git (PR y checks)	Semanal (cada PR)
2	**SAST y SBOM.** QA ejecuta Semgrep (React/Node) y genera reportes SARIF; produce SBOM CycloneDX.	Equipo de Seguridad de la Información (QA)	Semgrep; SARIF; CycloneDX	Semanal
3	**Dependencias y secretos.** QA ejecuta `npm audit` y Gitleaks, valida hallazgos críticos y almacena reportes.	Equipo de Seguridad de la Información (QA)	npm audit; Gitleaks (gitleaks.json)	Semanal
4	**DAST en staging.** DevOps despliega la app en entorno de prueba y ejecuta OWASP ZAP Baseline.	Sebastian Morales — DevOps Engineer	OWASP ZAP Baseline (reporte)	Semanal
5	**Orquestación con CAI.** Security Lead consolida SAST/DAST/SBOM, prioriza por CVSS y aplica políticas (`.cai/policy.json`).	Victor Andres Rigacci — Security Lead	Agente CAI; reporte consolidado	Semanal
6	**Reportes y tickets.** CAI publica resultados estructurados y crea tickets en Jira asignados a los squads.	Victor Andres Rigacci — Security Lead	Jira; reportes JSON/XML	Semanal
7	**Gate de criticidad.** Comité de Cambios evalúa hallazgos críticos y decide bloqueo/advertencia.	Nicolas Lercari — Comité de Cambios	Acta de Comité; notificaciones CI/CD	Semanal
8	**Retroalimentación y políticas.** Security Lead revisa métricas, ajusta umbrales y actualiza el manual DevSecOps.	Victor Andres Rigacci — Security Lead	Políticas actualizadas; dashboard métricas	Semanal

5. Salidas

- Reportes de vulnerabilidades (JSON/XML) archivados internamente.
- Tickets en Jira asignados automáticamente a los squads.
- Métricas de seguridad (MTTR, severidad, cobertura) en panel trimestral.
- Actas del Comité de Cambios archivadas en Governance/Seguridad.

6. Responsables

- **David Sandoval** — Frontend Engineer: prepara y entrega PRs.
- **Sebastian Morales** — DevOps Engineer: despliegues y DAST.
- **Victor Andres Rigacci** — Security Lead: orquestación CAI y políticas.
- **Nicolas Lercari** — Comité de Cambios: decide bloqueos/liberaciones.
- **David Cingolani** — CTO: aprobación y supervisión del procedimiento.

7. Control del documento

Revisión semestral o ante cambios relevantes en pipeline, CAI o políticas. Revisión a cargo de Seguridad de la Información; aprobación por el CTO y el Comité de Seguridad de la Información.

ANEXO 4

YAML PARA CI.

