Red teaming de aplicaciones LLM

Auditoría de modelos, RAG y agentes. Metodología, taxonomía de ataques, automatización de pruebas y métricas.

Iker Foronda Carrasco

Nombre del Programa
Seguridad en redes y sistemas

Nombre Tutor/a de TF
Francisco José Ramírez Vicente
Profesor/a responsable de la asignatura
Jordi Serra Ruiz

Fecha Entrega
05/06/2026

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Red teaming de aplicaciones LLM* *Auditoría de modelos, RAG y agentes. Metodología, taxonomía de ataques, automatización de pruebas y métricas.*
Nombre del autor:	*Iker Foronda Carrasco*
Nombre del consultor/a:	*Francisco José Ramírez Vicente*
Nombre del PRA:	*Jordi Serra Ruiz*
Fecha de entrega (mm/aaaa):	*06/2026*
Titulación o programa:	*Master en Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Seguridad en redes y sistemas*
Idioma del trabajo:	*Castellano*
Palabras clave	*Ciberseguridad, Inteligencia Artificial*

Resumen del Trabajo

El despliegue masivo de sistemas basados en modelos de lenguaje de gran tamaño (LLMs), combinado con los mecanismos para enriquecer su funcionamiento, como los sistemas de recuperación aumentada de información (RAG) o agentes con capacidad de ejecutar herramientas, introducen una superficie de ataque que las metodologías de auditoría de seguridad tradicionales no cubren de forma adecuada.

Vulnerabilidades como la inyección de prompts directa e indirecta, el envenenamiento de RAG, el abuso de herramientas o el escalado de permisos en sistemas agénticos carecen actualmente de un marco de evaluación formal, reproducible y orientado a evidencias.

Este trabajo propone el diseño, implementación y validación de una metodología de red teaming ofensivo aplicable de forma progresiva a tres niveles: LLMs en modo *standalone*, aplicaciones LLM con RAG y agentes con herramientas. Para complementar la metodología, se desarrollará una taxonomía estructurada de ataques específicos para cada capa y un framework de automatización de pruebas que permitirá medir, reproducir y reportar vulnerabilidades de forma sistemática.

La validación experimental será realizada sobre entornos controlados que reproduzcan los tres niveles de despliegue, utilizando tanto modelos de código abierto como APIs comerciales. Se definen métricas cuantitativas específicas que permitan comparar configuraciones y evaluar la efectividad de los controles de seguridad.

Abstract

The deployment of LLMs, in conjunction with the systems which consume and enhance these applications, such as RAG and agents with access to tools which can act autonomously. All these advancements broaden the attack surface of these systems, which traditional methodologies may fail to cover correctly.

Vulnerabilities such as indirect or direct prompt injection, RAG poisoning, tool abuse or privilege escalation in agentic systems lack reproducible and evidence oriented formal evaluation frameworks.

This work aims to provide the design, implementation and validation of this red teaming methodology, based on a progressive approach along three levels: standalone LLM deployments, RAG applications and agents with tools. In order to enhance the methodology, a structured taxonomy of specific attacks will be developed, combined with an automated testing framework to provide systematical reproducible measurements and reporting.

The experimental phase will be carried within controlled environments that reproduce the three deployment styles, using both open source models and API based commercial offerings. Quantitative metrics will be defined in order to allow evaluation of the safety checks and the comparison between configurations.

Índice

Lista de figuras

1. Introducción

La adopción acelerada de modelos de lenguaje de gran tamaño (LLMs, del inglés Large Language Models) en entornos productivos ha transformado de forma significativa el panorama del desarrollo de software. Lo que comenzó como interfaces conversacionales ha evolucionado rápidamente hacia arquitecturas complejas: sistemas de recuperación aumentada de información (RAG, Retrieval-Augmented Generation) que integran bases de conocimiento externas, y agentes autónomos capaces de planificar, razonar y ejecutar acciones sobre herramientas reales como sistemas de ficheros, APIs, bases de datos o navegadores web. Más recientemente, la adopción del protocolo MCP (Model Context Protocol) está acelerando la integración de capacidades agénticas en entornos de producción.

Esta evolución introduce una superficie de ataque cualitativamente distinta a la del software tradicional. A diferencia de una aplicación web convencional, cuyo comportamiento es determinista y auditable mediante técnicas establecidas, un sistema basado en LLMs presenta características que dificultan su evaluación de seguridad: comportamiento no determinista, fronteras difusas entre datos e instrucciones, dependencia de fuentes de conocimiento externas potencialmente no confiables, y capacidad de actuar sobre el entorno con consecuencias reales. Vulnerabilidades como la inyección de prompts directa e indirecta, el envenenamiento de índices RAG, el abuso de herramientas o el escalado de permisos en cadenas agénticas no tienen equivalente directo en las metodologías de auditoría clásicas.

1.1. Contexto y justificación del Trabajo

El campo de la seguridad en IA, aunque en rápida expansión, presenta aún una fragmentación notable. Iniciativas como el OWASP Top 10 for LLM Applications [1], el framework MITRE ATLAS [2] o el NIST AI Risk Management Framework [3] ofrecen taxonomías y marcos de referencia valiosos, pero no constituyen metodologías operativas completas. Las herramientas de red teaming disponibles cubren parcialmente el espacio de amenazas. Garak [4] proporciona una amplia biblioteca de *probes* para evaluar LLMs en modo *standalone*, con soporte para *jailbreaks*, extracción de información y detección de toxicidad, pero sin cobertura de las capas RAG ni agéntica. PyRIT [5], desarrollado por Microsoft, introduce un enfoque orientado a riesgo y soporta escenarios algo más complejos, pero su integración con sistemas RAG o agentes con herramientas es limitada y no está documentada como metodología formal. Promptfoo [6] destaca por su integración en pipelines CI/CD y su facilidad de uso, pero está orientado principalmente a la evaluación de calidad de salidas más que a la auditoría de seguridad. Ninguna de estas herramientas aborda de forma integrada la capa agéntica con herramientas reales ni proporciona un marco de métricas cuantitativas comparables entre sistemas.

Esta evolución introduce vulnerabilidades cualitativamente distintas a las del software tradicional. Perez y Ribeiro [7] demostraron que los LLMs pueden ser manipulados para ignorar sus instrucciones de sistema mediante prompt injection directa. Greshake et al. [8] extendieron este concepto al escenario de indirect prompt injection: un atacante puede embeber instrucciones maliciosas en contenido externo, ya sea mediante documentos, resultados de páginas web o búsquedas, las cuales son procesadas como parte de su contexto, comprometiendo la integridad de la interacción final con el usuario. Zou et al. [9] demostraron que es posible generar sufijos adversariales universales y transferibles entre modelos que producen comportamientos no deseados de forma sistemática. Carlini et al. [10] documentaron que los LLMs memorizan y pueden exfiltrar datos de entrenamiento bajo condiciones específicas de prompting. Más recientemente, en el ámbito específico de los sistemas RAG, Zou et al. [11] introdujeron el concepto de PoisonedRAG, demostrando que la inyección de documentos adversariales en el índice de recuperación permite controlar las respuestas del modelo con alta fiabilidad. En la capa agéntica, Zhan et al. [12] presentaron InjecAgent, un benchmark que evidencia que los agentes basados en herramientas son sistemáticamente vulnerables a la inyección de prompts indirecta a través de los resultados de las herramientas.

La relevancia del problema se ve amplificada por el contexto de despliegue. Según el informe Threat Landscape 2024 de ENISA [13], los sistemas de IA generativa han pasado a formar parte de la superficie de ataque crítica de organizaciones en sectores como sanidad, finanzas y administración pública. La directiva NIS2 y el AI Act europeo establecen obligaciones de evaluación de riesgos para sistemas de IA de alto impacto, sin que existan aún metodologías técnicas estandarizadas que permitan cumplir dichas obligaciones en el caso específico de LLMs y agentes.

Respecto a las categorizaciones en el sentido más académico, Weidinger et al. [14] elaboraron una taxonomía exhaustiva de riesgos de los LLMs que incluye dimensiones de seguridad, privacidad y daño social, pero sin orientación operativa para auditores y profesionales que quisieran categorizar los riesgos. Derner y Batistič [15] analizaron específicamente los riesgos de seguridad derivados de la integración de plugins y herramientas externas en LLMs, anticipando muchos de los vectores que hoy caracterizan a los sistemas agénticos.

Esta demanda creciente de aproximaciones holísticas se refleja en investigaciones recientes. Por un lado, Purpura et al. [16] coinciden en la imperiosa necesidad de aproximaciones funcionales *end-to-end* al exponer y sistematizar de forma detallada los diferentes componentes y métricas necesarias para construir aplicaciones que usen LLMs de manera segura. En adición, Verma et al. [17] ponen de manifiesto la urgencia de operativizar modelos de amenaza formales que abarquen íntegramente las etapas de desarrollo y despliegue del LLM, proponiendo una taxonomía sistemática de ataques (*Systematization of Knowledge*, SoK) que complemente las soluciones técnicas. Adicionalmente, esta ambigüedad conceptual en la industria comporta riesgos propios, tal y como como señalan Feffer et al. [18], la ausencia de

objetivos concretos, métricas y límites bien definidos puede convertir las prácticas corporativas de *red teaming* sobre IA generativa en mero "teatro de seguridad" en lugar de un proceso riguroso y efectivo para mitigar riesgos.

Este trabajo se justifica, por tanto, en la necesidad identificada de una metodología formal, progresiva y orientada a evidencias que integre el estado del arte en una aproximación unificada, auditable y reproducible para evaluar la seguridad de los tres niveles de despliegue más habituales en la industria actual, ya que no existe una metodología que integre de forma progresiva y operativa la auditoría de seguridad con métricas cuantitativas, plantillas reproducibles y soporte para herramientas y RAG.

1.2. Objetivos del Trabajo

El objetivo principal del proyecto será diseñar, implementar y validar una metodología de auditoría ofensiva (red teaming) para sistemas basados en LLMs, aplicable de forma progresiva a modelos *standalone*, aplicaciones con RAG y agentes con herramientas, acompañada de las herramientas y métricas necesarias para trazar, reproducir y reportar vulnerabilidades de forma sistemática y auditable.
Para desempeñar con éxito el proyecto, se proponen los siguientes objetivos intermedios:

ID	Objetivo	Entregable asociado
TFM-01	Revisión del estado del arte en el ámbito de la seguridad ofensiva de LLMs, RAG y agentes	Estado del arte actualizado, taxonomía de ataques de referencia
TFM-02	Desarrollar una taxonomía propia y estructurada de ataques organizada por capas (modelo, RAG, agente)	Taxonomía propia documentada
TFM-03	Diseñar una metodología de auditoría progresiva con fases, criterios de entrada/salida y plantillas de reporte	Documentos de metodología, plantillas
TFM-04	Implementar un framework de automatización de pruebas que soporte la metodología diseñada	Herramienta software
TFM-05	Definir un conjunto de métricas cuantitativas para evaluar la cobertura y severidad de las pruebas	Especificación de las métricas
TFM-06	Validar la metodología y la herramienta sobre entornos controlados que representen los	Informe de validaciones

	tres niveles de despliegue	
TFM-07	Comparar la cobertura y capacidades de la metodología propuesta frente a herramientas existentes	Análisis comparativo

Figura 1: Tabla de hitos TFM

Al completar estos objetivos, se espera haber producido:

- Una metodología documentada de red teaming para LLMs, RAG y agentes, con la aplicabilidad suficiente para ser adoptable por equipos de seguridad y auditores externos.
- Una biblioteca de casos de prueba estructurados y parametrizados, organizada según la taxonomía desarrollada.
- Un framework software funcional que automatice la ejecución de pruebas, la recolección de evidencias y la generación de reportes.
- Un conjunto de métricas y criterios de severidad adaptados a las particularidades de los sistemas basados en LLMs.

1.3. Análisis de riesgos del proyecto

Matriz de riesgos

ID	Riesgo	Probabilidad	Impacto	Exposición
R01	Cambios del estado del arte que desvíen o cambien el enfoque de la investigación	Alta	Medio	Alta
R02	Limitación en la reproducibilidad experimental debido al no determinismo	Alta	Medio	Alta
R03	Aumento de la complejidad del framework en el momento de la implementación	Media	Alto	Alta
R04	Dificultades para construir un entorno de pruebas para agentes	Media	Alto	Alta

R05	Indisponibilidad de las APIs de los proveedores principales en la experimentación	Media	Medio	Media
R06	Dificultad para encontrar fuentes científicas de calidad alta	Baja	Medio	Medio
R07	Uso indebido de las herramientas desarrolladas	Medio	Alto	Medio

Figura 2: Tabla de riesgos asociada al TFM

Estrategias de mitigación del riesgo

Una vez definidos los riesgos principales que pueden afectar a la ejecución del TFM, se elabora una lista de posibles contramedidas y mitigaciones con intención de limitar el impacto de estos en los entregables:

R01
Considerando la velocidad extrema de la investigación en IA, se asume que ocurrirán avances tecnológicos y metodológicos durante el proceso de elaboración del TFM, por lo que se mantendrá la investigación actualizada con las fuentes más relevantes hasta un momento de corte, que estará alineado con el inicio de la implementación de la herramienta.

R02
Para mitigar la variabilidad de las respuestas entregadas por los modelos y demás componentes, se reducirá al máximo posible la variabilidad mediante el ajuste de hiperparámetros de los modelos a utilizar, siempre que sea posible.

R03
Para limitar el alcance de este riesgo, se plantea un desarrollo incremental de características de la herramienta, de tal manera que la funcionalidad central pudiera ser implementada como un MVP (Minimal Viable Product), y el resto de características clave sean implementadas como extras que amplían la funcionalidad principal.

R04
El riesgo de implementación de la capa agéntica surge de la carencia de referencias para este tipo de implementaciones en el contexto de auditorías de seguridad. Para mitigar este riesgo, se pretende apalancar un *framework* existente de agentes como base, al que se le harán ajustes para adaptar el caso de uso.

R05
En las fases más avanzadas del proyecto, si se desean realizar pruebas con los proveedores de servicios de IA más relevantes, es probable que las APIs estén limitadas, o que las auditorías vayan en contra de los términos de servicio, por lo que se limitarán en la medida de lo posible, y las pruebas se realizarán en la línea de una validación de lo desarrollado, más que vinculado al desarrollo como tal.

R06
Pese a la abundancia de literatura sobre el tema, el exceso de esta tiene un efecto saturante en la atención y puede conllevar dificultades a la hora de encontrar fuentes relevantes y/o de calidad. Se mitigará este riesgo confiando en la pericia del profesorado y los tutores para guiar la investigación a fuentes de calidad.

R07
Una vez finalizado el producto software del TFM, su riesgo principal es un uso indebido de la herramienta, fuera de su alcance y propósito originales. Para mitigar el riesgo, se embeberán las salvaguardas necesarias para no subvertir el propósito original, combinado con un modelo de licenciamiento que permita el uso legítimo y pretenda prevenir el uso indebido.

1.4. Impacto en sostenibilidad, ético-social y de diversidad

Impacto ético

El desarrollo de herramientas y metodologías ofensivas conlleva una responsabilidad ética inherente. Este trabajo opera bajo los principios de divulgación responsable (responsible disclosure): todos los experimentos se realizan en entornos controlados construidos específicamente para el proyecto, sin interacción con sistemas en producción ajenos. La biblioteca de payloads y casos de prueba desarrollada se publica con una licencia que explicita su uso exclusivamente para fines defensivos, auditoría autorizada e investigación académica.