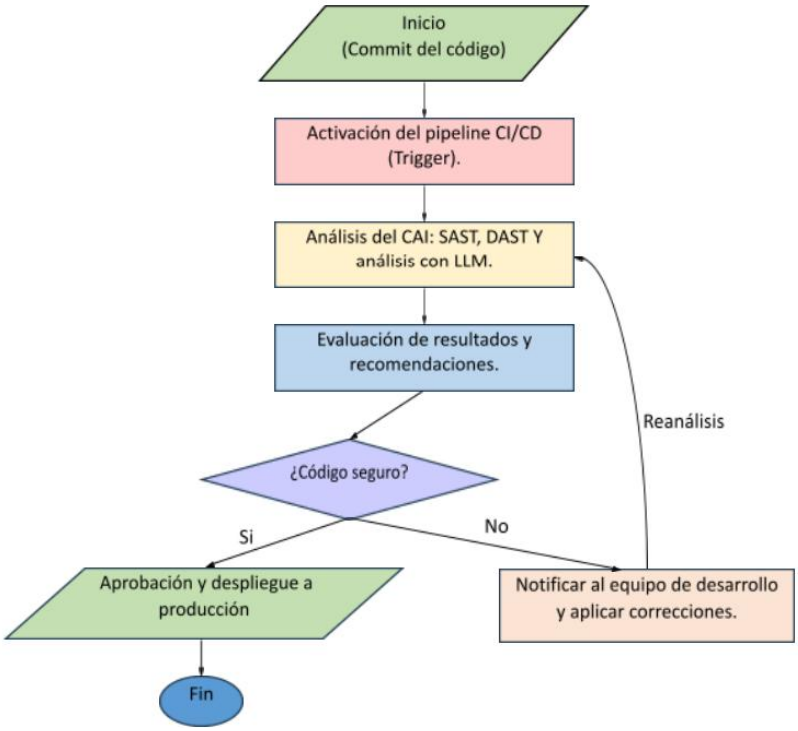
```yaml
name: ci-cd-react-with-cai

on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]

# Permisos mínimos: lectura de contenidos y publicación de resultados de seguridad (SARIF)
permissions:
  contents: read
  security-events: write

env:
  NODE_VERSION: 20.x      # Versión de Node usada por el proyecto
  CI: true

jobs:
  build-test-scan:
    name: Build, Test & Security Scans
    runs-on: ubuntu-latest

    steps:
      # 1) Clonar el repo
      - name: Checkout
        uses: actions/checkout@v4

      # 2) Instalar Node y cache de npm
      - name: Use Node.js ${{ env.NODE_VERSION }}
        uses: actions/setup-node@v4
        with:
          node-version: ${{ env.NODE_VERSION }}
          cache: 'npm'

      # 3) Instalar dependencias de forma determinística
      - name: Install dependencies
        run: npm ci

      # 4) Lint (si tu proyecto define el script)
      - name: Lint
        run: npm run lint --if-present

      # 5) Tests (si tu proyecto define el script)
      - name: Tests
        run: npm test --if-present -- --ci --reporters=default --watch=false
```

```yaml
      # 6) Build de producción (si tu proyecto define el script)
      - name: Build
        run: npm run build --if-present

      # 7) Generar SBOM con CycloneDX (permite inventario de dependencias)
      - name: Generate SBOM (CycloneDX)
        run: |
          npx @cyclonedx/cyclonedx-npm --output-format json --output-file sbom.json
        continue-on-error: true  # no bloquea el pipeline si falla

      # 8) Auditoría de dependencias (no bloqueante por falsos positivos)
      - name: Audit dependencies
        run: npm audit --audit-level=high || true

      # 9) Detección de secretos con Gitleaks (recomendado)
      - name: Secret scan (gitleaks)
        uses: gitleaks/gitleaks-action@v2
        with:
          args: detect --no-color --redact --verbose --report-format json --report-path gitleaks.json
        continue-on-error: true

      # 10) SAST con Semgrep OSS (sube SARIF a la pestaña Security de GitHub)
      - name: Static AppSec (Semgrep OSS)
        uses: returntocorp/semgrep-action@v1
        with:
          generateSarif: true
          config: p/ci            # conjunto de reglas mantenido por Semgrep
        continue-on-error: true   # evitar bloqueos por ruido inicial

      - name: Upload SARIF
        uses: github/codeql-action/upload-sarif@v3
        with:
          sarif_file: semgrep.sarif
        if: always()              # siempre sube el SARIF si existe

      # 11) (Opcional) DAST contra una URL de preview (ZAP Baseline)
      - name: DAST (OWASP ZAP Baseline) - optional
      # Ejecutar solo en PR y si hay PREVIEW_URL definida (recomendado)
        if: ${{ github.event_name == 'pull_request' && secrets.PREVIEW_URL != '' }}
        uses: zaproxy/action-baseline@v0.11.0
        with:
          target: ${{ secrets.PREVIEW_URL }}
          cmd_options: '-a -I'    # modo agresivo (-a), ignora fallos de salida (-I)
        continue-on-error: true
```

```
      # 12) Orquestación CAI: consolida hallazgos y aplica políticas (gates)
      - name: CAI — Orquestación y Puerta de Calidad
        # Reemplaza por la imagen real de tu agente CAI
        uses: docker://ghcr.io/example/cai-agent:latest
        env:
          CAI_API_KEY: ${{ secrets.CAI_API_KEY }}         # token del agente
          CAI_MODEL: gpt-4o                               # modelo a usar (ejemplo)
          CAI_POLICY_JSON: '.cai/policy.json'             # archivo con umbrales
        with:
          args: |
            --inputs sbom.json gitleaks.json semgrep.sarif zap_report.json              --output
out/cai-report.json               --tickets ${{ secrets.TICKETS_WEBHOOK_URL }}              --
thresholds-from-file .cai/policy.json

      # 13) Publicar artefactos para auditoría/descarga
      - name: Publicar artefactos CAI
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: cai-artifacts
          path: |
            out/cai-report.json
            sbom.json
            gitleaks.json
            semgrep.sarif

      # 14) Gate final: falla el build si la severidad máxima excede el umbral
      - name: Gate por criticidad (fail on HIGH/CRITICAL)
        run: |
          node -e '
            const fs = require("fs");
            // Leer políticas
            const pol = JSON.parse(fs.readFileSync(".cai/policy.json","utf8"));
            // Leer reporte CAI consolidado
            const rep = JSON.parse(fs.readFileSync("out/cai-report.json","utf8"));
            const max = rep.summary?.maxSeverity || "NONE"; // severidad máxima
            const order = ["NONE","LOW","MEDIUM","HIGH","CRITICAL"];
            // Comparar contra el umbral global failOn
            if (order.indexOf(max) >= order.indexOf(pol.failOn || "HIGH")) {
              console.error(`CAI Gate: build failed por severidad ${max} (umbral ${pol.failOn})`);
              process.exit(1);
            } else {
              console.log(`CAI Gate: build passed (max ${max}, umbral ${pol.failOn})`);
            }
          '

  # Job de despliegue (placeholder). Depende de pasar el gate.
  deploy:
    needs: [ build-test-scan ]
    if: github.ref == 'refs/heads/main'
    runs-on: ubuntu-latest
    steps:
      - name: Deploy (placeholder)
        run: echo "Despliegue aquí (Vercel/Netlify/S3/CloudFront)."
```