Existe además el riesgo de doble uso: una taxonomía detallada de ataques puede ser utilizada tanto para defender sistemas como para atacarlos. Este riesgo se gestiona mediante la contextualización permanente de cada técnica en términos de mitigación y la omisión deliberada de detalles que faciliten la evasión de sistemas en producción sin autorización.

En cuanto a la privacidad, los entornos de validación experimental no utilizan datos personales reales. Los datasets de prueba son sintéticos o públicos, y los modelos evaluados se ejecutan localmente cuando es posible para minimizar la exposición de datos a terceros.

Impacto social

Los sistemas basados en LLMs y agentes se están desplegando en sectores de alto impacto social: sanidad, servicios financieros, administración pública y educación. Una vulnerabilidad de inyección de prompts en un agente con acceso a un sistema de gestión hospitalaria, o el envenenamiento de un índice RAG utilizado por un asistente jurídico, puede tener consecuencias que van más allá del ámbito técnico. Este trabajo contribuye a reducir la brecha de seguridad en estos despliegues al proporcionar herramientas y metodologías accesibles a equipos de seguridad que no necesariamente son expertos en IA.

Adicionalmente, la publicación en abierto de la metodología y las herramientas tiene un impacto democratizador: organizaciones con recursos limitados como son PYMEs, instituciones públicas u organizaciones sin ánimo de lucro, pueden beneficiarse de un *framework* de auditoría que de otro modo requeriría conocimientos especializados difíciles de adquirir.

Respecto a la diversidad, el trabajo pretende democratizar el acceso a recursos sobre ciberseguridad a profesionales de todo tipo, sin discriminarlos en base a su experiencia previa ni familia profesional, con intención de hacer de la securización de los sistemas algo aplicable para todos ellos.

Impacto ambiental

El coste computacional del proyecto es moderado. La mayor parte del trabajo experimental se realiza sobre modelos de código abierto ejecutados localmente (Llama, Mistral, Qwen), evitando llamadas innecesarias a APIs comerciales. Se aplican estrategias de eficiencia energética en la fase de experimentación: ejecución en lotes, early stopping en pruebas iterativas, y reutilización de resultados cacheados.

Pese a ser un proceso fundamentalmente adverso para el medio ambiente, la metodología desarrollada podría tener, paradójicamente, un impacto ambiental positivo indirecto. Mediante la detección temprana de vulnerabilidades en los sistemas auditados, se pueden prevenir incidentes de seguridad cuya remediación (redespliegues, auditorías forenses, pérdida de datos, …) tendría costes computacionales y energéticos considerablemente mayores.

1.5. Enfoque y método seguido

El trabajo adopta un enfoque de investigación aplicada, combinando revisión sistemática de la literatura con desarrollo e implementación de artefactos software y validación experimental.

Respecto al alcance, el trabajo delimita explícitamente su objeto de estudio a los tres niveles de despliegue referenciados, puesto que son los más habituales en la industria actual (modelos *standalone*, RAG, agentes con herramientas). Quedan fuera del alcance los sistemas multimodales, el entrenamiento adversarial o los ataques sobre el proceso de fine-tuning, entre otros, para poder delimitar en un volúmen aceptable de trabajo dentro del TFM.

Sobre los entornos de pruebas y despliegues, se construirán entornos controlados y reproducibles para cada nivel, preferiblemente documentados como código. Esto permite que cualquier investigador pueda replicar los experimentos de forma independiente.

Las métricas tendrán un enfoque cuantitativo basado en las métricas definidas, complementado con el análisis cualitativo de los casos de mayor severidad para matizar su impacto y posibles consecuencias.

Las herramientas que existen en el ecosistema tecnológico actual como Garak, PyRIT y Promptfoo se integran como componentes de referencia y comparación en la metodología. El análisis de sus capacidades y limitaciones forma parte explícita de la contribución, y los sistemas que se generen a raíz del proyecto no constituyen un sustitutivo para estas herramientas referenciadas.existentes

1.6. Planificación del Trabajo

La planificación del trabajo se alineará con la estructura de la PECs referenciadas en el plan de estudios de la asignatura vinculada al desarrollo del TFM. El desglose de tareas por objetivo en cada fase del proyecto es el siguiente, segregado por cada entregable de las PECs:

ID	Tarea	Objetivo	PEC	Inicio	Fin	Duración (días)
T01	TFM-01 · Búsqueda bibliográfica	TFM-01	PEC 1	28/02/2026	09/03/2026	10
T02	TFM-01 · Análisis de literatura	TFM-01	PEC 1	02/03/2026	14/03/2026	13
T03	TFM-01 · Redacción estado del arte	TFM-01	PEC 1	09/03/2026	23/03/2026	15
M01	**PEC1 - Planificación inicial**	**HITO**	**PEC 1**	**02/03/2026**	**02/03/2026**	**0**
T05	TFM-01 · Cierre estado del arte	TFM-01	PEC 2	09/03/2026	01/04/2026	24
T06	TFM-02 · Taxonomía L1 (modelo)	TFM-02	PEC 2	14/03/2026	23/03/2026	10
T07	TFM-02 · Taxonomía L2 (RAG)	TFM-02	PEC 2	16/03/2026	25/03/2026	10
T08	TFM-02 · Taxonomía L3 (agentes)	TFM-02	PEC 2	20/03/2026	30/03/2026	11
T09	TFM-03 · Diseño de metodología	TFM-03	PEC 2	16/03/2026	01/04/2026	17
T10	TFM-05 · Definición de métricas	TFM-05	PEC 2	23/03/2026	01/04/2026	10
M02	**PEC2 - Desarrollo (avance 1)**	**HITO**	**PEC 2**	**01/04/2026**	**01/04/2026**	**0**

T12	TFM-02 · Cierre taxonomía	TFM-02	PEC 3	01/04/2026	13/04/2026	13
T13	TFM-03 · Plantillas de reporte	TFM-03	PEC 3	01/04/2026	13/04/2026	13
T14	TFM-04 · Setup entornos de prueba	TFM-04	PEC 3	06/04/2026	17/04/2026	12
T15	TFM-04 · Implementación core CLI	TFM-04	PEC 3	13/04/2026	04/05/2026	22
T16	TFM-04 · Biblioteca de payloads	TFM-04	PEC 3	13/04/2026	04/05/2026	22
T17	TFM-05 · Cierre especif. métricas	TFM-05	PEC 3	06/04/2026	20/04/2026	15
T18	TFM-06 · Validación L1 (modelo base)	TFM-06	PEC 3	20/04/2026	04/05/2026	15
T19	TFM-06 · Validación L2 (RAG)	TFM-06	PEC 3	27/04/2026	10/05/2026	14
T20	TFM-07 · Análisis comparativo (inicio)	TFM-07	PEC 3	27/04/2026	10/05/2026	14
M03	**PEC3 - Desarrollo (avance 2)**	**HITO**	**PEC 3**	**10/05/2026**	**10/05/2026**	**0**
T22	TFM-04 · Cierre framework + docs	TFM-04	PEC 4	11/05/2026	22/05/2026	12
T23	TFM-06 · Validación L3 (agentes)	TFM-06	PEC 4	11/05/2026	22/05/2026	12
T24	TFM-06 · Cierre informe validación	TFM-06	PEC 4	18/05/2026	29/05/2026	12
T25	TFM-07 · Cierre análisis comparativo	TFM-07	PEC 4	18/05/2026	29/05/2026	12
T26	TFM-04 · Revisión integral framework	TFM-04	PEC 4	25/05/2026	02/06/2026	9
M04	**PEC4 - Entrega final y memoria**	**HITO**	**PEC 4**	**05/06/2026**	**05/06/2026**	**0**

Figura 3: Tabla de tareas Gantt

Se adjunta el diagrama Gantt que se desprende de la lista de entregables identificados y definidos en base a la tabla anterior:

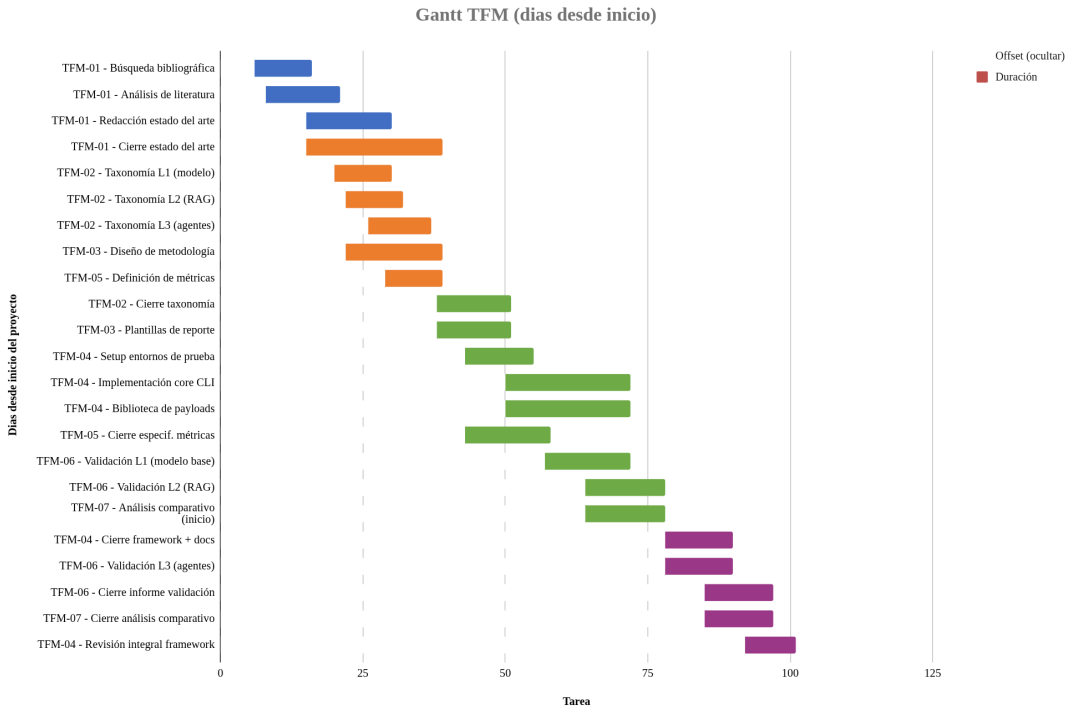

Figura 4: Gantt TFM

2. Materiales y métodos

2.1. Diseño de la metodología

El desarrollo del presente trabajo se ha planteado como una investigación aplicada con componente de ingeniería, orientada a diseñar y validar una metodología de red teaming ofensivo para aplicaciones basadas en modelos de lenguaje de gran tamaño. El enfoque metodológico se estructura de forma progresiva en tres capas técnicas, definidas como L1 (modelo standalone), L2 (sistemas con RAG) y L3 (agentes con herramientas), en coherencia con la evolución real de los despliegues en entornos productivos y con el incremento gradual de su superficie de ataque.

La base conceptual y operativa del trabajo se sostiene sobre tres ejes complementarios. En primer lugar, una revisión del estado del arte en seguridad ofensiva de IA generativa, con especial foco en inyecciones de *prompt* (directa e indirecta), ataques adversariales transferibles, extracción de información sensible, envenenamiento de conocimiento en RAG y abuso de herramientas en arquitecturas agénticas. En segundo lugar, una taxonomía propia estructurada por capas, técnicas, mitigaciones, métricas y metodología, diseñada para traducir conocimiento teórico en procedimientos auditables. En tercer lugar, se utilizan OWASP Top 10 for LLM Applications, MITRE ATLAS y NIST AI RMF 1.0 como marcos de referencia externos con el fin de asegurar trazabilidad, comparabilidad y alineación con prácticas reconocidas de gestión del riesgo.

Desde el punto de vista de diseño metodológico, se han evaluado distintas alternativas. Se descartó un enfoque monolítico de evaluación por su limitada capacidad para aislar variables y por su escasa utilidad comparativa entre arquitecturas heterogéneas. También se descartó una dependencia exclusiva de herramientas existentes del ecosistema, debido a su cobertura parcial en escenarios RAG y, especialmente, en capa agéntica. Como resultado, se adoptó un modelo híbrido: metodología propia y automatización específica, complementadas con benchmarking frente a herramientas de referencia.

El procedimiento experimental se organiza en campañas repetibles. Para cada prueba se selecciona un vector de ataque según la taxonomía, se parametriza el caso de ensayo, se ejecuta en entorno controlado y se registran evidencias técnicas para su evaluación posterior. Dado el carácter no determinista de los LLMs, se incorporan medidas de control metodológico mediante ajuste de parámetros, repetición de ejecuciones y análisis agregado de resultados. Esta estrategia permite reducir sesgos puntuales y mejorar la robustez de las conclusiones.

En términos de materiales, el trabajo se apoya en una estructura taxonómica jerárquica por capas y propósito, combinados con un conjunto de artefactos metodológicos (protocolos de test, scoring de severidad, checklist de reproducibilidad y plantillas de reporte). De esta manera, el proyecto no se limita a una recopilación bibliográfica, sino que produce un marco técnico reutilizable para auditorías ofensivas en sistemas LLM.

Como productos obtenidos en esta fase se identifican: una metodología progresiva formalizada, una taxonomía operativa de amenazas y mitigaciones, un esquema métrico para evaluación cuantitativa, y una base estructural para la automatización de pruebas en formato CLI. Estos productos se conciben con orientación práctica, de modo que puedan ser aplicados tanto en validaciones académicas como en contextos profesionales de seguridad.

Cuando procede la valoración económica, el coste principal se concentra en el esfuerzo técnico de diseño, implementación y validación, seguido del coste computacional asociado a campañas de prueba y, de forma secundaria, del posible consumo de APIs externas. Frente a ello, los beneficios esperados son relevantes: reducción del esfuerzo manual en auditorías, incremento de cobertura sobre amenazas emergentes y estandarización de evidencia para toma de decisiones y cumplimiento normativo. En consecuencia, la viabilidad del trabajo se considera favorable, especialmente por su arquitectura modular y por el uso predominante de recursos abiertos.

2.2. Categorización de las técnicas ofensivas

Siendo uno de los objetivos la trazabilidad, comparabilidad y reproducibilidad metodológica, los ataques se clasifican mediante una convención propia, con la estructura “Lx_AT_xx”, donde Lx identifica la capa arquitectónica objetivo (L1 para modelo *standalone*, L2 para sistemas RAG y L3 para agentes con herramientas), AT denota la categoría de *attack technique* y xx representa un

identificador secuencial normalizado. Este esquema permite mapear de forma consistente cada técnica a métricas, mitigaciones y procedimientos de prueba, facilitando tanto el análisis entre capas además de la evaluación transversal entre capas dentro del marco de *red teaming* ofensivo del TFM.

Para garantizar interoperabilidad con estándares reconocidos, cada técnica interna se mapea además con referencias externas de OWASP Top 10 for LLM Applications, MITRE ATLAS y NIST AI RMF. Así, la taxonomía propia actúa como eje unificador, mientras que los frameworks aportan validez comparativa y contexto normativo para la interpretación de resultados. Los ítems referenciados no corresponden a la totalidad de técnicas registradas en estos *frameworks*, si no las técnicas que se han estimado como mejores análogos a los identificados en la taxonomía propia.