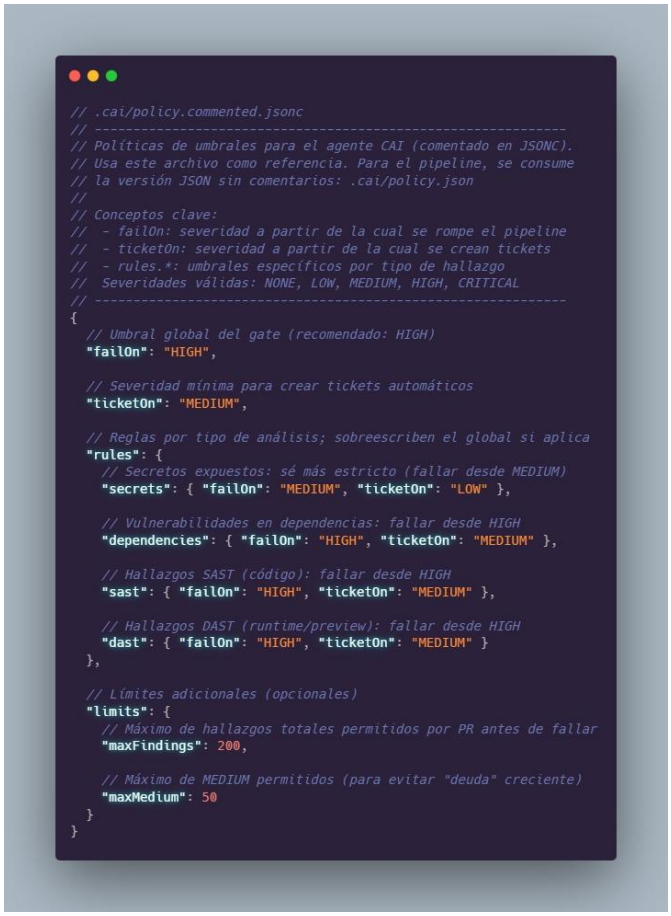
ANEXO 5

DIAGRAMA DE FLUJO EXPLICATIVO DE LA FASE DE MECANISMO DE ANÁLISIS Y ORQUESTACIÓN CON IA.