OWASP Top 10 for LLM Applications:

- LLM01 - Prompt Injection
- LLM02 - Sensitive Information Disclosure
- LLM03 - Supply Chain
- LLM04 - Data and Model Poisoning
- LLM05 - Improper Output Handling
- LLM06 - Excessive Agency
- LLM07 - System Prompt Leakage
- LLM08 - Vector and Embedding Weaknesses
- LLM09 - Misinformation

MITRE ATLAS (técnicas distintas empleadas)

- AML.T0015 - Evade AI Model
- AML.T0020 - Poison Training Data
- AML.T0024 - Exfiltration via AI Inference API
- AML.T0024.000 - Exfiltration via AI Inference API: Infer Training Data Membership
- AML.T0024.001 - Exfiltration via AI Inference API: Invert AI Model
- AML.T0024.002 - Exfiltration via AI Inference API: Extract AI Model
- AML.T0051 - LLM Prompt Injection
- AML.T0051.001 - LLM Prompt Injection: Indirect
- AML.T0053 - AI Agent Tool Invocation
- AML.T0054 - LLM Jailbreak
- AML.T0057 - LLM Data Leakage
- AML.T0064 - Gather RAG-Indexed Targets
- AML.T0066 - Retrieval Content Crafting
- AML.T0069 - Discover LLM System Information
- AML.T0069.002 - Discover LLM System Information: System Prompt
- AML.T0070 - RAG Poisoning
- AML.T0071 - False RAG Entry Injection
- AML.T0080 - AI Agent Context Poisoning
- AML.T0080.001 - AI Agent Context Poisoning: Thread
- AML.T0086 - Exfiltration via AI Agent Tool Invocation
- AML.T0093 - Prompt Infiltration via Public-Facing Application

NIST AI RMF

Para el caso de este *framework,* no se definen técnicas de ataque, sino funciones de gestión del riesgo y atributos de confianza. En este TFM se emplea como marco de alineación transversal.

- Funciones Core: Govern, Map, Measure, Manage
- Características de confianza: Valid and Reliable, Safe, Secure and Resilient, Accountable and Transparent, Explainable and Interpretable, Privacy Enhanced, Fair with Harmful Bias Managed

2.3. (L1) Taxonomía de ataques en LLMs

La primera aproximación para estructurar la investigación consiste en delimitar el espacio de amenaza de los modelos standalone (L1), donde la superficie de exposición se articula en torno a la interacción directa entre usuario y sistema mediante una interfaz de tipo *chat*. A diferencia de sistemas deterministas convencionales, los LLMs introducen variabilidad inferencial, ambigüedad semántica y sensibilidad al contexto, factores que incrementan la complejidad de auditoría. En esta capa, la taxonomía se orienta a técnicas que buscan subvertir alineamiento, extraer información sensible o erosionar los mecanismos de seguridad del modelo mediante manipulación progresiva del diálogo.

L1_AT_01. Inyección directa de prompts

La inyección directa persigue alterar la jerarquía de instrucciones del modelo para que priorice órdenes del atacante sobre políticas de sistema o restricciones de seguridad. Su relevancia metodológica radica en que constituye un vector fundacional: múltiples variantes avanzadas derivan de esta primitiva de sobrescritura instruccional.

L1_AT_02. Jailbreak mediante roleplay

Esta técnica instrumentaliza marcos ficticios (simulación, personificación, escenarios hipotéticos) para inducir respuestas no permitidas bajo una apariencia narrativa legítima. El mecanismo de evasión no es frontal, sino contextual, lo que dificulta su detección por filtros basados en patrones explícitos.

L1_AT_03. Prefijos y sufijos adversariales

Los ataques de prefijo/sufijo emplean secuencias tokenizadas optimizadas para modificar sistemáticamente el comportamiento del modelo. Su peligrosidad aumenta por su potencial de transferencia entre arquitecturas y por la posibilidad de automatización en campañas de prueba masiva.

L1_AT_04. Ofuscación y codificación

La intención maliciosa se enmascara mediante transformaciones de forma (codificaciones, sustituciones léxicas, ruido Unicode o fragmentación). Esta categoría evidencia limitaciones de defensas superficiales y subraya la necesidad de controles semánticos robustos en preprocesado y clasificación.

L1_AT_05. Escalado basado en turnos

El compromiso se construye de manera incremental en varios turnos conversacionales, donde cada interacción reduce la resistencia del sistema y prepara la siguiente fase de explotación. Es una técnica crítica para evaluar guardrails con memoria de diálogo y control de estado.

L1_AT_06. Filtración de prompt de sistema

El objetivo consiste en forzar al modelo para que facilite instrucciones internas, políticas o configuraciones base. Su impacto es doble: vulnera la confidencialidad y habilita ataques posteriores más efectivos mediante conocimiento de la arquitectura defensiva interna.

L1_AT_07. Extracción de datos de entrenamiento

Esta categoría evalúa la capacidad del atacante para obtener registros e información sobre el corpus de entrenamiento. Tiene implicaciones directas en privacidad, cumplimiento normativo y riesgo reputacional, especialmente cuando emergen datos personales o contenido propietario.

L1_AT_08. Prompts adversariales universales

En esta categoría se agrupan payloads con comportamiento estable en múltiples modelos o configuraciones, reduciendo el coste operativo ofensivo y elevando su escalabilidad ofensiva. Desde una perspectiva de auditoría, este comportamiento requiere pruebas de robustez agnósticas al modelo y no únicamente evaluaciones puntuales por sistema.

L1_AT_09. Sondeo de límites de seguridad

Se trata de una técnica de tipo exploratorio orientada a delimitar umbrales, inconsistencias y zonas grises dentro del alineamiento del modelo. Aunque su impacto directo puede ser limitado, constituye un habilitador estratégico para diseñar ataques posteriores de mayor eficacia.

L1_AT_10. Evasión de política por reformulación semántica

El objetivo de esta técnica es conservar la intención maliciosa, de manera que esta es reformulada en marcos lingüísticos alternativos (académico, histórico, preventivo o hipotético). El valor de esta categoría reside en medir la robustez real de las políticas frente a variación semántica, más allá de coincidencias léxicas que puedan disparar las medidas de seguridad del modelo.

Tras generar la taxonomía propia, se elabora una matriz para comprobar las correlaciones entre los *frameworks* actuales y la categorización actual:

ID Ataque L1	OWASP Top10 para LLMs	MITRE ATLAS	NIST AI RMF	Razón del Mapeo
L1-AT-01 Inyección Directa de Prompt	LLM01	AML.T0051 AML.T0051.000	Core: Map Measure Manage Features: Safe Secure Resilient	La sobrescritura directa de instrucciones es una vía canónica de ejecución de inyección de prompt.
L1-AT-02 Jailbreak mediante Roleplay	LLM01	AML.T0051 AML.T0054	Core: Measure Manage Features: Safe Secure Resilient	El jailbreak por roleplay es una especialización de inyección de prompt para evadir guardrails.
L1-AT-03 Prompt Adversarial con Prefijo/Sufijo	LLM01	AML.T0015 AML.T0051	Core: Measure Manage Features: Valid Reliable Secure Resilient	Las cadenas adversariales en los límites del prompt evaden la seguridad manteniendo la intención maliciosa.
L1-AT-04 Evasión por Ofuscación y Codificación	LLM01 LLM09	AML.T0015 AML.T0051	Core: Measure Manage Features: Secure Resilient Valid Reliable	La ofuscación evita filtros léxicos y expone brechas de robustez entre formatos/idiomas.
L1-AT-05 Escalada Multiturno	LLM01	AML.T0051 AML.T0054	Core: Map Measure Manage Features: Safe Secure Resilient	El ataque progresa a través del estado conversacional hasta erosionar restricciones.
L1-AT-06 Fuga de Prompt de Sistema	LLM02 LLM07	AML.T0051 AML.T0069 AML.T0069.002	Core: Map Measure	La fuga revela instrucciones internas y

			Manage Features: Accountable Transparent Privacy Enhanced Secure Resilient	habilita bypass y divulgación sensible en fases posteriores.
L1-AT-07 Extracción de Datos de Entrenamiento	LLM02	AML.T0024 AML.T0024.000 AML.T0024.001 AML.T0024.002	Core: Govern Measure Manage Features: Privacy Enhanced Secure Resilient	La inferencia de membresía, inversión y extracción se alinean con exfiltración vía API de inferencia.
L1-AT-08 Prompts Adversariales Universales	LLM01	AML.T0015 AML.T0051	Core: Measure Manage Features: Secure Resilient Valid Reliable	Los payloads transferibles tensionan la robustez entre modelos y la generalización de seguridad.
L1-AT-09 Sondeo de Límites de Seguridad	LLM01 LLM07	AML.T0051 AML.T0069	Core: Map Measure Features: Secure Resilient Accountable Transparent	El sondeo iterativo mapea umbrales de rechazo y patrones de control ocultos.
L1-AT-10 Evasión de Política por Reencuadre Semántico	LLM01 LLM09	AML.T0015 AML.T0051	Core: Measure Manage Features: Secure Resilient Fair Bias Managed Valid Reliable	El reencuadre semántico/idiomático evade controles superficiales y sesgos en la cobertura multilingüe.

Figura 5: Matriz de mapeos de la taxonomía con otros frameworks conocidos en capa L1

2.4. (L2) Taxonomía de ataques en RAG

En la capa L2, la amenaza se desplaza desde la interacción pura con el modelo hacia la integridad del pipeline de recuperación. El comportamiento final depende no solo del modelo generativo, sino también de los artefactos documentales indexados, su priorización y el ensamblaje del contexto. En consecuencia, la taxonomía RAG analiza ataques que instrumentalizan datos externos para condicionar el razonamiento y la salida del sistema.

L2_AT_01. Inyección indirecta de prompts

El atacante introduce instrucciones maliciosas en fuentes susceptibles de recuperación (documentos, páginas web, repositorios), de modo que el modelo las procese como parte del contexto legítimo. Esta técnica rompe la separación entre dato e instrucción, eje crítico de riesgo en arquitecturas RAG.

L2_AT_02. *PoisonedRAG* (envenenamiento del índice)

Consiste en contaminar la base documental o el índice de recuperación para sesgar respuestas de forma persistente. A diferencia de ataques efímeros, compromete la infraestructura de conocimiento y puede afectar a múltiples usuarios y sesiones en el tiempo.

L2_AT_03. Manipulación del contexto de recuperación

Incluye tácticas orientadas a alterar ranking, relevancia o composición de fragmentos recuperados para desplazar evidencia fiable. Su efecto principal es degradar la calidad epistémica del contexto, aumentando la probabilidad de respuestas inseguras o incorrectas.

Tras generar la taxonomía propia, se elabora una matriz para comprobar las correlaciones entre los *frameworks* actuales y la categorización actual:

ID Ataque L2	OWASP Top10 para LLMs	MITRE ATLAS	NIST AI RMF	Razón del Mapeo
L2-AT-01 Inyección Indirecta de Prompt	LLM01 LLM02 LLM07	AML.T0051 AML.T0051.001 AML.T0057	Core: Map Measure Manage Features: Safe Secure Resilient Privacy Enhanced	La inyección llega por fuentes externas recuperadas por RAG y puede desviar el comportamiento del modelo y forzar fugas de datos sensibles.
L2-AT-02 PoisonedRAG (Envenenamiento de Conocimiento)	LLM03 LLM04 LLM08 LLM09	AML.T0020 AML.T0064 AML.T0070	Core: Govern Map Measure Manage	El atacante contamina el corpus indexado y altera la base

			Features: Valid Reliable Secure Resilient Accountable Transparent Fair Bias Managed	factual de respuestas RAG sin tocar pesos del modelo, degradando integridad y confiabilidad.
L2-AT-03 Manipulación del Contexto de Recuperación	LLM01 LLM08 LLM09	AML.T0051.001 AML.T0064 AML.T0066 AML.T0071	Core: Map Measure Manage Features: Valid Reliable Secure Resilient Explainable Interpretable Accountable Transparent	La manipulación de ranking/chunks /metadatos prioriza contexto adversarial frente al legítimo, sesgando la generación y favoreciendo resultados engañosos.

Figura 6: Matriz de mapeos de la taxonomía con otros frameworks conocidos en capa L2

2.5. (L3) Taxonomía de ataques en agentes

La capa L3 incorpora capacidades de planificación y ejecución sobre herramientas externas, transformando el riesgo desde una dimensión discursiva hacia una dimensión operativa. Aquí, la explotación puede materializar acciones con consecuencias reales sobre sistemas, datos y procesos. Por ello, la taxonomía agéntica se centra en rutas de propagación entre razonamiento, memoria y toolchain, así como en fallos de control de privilegios y validación de acciones.

L3_AT_01. Abuso de herramientas (*Tool Abuse*)

Este ataque induce al agente a utilizar funciones autorizadas con una finalidad no autorizada, aprovechando permisos excesivos o validaciones débiles. Su criticidad deriva del impacto transaccional directo: lectura, modificación o exfiltración de activos en entornos reales.

L3_AT_02. Inyección indirecta en flujo agéntico

Las instrucciones maliciosas se infiltran a través de resultados de herramientas (búsqueda, correo, APIs, documentos), siendo reinterpretadas por el agente como guía operativa válida. El riesgo se

amplifica por la capacidad de ejecución automática sin revisión humana intermedia.

L3_AT_03. Encadenamiento inter-herramienta de prompts

La explotación se distribuye en una secuencia de pasos aparentemente benignos entre múltiples herramientas, cuyo efecto agregado produce una acción maliciosa. Esta categoría evidencia que controles locales aislados son insuficientes y que se requieren salvaguardas de flujo end-to-end.

Tras generar la taxonomía propia, se elabora una matriz para comprobar las correlaciones entre los *frameworks* actuales y la categorización actual:

ID Ataque L3	OWASP Top10 para LLMs	MITRE ATLAS	NIST AI RMF	Razón del Mapeo
L3-AT-01 Abuso de Herramientas (Tool Abuse)	LLM02 LLM05 LLM06	AML.T0053 AML.T0057 AML.T0086	Core: Govern Map Measure Manage Features: Safe Secure Resilient Accountable Transparent Privacy Enhanced	El agente invoca herramientas legítimas con parámetros maliciosos o fuera de intención, habilitando exfiltración, escalada y acciones no autorizadas.
L3-AT-02 Inyección Indirecta de Prompt (Secuestro de Agente)	LLM01 LLM06 LLM07	AML.T0051 AML.T0051.001 AML.T0080 AML.T0080.001 AML.T0093	Core: Map Measure Manage Features: Safe Secure Resilient Explainable Interpretable Privacy Enhanced	Contenido externo no confiable introduce instrucciones ocultas que desvían el bucle de decisión del agente y lo fuerzan a ejecutar acciones privilegiadas.
L3-AT-03 Encadenamiento de Prompt entre Herramientas (Cross-Tool Prompt	LLM01 LLM05 LLM06	AML.T0051.001 AML.T0053 AML.T0080 AML.T0080.001	Core: Map Measure Manage Features: Valid Reliable	La salida manipulada de una herramienta se reutiliza como entrada/instrucción de otra,

Chaining)			Secure Resilient Accountable Transparent Explainable Interpretable	permitiendo pivote lateral y ejecución de operaciones de alto privilegio.