ANEXO 6

POLÍTICAS DE UMBRALES PARA EL AGENTE CAI.



```jsonc
// .cai/policy.commented.jsonc
// ------------------------------------------------------------
// Políticas de umbrales para el agente CAI (comentado en JSONC).
// Usa este archivo como referencia. Para el pipeline, se consume
// la versión JSON sin comentarios: .cai/policy.json
//
// Conceptos clave:
//  - failOn: severidad a partir de la cual se rompe el pipeline
//  - ticketOn: severidad a partir de la cual se crean tickets
//  - rules.*: umbrales específicos por tipo de hallazgo
//  Severidades válidas: NONE, LOW, MEDIUM, HIGH, CRITICAL
// ------------------------------------------------------------
{
  // Umbral global del gate (recomendado: HIGH)
  "failOn": "HIGH",

  // Severidad mínima para crear tickets automáticos
  "ticketOn": "MEDIUM",

  // Reglas por tipo de análisis; sobreescriben el global si aplica
  "rules": {
    // Secretos expuestos: sé más estricto (fallar desde MEDIUM)
    "secrets": { "failOn": "MEDIUM", "ticketOn": "LOW" },

    // Vulnerabilidades en dependencias: fallar desde HIGH
    "dependencies": { "failOn": "HIGH", "ticketOn": "MEDIUM" },

    // Hallazgos SAST (código): fallar desde HIGH
    "sast": { "failOn": "HIGH", "ticketOn": "MEDIUM" },

    // Hallazgos DAST (runtime/preview): fallar desde HIGH
    "dast": { "failOn": "HIGH", "ticketOn": "MEDIUM" }
  },

  // Límites adicionales (opcionales)
  "limits": {
    // Máximo de hallazgos totales permitidos por PR antes de fallar
    "maxFindings": 200,

    // Máximo de MEDIUM permitidos (para evitar "deuda" creciente)
    "maxMedium": 50
  }
}
```

ANEXO 7

ESTRUCTURA GENERAL DE LA PROPUESTA CAI

Estructura General de la Propuesta con CAI en CI/CD

Procesos, Herramientas, Entradas/Salidas, Roles y Criterios

Autor: Leonardo Antonio Sánchez López
Fecha: 03 de September de 2025

Introducción

Este documento describe, con precisión y en un lenguaje comprensible para terceros, la estructura general de una propuesta para integrar un agente CAI en pipelines de CI/CD. Se detallan procesos, herramientas, entradas y salidas, roles, criterios de aceptación, riesgos y mitigaciones, con el objetivo de facilitar su adopción y escalamiento.

1. Contextualización de la problemática y riesgos

Objetivo

Establecer el diagnóstico técnico-organizacional y el marco de referencia que justifica el agente CAI en un pipeline CI/CD.

Alcance

Repositorios de software (backend/frontend), pipelines de CI/CD, prácticas de seguridad, gobernanza y adopción del cambio.

Entorno base (referencial)

SO/Infra: Linux (Ubuntu LTS), Docker/Podman, contenedores inmutables.

Repositorios: GitHub/GitLab, ramas main/develop, PR/MR obligatorios.

CI/CD: GitHub Actions / GitLab CI / Jenkins (cualquiera).

Lenguajes típicos: Python, JavaScript/TypeScript.

Artefactos: imágenes Docker, reportes SARIF/JSON, badges de estado.

Procesos (paso a paso)

1. **Revisión del estado del arte:** búsqueda sistemática (bases científicas, blogs técnicos, OWASP, NIST); clasificación de enfoques (SAST, DAST, SCA, LLM-assisted).
2. **Levantamiento de contexto:** inventario de repos, pipelines, dependencias, secretos, políticas de ramas, cobertura de pruebas, criticidad de servicios.
3. **Identificación de riesgos:**
    - **Técnicos:** dependencias vulnerables, exposición de secretos, mala configuración de contenedores/red, baja cobertura de tests, pipeline drift.