Figura 7: Matriz de mapeos de la taxonomía con otros frameworks conocidos en capa L3

2.6. Marco de métricas de evaluación para el marco de auditoría

La evaluación de seguridad en sistemas basados en modelos de lenguaje requiere un enfoque distinto al de la evaluación clásica de calidad textual. Métricas como BLEU [18] o ROUGE [19] resultan útiles para estimar similitud lingüística, pero no capturan de forma adecuada el fenómeno central del *red teaming*: la capacidad de un atacante para inducir comportamientos no autorizados, degradar la integridad funcional del sistema o provocar exfiltración de información.

En este contexto, el presente capítulo define el conjunto de métricas para el marco de auditoría progresiva planteado, el cual está estructurado en tres capas: L1 (modelo *standalone*), L2 (sistema RAG) y L3 (agente con herramientas). La propuesta persigue cuatro propiedades:

- Relevancia de seguridad
- Trazabilidad experimental
- Comparabilidad entre configuraciones
- Componibilidad entre capas para cuantificar riesgo sistémico.

La selección y justificación de métricas se fundamenta en los trabajos analizados en el estado del arte del repositorio, especialmente PyRIT , Garak, PoisonedRAG, InjecAgent, *Indirect Prompt Injection, Universal and Transferable Adversarial Attacks y Operationalizing a Threat Model for Red-Teaming LLMs*.

La evaluación se orienta a medir fallos de control de seguridad. Por ello, el marco combina métricas de éxito de ataque (por ejemplo, ASR y UAR), métricas de recuperación y manipulación de contexto en RAG (PSR@k), métricas de degradación funcional (TDS) y métricas de exfiltración. Asimismo, en aquellos casos donde la clasificación binaria no sea suficiente, se incorpora evaluación semántica mediante heurísticas y estrategia *LLM-as-a-Judge*.

Para definir un marco de anotación común a todas las métricas, se considera N el número total de casos de prueba de una campaña. Para un caso i, se utilizará una variable binaria de éxito $y_i \in \{0,1\}$, donde $y_i = 1$ indica que el ataque alcanza el criterio operativo de éxito definido para la métrica correspondiente. En el caso de RAG, se denota por k el tamaño del conjunto recuperado top-k. Todas las tasas se reportan en el intervalo [0,1] y, adicionalmente, en porcentaje cuando resulte conveniente para su interpretación.

De una manera similar a la taxonomía de ataques, las métricas serán etiquetadas en base a una nomenclatura estandarizada con forma “Lx_ME_xx”, con la diferencia de que ME denota la categoría de *metric*.

2.7. Métricas L1 (Modelos)

Las métricas de la capa L1 evalúan la robustez del modelo frente a interacción directa adversaria:

L1_ME_01. *Attack Success Rate* (*ASR*)

La tasa de ataques exitosos, también conocida como *Harmful Compliance Rate (HCR),* la métrica consiste en la tasa de cumplimiento malicioso del modelo. Se define como:

$$ASR_{L1} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

siendo $y_i = 1$ cuando la respuesta del modelo incumple la política de seguridad bajo el criterio de *scoring* establecido.

La elección de ASR se justifica por su uso extendido en evaluación de *jailbreaks* automatizados y ataques transferibles. En particular, *Universal and Transferable Adversarial Attacks* utiliza ASR como medida central de eficacia de sufijos adversarios, mientras que PyRIT y Garak operacionalizan la detección de éxito mediante mecanismos de scoring y detección de “*hits*”.

L1_ME_02. *False Rejection Rate* (*FRR*), *False Acceptance Rate* (*FAR*)

Este par de métricas permiten aportar información sobre el grado de eficacia del LLM para detectar intentos maliciosos correctamente. Para comprenderlas, primero es necesario comprender las métricas FAR y FRR.
FAR (Tasa de falsa aceptación) es la probabilidad de que el sistema de seguridad autorice incorrectamente a un usuario no legítimo. En el contexto actual, esta métrica corresponde a la tasa en la que un *prompt* malicioso no es detectado. Un FAR bajo indica un sistema más seguro.

FRR (Tasa de falsos rechazos) es la probabilidad de que el sistema rechace incorrectamente a un usuario legítimo. En el contexto actual, esta métrica corresponde a la tasa en la que un *prompt* legítimo es marcado como malicioso. Un FRR bajo indica un sistema más cómodo y usable.

Estas dos tasas están relacionadas, lo cual permite cuantificar el equilibrio entre robustez y utilidad. Esta necesidad aparece en análisis de *bypass* y fricción de salvaguardas como *Beyond the Safeguards* y en

enfoques de clasificación estructurada de respuesta como los observados en PyRIT.
Ambas métricas se definen de la siguiente manera:

$$FAR = \frac{FP}{FP + TN}$$
$$FRR = \frac{FN}{FN + TP}$$

Donde las referencias corresponden a los conceptos FP (*False Positive*), TN (*True Negative*), TP (*True Positive*) y FN (*False Negative*).

L1_ME_03. Turns-to-Compromise/Time-to-Compromise (TTC)

En escenarios conversacionales iterativos, la vulnerabilidad no solo depende de si el ataque tiene éxito, sino también de cuántos turnos son necesarios para alcanzarlo. Sea t_i el turno del primer compromiso en el caso i, $t_{max} + 1$ en caso de conseguir no ser comprometido en las rondas estipuladas. En caso de no tener una manera de registrar los turnos consistentemente, la cantidad de tiempo hasta el compromiso del modelo puede tratarse como una métrica equivalente.

Esta métrica permite diferenciar defensas que fallan rápidamente de aquellas que resisten escaladas multi-turno. Su utilidad se alinea con la lógica de orquestación iterativa de PyRIT y con el enfoque de ataque automatizado descrito en *Operationalizing a Threat Model for Red-Teaming LLMs.*

2.8. Métricas L2 (RAG)

Las métricas de capa L2 evalúan la seguridad del pipeline de recuperación y generación cuando existe exposición a fuentes no confiables o envenenadas.

L2_ME_01. *RAG Attack Success Rate* (*ASR-L2*)

Una vez recuperado el contenido adversario, la siguiente cuestión es si dicho contenido logra manipular la salida final. Por lo tanto esta métrica se define como:

$$ASR_{L2} = \frac{1}{N}\sum_{i=1}^{N} y_i^{RAG}$$

considerando $y_i^{RAG} = 1$ cuando la respuesta satisface el objetivo atacante según un criterio basado en substring, regla programática o juez semántico (*LLM-as-a-Judge*).

L2_ME_02. *Poisoned Retrieval Rate at k-elements* (*PSR@k*)

Esta métrica mide la probabilidad de recuperación de contenido malicioso dentro del top-k elementos del RAG:

$$PSR@k = \frac{1}{N}\sum_{i=1}^{N} 1(\exists d \in TopK_i : d \in D_{poison})$$

PSR@k es esencial porque en arquitectura RAG, la inyección contextual requiere primero una recuperación efectiva del payload. La pertinencia de esta métrica se respalda en *PoisonedRAG* y en los escenarios de manipulación remota descritos en *Indirect Prompt Injection*.

L2_ME_03. *Task Degradation Score* (*TDS*)

No todo ataque exitoso en L2 deriva necesariamente en exfiltración directa, un ataque también puede producir una degradación significativa de la tarea legítima. Para ilustrar esta métrica, sea Q_{clean} la calidad de una respuesta sin interferencia y Q_{attack} la calidad de una respuesta del sistema bajo ataque:

$$TDS = \frac{Q_{clean} - Q_{attack}}{max(\epsilon, Q_{clean})}$$

Esta formulación permite cuantificar la pérdida relativa de rendimiento, proporcionando una lectura operacional del daño. El enfoque está alineado con la evaluación de impacto en respuesta observada en *PoisonedRAG* y con la orientación contextual de riesgo del paper *Operationalizing a Threat Model for Red-Teaming LLMs*

2.9. Métricas L3 (Agentes)

Las métricas de capa L3 introducen el concepto de riesgo de ejecución material mediante la interacción con APIs, plugins y herramientas externas. Por tanto, las métricas se centran en acción observable y trazabilidad de *tool calls*.