- **Organizacionales:** resistencia al cambio, gaps de capacitación, falta de ownership, tiempos de entrega rígidos.

4. **Priorización y plan de mitigación:** matriz probabilidad/impacto; definición de quick wins (p.ej., activar Semgrep básico, bloqueos de secrets).
5. **Definición de objetivos de seguridad:** metas medibles (↓MTTR, ↓falsos positivos, ↑cobertura SAST/DAST, políticas de quality gates).

Herramientas

Gestor de conocimiento (Notion/Confluence), encuestas (Google Forms/Qualtrics), análisis estático inicial (Semgrep rulesets base), escaneo de dependencias (SCA: npm audit/pip-audit), detección de secretos (gitleaks/trufflehog).

Datos de entrada

Repos y pipelines actuales, políticas internas, resultados preliminares de escaneo, encuestas a devs/DevOps/Sec.

Salidas/entregables

Informe de diagnóstico, matriz de riesgos, mapa de madurez, hoja de ruta de seguridad (roadmap trimestral).**Roles y responsabilidades**

- **Líder técnico/arquitecto:** guía técnica, criterios de aceptación.
- **DevSecOps:** integra herramientas y automatiza.
- **Equipo de desarrollo:** proporciona contexto de código y revisa PRs.
- **Seguridad (AppSec):** establece políticas, revisa hallazgos críticos.
- **Gestión (PM/PO):** prioriza mitigaciones y asigna capacidad.

Criterios de aceptación (ejemplos)

- Diagnóstico revisado por AppSec y Tech Lead.
- Riesgos categorizados con acciones y responsables.
- Hoja de ruta aprobada con objetivos trimestrales.

Riesgos comunes y mitigaciones

- **Falsos positivos sobreabundantes →** afinar reglas, whitelists contextuales.
- **Sobrecarga de alertas →** umbrales y puertas de calidad progresivas.
- **Resistencia al cambio →** capacitación corta, guías de corrección tipo playbook.Supuestos y restricciones

Conexión a repos, runners disponibles, sin datos personales sensibles en logs; cumplimiento con políticas internas (ISO/IEC 27001:2022) y, si aplica, IA responsable (alineable a ISO/IEC 42001).

2. Diseño de la propuesta: agente "CAI"

Objetivo

Definir la arquitectura modular (Entrada → Procesamiento → Salida) e integrar análisis tradicionales con LLM.

Arquitectura (alto nivel)

Entrada: código fuente, manifests (package.json, requirements.txt), Dockerfiles, variables de entorno, imágenes preview.

Procesamiento:

- **SAST:** Semgrep (reglas security, best-practices y reglas internas).
- **DAST:** OWASP ZAP (scan activo/passive), rutas críticas y autenticación.
- **Contextual-LLM:** normaliza hallazgos (SARIF/JSON) → prompt seguro → explicación priorizada y fix hints.

Salida: reporte unificado (SARIF/HTML/JSON), comentarios en PR, etiquetas (severity, CWE, CVSS), badges de estado y registro histórico.

Procesos (paso a paso)

1. **Modelado de componentes:** flujos de datos, contratos entre fases (formatos de entrada/salida).
2. **Definición de quality gates:** p.ej., bloquear merge si CVSS ≥ 7.0, o >N hallazgos HIGH.
3. **Selección y afinamiento de reglas SAST/DAST:** partir de rulepacks oficiales y crear reglas internas (nombres, patrones, falsos positivos conocidos).
4. **Integración con CI/CD:** jobs secuenciales/paralelos, artifacts, condicionantes por rama (PR vs main).
5. **Diseño LLM seguro:** prompts estables, redacción clara, anonimización de secretos/PII, guardrails ante hallucinations.
6. **Trazabilidad:** versionar configuraciones (yaml, rulepacks), almacenar reportes y métricas.

Herramientas clave

- **SAST:** Semgrep (+ reglas personalizadas).

- **DAST:** OWASP ZAP (con authenticated scans).
- **SCA/Secret scanning:** gitleaks, trufflehog, npm-audit/pip-audit.
- **CI/CD:** GitHub Actions / GitLab CI / Jenkins.
- **LLM API:** proveedor de LLM (p. ej., GPT-4-class), prompt templates, rate limiting, retry exponencial.
- **Formato de resultados:** SARIF/JSON + HTML legible para humanos.
- **Gestión de configuración:** YAML versionado, env vars con secretos en vault.

Datos de entrada

PR/MR, diffs, árbol del repo, registros del runtime en entorno de pruebas, credenciales de test.

Salidas/entregables

Comentarios en PR con explicación y fix-hints, reporte por build, métricas agregadas por sprint/release.

Criterios de calidad

- Tasa de falsos positivos ≤ X% (definido).
- Cobertura de rutas críticas en DAST ≥ Y%.
- Quality gate aplicado en ramas protegidas.
- Latencia razonable del pipeline (p. ej., <12 min en PR estándar).

Ejemplo mínimo de job (ilustrativo)

```
jobs:
  security-checks:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: SAST (Semgrep)
        run: semgrep --config p/ci --sarif -o reports/semgrep.sarif
      - name: DAST (ZAP baseline)
        run: zap-baseline.py -t $TARGET_URL -J reports/zap.json
      - name: Consolidar
        run: python scripts/merge_reports.py reports/*.sarif reports/zap.json
> reports/unified.json
      - name: LLM Contextual
        run: python scripts/llm_explainer.py --in reports/unified.json --out
reports/human_readable.html
```

3. Mecanismo de análisis y orquestación con IA

Objetivo

Ejecutar análisis automáticos por evento de código y producir reportes priorizados y comprensibles.

Procesos (paso a paso)

1. **Activación automática:** webhook de PR/commit dispara el pipeline; caching para acelerar análisis repetidos.
2. **Orquestación de análisis:**
    - SAST primero (rápido y barato).
    - DAST en entorno preview o staging (con autenticación si aplica).
    - Secret/SCA en paralelo.
3. **Normalización y priorización:** convertir a formato común, agrupar duplicados, calcular CVSS (base/temporal/ambiental), marcar regressions.
4. **Explicación contextual (LLM):**
    - Sanitizar entradas (eliminar secretos/PII, truncar a budget de tokens).
    - Prompting estructurado: contexto del proyecto → hallazgo → why it matters → how to fix → example patch.
    - Guardrails: límites de longitud, evidencia requerida (p. ej., show the code line), negarse si falta contexto suficiente.
5. **Publicación de resultados:** comentarios en PR (anclados a líneas), reporte HTML para stakeholders, salida SARIF para IDEs/SCMs.
6. **Ciclo de mejora:** etiquetar falsos positivos, feedback loop para afinar reglas y prompts; dashboards (MTTD/MTTR, severidades por repo).

Herramientas

Colas/eventos del CI, runners autoscalables, almacenamiento de artefactos, API LLM con rate limit y retries, conversor SARIF ↔ HTML, badges de status.

Métricas operativas y de valor

- **MTTD/MTTR** (detección y remediación).
- **FPR/TPR** (falsos verdaderos positivos).

- **Severidad promedio** por PR/release; densidad de hallazgos por KLOC.
- **Cobertura DAST** de rutas críticas; % PRs bloqueados por quality gate.
- **Tendencia** (7/30 días): ¿baja la severidad? ¿baja el MTTR?

Criterios de aceptación (ejemplos)

- Reporte consolidado y comentario en PR en < X min.
- Priorización por CVSS correcta y consistente.
- Sugerencias de corrección verificables (snippet o referencia).

Riesgos y mitigaciones específicos de IA

- **Alucinaciones del LLM** → exigir evidencia (línea de código/stack trace), plantillas con checklist, revisión humana en hallazgos críticos.
- **Exposición de secretos** → sanitización previa y deny-list de patrones; uso de entornos de prueba.
- **Costos/latencia** → batching de hallazgos, caching, timeout y fall-back a explicación breve.

4. Validación de la metodología

Objetivo

Demostrar pertinencia, utilidad y adoptabilidad con juicio experto.

Procesos (paso a paso)

1. **Diseño del instrumento:** cuestionario con escalas Likert y preguntas abiertas (claridad de reportes, relevancia de priorización, utilidad de fix-hints, impacto en flujo).
2. **Muestreo de expertos:** AppSec, DevSecOps, Dev leads; ≥ 2 por cada disciplina.
3. **Aplicación y recolección:** formulario digital (Google Forms/Qualtrics), anonimato opcional; período mínimo 1 semana.
4. **Análisis:** descriptivo (promedios, desviaciones), codificación temática en respuestas abiertas; mapa de fortalezas y mejoras.
5. **Plan de acción:** incorporar sugerencias (p. ej., nuevas reglas, umbrales revisados, formatos de reporte).

6. **Cierre y revalidación:** publicar resultados y repetir medición tras 4–6 semanas para verificar mejora.

Herramientas

Forms/Qualtrics, hoja de cálculo/BI (Looker/Metabase), repositorio de evidencias (informes antes/después).

Entradas

Reportes reales de 2–3 PRs por servicio representativo.

Salidas

Informe de validación, lista de mejoras, decisión de go/live por etapas.

Criterios de aceptación

- ≥ 80% de expertos califican la utilidad como "Alta/Muy alta".
- Reducción observada de MTTR o severidad promedio en el periodo piloto.
- Aprobación de AppSec para despliegue por defecto en repos críticos.

Checklist de implementación (resumen operativo)

1. Inventariar repos/pipelines y activar branch protection.
2. Añadir secret scanning y SCA básicos.
3. Integrar Semgrep con reglas base y reporte SARIF.
4. Preparar entorno preview para ZAP (auth incluida).
5. Consolidar resultados (SARIF/JSON) en artifact único.
6. Diseñar prompt template del LLM (con sanitización).
7. Publicar comentarios en PR (+ HTML para stakeholders).
8. Definir quality gates y aplicar en ramas protegidas.
9. Medir MTTD/MTTR, FPR/TPR y cobertura DAST.
10. Ejecutar validación con expertos y ajustar reglas/umbrales.
11. Documentar playbooks de corrección por tipo de hallazgo.
12. Expandir a más repos y automatizar dashboards.

ANEXO 8

FORMATO DE VALIDACIÓN DE ESPECIALISTAS UISRAEL: MGST. ANDRÉS LOJA

Universidad Israel
ESPOG | Escuela de Posgrados

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMATICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido "Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA". Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Andrés Fernando Loja Morocho
Título obtenido: Magister en seguridad de la información.
C.I.: 0107137416
E-mail: andres.loja@bestpointauditores.com
Institución de Trabajo: BestPoint
Cargo: Auditor de sistemas
Años de experiencia en el área: 4 años

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.

Universidad Israel
ESPOG | Escuela de Posgrados

- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia	X				
Aplicabilidad	X				
Factibilidad	x				
Novedad		X			
Fundamentación pedagógica		X			
Fundamentación tecnológica		X			
Indicaciones para su uso		X			
TOTAL	15	16			

Observaciones: La propuesta se considera funcional y útil, especialmente para proyectos que manejan datos sensibles. Se observa que la calidad del código generado por IA depende en gran medida de la calidad del prompt ingresado por el usuario, por lo que una herramienta de verificación es clave.

Recomendaciones: La principal recomendación es asegurar la precisión del agente. Es crucial que la herramienta "especifique exactamente dónde está el problema y no dé falsos positivos o falsos negativos" para ser efectiva. Además, debe ser una herramienta que "transmita confianza" a los desarrolladores para que sea adoptada.

Lugar, fecha de validación: Cuenca, agosto de 2025

Ing. Andrés Fernando Loja Morocho

ANEXO 9

FORMATO DE VALIDACIÓN DE ESPECIALISTAS UISRAEL: MGST. LILIANA CAJAS

Universidad Israel
ESPOG | Escuela de Posgrados

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMATICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido "Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA". Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Liliana Magaly Cajas Chuqui
Título obtenido: Magister en seguridad de la información.
C.I.: 0302647615
E-mail: lcajas@goldenaudit.com.ec
Institución de Trabajo: Golden auditores
Cargo: Auditor de sistemas
Años de experiencia en el área: 4 años

Universidad Israel
ESPOG | Escuela de Posgrados

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.
- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia	X				
Aplicabilidad		X			
Factibilidad			X		
Novedad		X			
Fundamentación pedagógica	X				
Fundamentación tecnológica	X				
Indicaciones para su uso	X				
TOTAL	20	8	3		

Observaciones: La solución es vista como "muy necesaria" para optimizar los procesos de seguridad. Se anticipan desafíos técnicos relacionados con la "integración a los diferentes pipelines" y la necesidad de que la herramienta sea "configurable" para adaptarse a cada proyecto.

Recomendaciones: La recomendación principal es enfocarse en la "integración con herramientas de gestión de tickets" para automatizar el flujo de trabajo de remediación de vulnerabilidades. Esto aseguraría que los hallazgos no se pierdan y sean gestionados adecuadamente.

Lugar, fecha de validación: Cuenca, agosto de 2025

LILIANA MAGALY CAJAS CHUQUI

Ing. Liliana Magaly Cajas Chuqui

ANEXO 10

FORMATO DE VALIDACIÓN DE ESPECIALISTAS UISRAEL: IGN. MAURICIO AGUILERA

Universidad Israel
ESPOG | Escuela de Posgrados

UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSGRADOS "ESPOG"

MAESTRÍA EN SEGURIDAD INFORMATICA

INSTRUMENTO PARA VALIDACIÓN DE LA PROPUESTA

Estimado colega:

Se solicita su valiosa cooperación para evaluar la calidad del siguiente contenido **"**Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA". Sus criterios son de suma importancia para la realización de este trabajo, por lo que se le pide que brinde su cooperación contestando las preguntas que se realizan a continuación.

Datos informativos

Validado por: Mauricio Aguilera
Título obtenido: Ingeniero en Tecnología de la Información
C.I.: 0704482223
E-mail: maguilera0810@gmail.com
Institución de Trabajo: Equifax
Cargo: Desarrollador de Software
Años de experiencia en el área: 5 años

Universidad Israel
ESPOG | Escuela de Posgrados

Instructivo:

- Responda cada criterio con la máxima sinceridad del caso.
- Revisar, observar y analizar la propuesta de la plataforma virtual, blog o sitio web.
- Coloque una X en cada indicador, tomando en cuenta que Muy adecuado equivale a 5, Bastante Adecuado equivale a 4, Adecuado equivale a 3, Poco Adecuado equivale a 2 e Inadecuado equivale a 1.

Tema: "Proceso sistemático para optimizar la seguridad del SDLC mediante un agente autónomo de pentesting ético, mitigando riesgos y vulnerabilidades en código asistido por IA"

Indicadores	Muy adecuado	Bastante Adecuado	Adecuado	Poco adecuado	Inadecuado
Pertinencia	X				
Aplicabilidad		X			
Factibilidad		X			
Novedad		X			
Fundamentación pedagógica	X				
Fundamentación tecnológica	X				
Indicaciones para su uso		X			
TOTAL	15	16			

Observaciones: Se confirma que la problemática es real, especialmente "en casos donde se confía todo a la IA". La solución es pertinente para los procesos de QA y seguridad. Sin embargo, se anticipan desafíos de compatibilidad con la gran variedad de frameworks y lenguajes existentes.

Recomendaciones: La principal barrera identificada es económica. Se recomienda enfocar la aplicabilidad de la herramienta en sectores de alta criticidad, ya que "requiere recursos" y sería más "factible para proyectos con uso delicado de datos como entidades bancarias"

Lugar, fecha de validación: Pasaje, agosto de 2025

Ing. Mauricio Aguilera