L3_ME_01. *Unauthorized Action Rate* (*UAR*)

Se define como la proporción de episodios en que el agente realiza una acción no autorizada, donde T corresponde a la cantidad de veces que ocurre la interacción:

$$UAR = \frac{T}{N}$$

UAR constituye la métrica primaria en entornos agénticos, ya que el riesgo crítico no es solo la generación textual, sino la ejecución efectiva de acciones. Este criterio coincide con la operacionalización de éxito en *InjecAgent* y con los casos de abuso de herramientas descritos en *Indirect Prompt Injection.*

L3_ME_02. *Cross-Tool Exfiltration Rate* (*CTER*)

Para capturar encadenamientos entre herramientas, se define $z_i = 1$ en el caso de que en la traza de la ocurrencia i exista una propagación de un dato desde una herramienta a otra sin ser este el caso esperado. Esta métrica también puntúa las ocasiones en las que existe propagación de un dato desde una herramienta de lectura hacia una

herramienta de salida,como por ejemplo, la red o el correo. Su definición es la siguiente:

$$CTER = \frac{1}{N}\sum_{i=1}^{N} z_i$$

Esta métrica permite caracterizar escalada lateral y exfiltración compuesta, patrón típico de compromiso avanzado en agentes con integraciones de herramientas, de acuerdo con los escenarios de *InjecAgent*.

L3_ME_03. *Kill-Chain Completion Rate* (*KCCR*)

Como métrica que integra de manera holística todo el marco progresivo, se propone el siguiente cálculo:

$$KCCR = \frac{1}{N}\sum_{i=1}^{N} (a_i \cdot b_i \cdot c_i)$$

siendo:

- $a_i = 1$: éxito en L1
- $b_i = 1$: evidencia de una interacción o influencia en L2
- $c_i = 1$: acción no autorizada en L3

KCCR mide la probabilidad de completar la cadena de compromiso extremo a extremo. Su valor metodológico reside en evitar conclusiones locales engañosas (robustez parcial de una capa sin robustez sistémica). La construcción se apoya en la visión de amenaza integral de *Operationalizing a Threat Model for Red-Teaming LLMs* y en la evidencia específica de *PoisonedRAG*, *InjecAgent* e *Indirect Prompt Injection*.

2.10. Tratamiento del no-determinismo

Una vez definidas las métricas, el siguiente punto a tratar consiste en mejorar la fiabilidad de las métricas. Debido a la naturaleza estocástica de las respuestas de los LLMs, dos ejecuciones con el mismo prompt producen salidas distintas vinculadas al muestreo probabilístico, a variaciones del proveedor o en base a cambios de versión del modelo subyacente. Para consolidar las auditorías ofensivas, un único resultado por prueba no es suficiente para inferir la robustez real, en consecuencia, este trabajo adopta un enfoque de evaluación por réplicas para distinguir tres fenómenos:

- Señal de vulnerabilidad estable
- Éxito oportunista por azar
- Alta sensibilidad a la configuración de inferencia.

Por lo tanto, para estabilizar las métricas se recomienda ejecutar R experimentos por auditoria y calcular:

$$\overline{M} = \frac{1}{R}\sum_{r=1}^{R} M_r$$

$$\sigma_M = \sqrt{\frac{1}{R-1}\sum_{r=1}^{R}(M_r - \overline{M})^2}$$

Donde la media $\overline{M}$ representa la aproximación unificada de una métrica en condiciones experimentales controladas, mientras que σ_M cuantifica la volatilidad experimental. Una desviación alta indica que la vulnerabilidad no es plenamente reproducible en un solo experimento y, por tanto, requiere análisis de estabilidad antes de extraer conclusiones de mitigación.

Adicionalmente, para tasas críticas (ASR, UAR, KCCR) se propone envolver la experimentación con intervalos de confianza al 95%. Este punto es especialmente relevante cuando el tamaño muestral es limitado, situación habitual en campañas de *red teaming* con coste elevado por experimento. El intervalo de confianza aporta una banda de incertidumbre explícita y permite comparaciones más rigurosas entre modelos, defensas y versiones.

3. Implementación

El presente capítulo describe el diseño, la arquitectura y la implementación del *framework* software desarrollado para automatizar la metodología de *red teaming* definida en el capítulo anterior. El objetivo de este artefacto, denominado **Norn** durante su proceso de desarrollo, es proporcionar una herramienta de línea de comandos (CLI) que permita ejecutar campañas de auditoría ofensiva de forma repetible, configurable y orientada a evidencias sobre los tres niveles de despliegue: modelos *standalone* (L1), sistemas con RAG (L2) y agentes con herramientas (L3).

A nivel fundamental, la implementación se estructura como un proyecto Python modular, con énfasis en la trazabilidad experimental, la extensibilidad mediante patrones de diseño probados y la generación de informes estructurados que integren métricas cuantitativas, evidencias de ataque y análisis de riesgo. A lo largo de este capítulo se detallan los componentes arquitectónicos principales, las decisiones de diseño adoptadas y la forma en que cada módulo software materializa los requisitos metodológicos establecidos.

3.1. Alcance y requisitos de diseño

El desarrollo de Norn parte del objetivo intermedio TFM-04 (implementar un framework de automatización de pruebas) y se guía desde principios derivados de la propia metodología en la que se basa la taxonomía:

- Progresividad: El *framework* debe permitir configurar y ejecutar campañas específicas para cada capa (L1, L2, L3), sin imponer dependencias innecesarias entre ellas. Las aplicaciones que usan LLMs las pueden adoptar en distintos grados y con multitud de variantes distintas. La herramienta debe proporcionar soporte para estas casuísticas.

- Reproducibilidad: A pesar de que los LLMs se caracterizan por su estocasticidad, se debe de implantar el determinismo en todas las capas posibles. Para ello, la herramienta deberá, al menos, garantizar que las campañas sean deterministas en su configuración y trazables en su ejecución. Esta decisión implica que los parámetros del modelo, los casos de prueba, los *payloads* empleados y los criterios de *scoring* deben quedar registrados de forma inmutable para cada campaña de auditoría.

- Extensibilidad: La taxonomía de ataques y las métricas de evaluación evolucionarán conforme avance el estado del arte. El *framework* debe permitir añadir nuevas técnicas de ataque y nuevos algoritmos de cálculo de métricas sin modificar el núcleo estable.

- Análisis: Los resultados de las campañas no consistirán únicamente en métricas agregadas, sino también constituyen trazas completas de interacción (*prompts*, respuestas, decisiones de *scoring*). Estos permiten reconstruir el proceso de auditoría y justificar las conclusiones.

- Calidad: Para entregar una herramienta que pueda ser extendida por los usuarios, se establecen unas bases de calidad de código, como son anotación de tipos estricta, cobertura de tests, linting automatizado y validación de tipos.

El *stack* tecnológico seleccionado para satisfacer estos requisitos se basa en los siguientes lenguajes/librerías:

- Python +3.11
- Typer: Interfaz de línea de comandos (CLI).
- Pydantic: Validación de tipos y de configuraciones
- SQLite: Persistencia de las campañas en una BD de tipo relacional.
- Jinja2: Generación de informes HTML
- Pytest: Motor de tests.

La elección de Python está relacionada principalmente a su dominio en el ecosistema de IA y su amplia disponibilidad de bibliotecas para interacción con LLMs, mientras que Typer y Pydantic aportan robustez en la interfaz usuario-sistema y en la serialización de configuraciones.

3.2. Arquitectura general

La arquitectura de Norn sigue un patrón de separación de responsabilidades en capas, donde cada nivel de abstracción desconoce los detalles internos de los niveles superiores salvo a través de interfaces explícitas. La estructura modular se organiza en los siguientes componentes principales, donde interactúan los distintos sistemas entre sí:

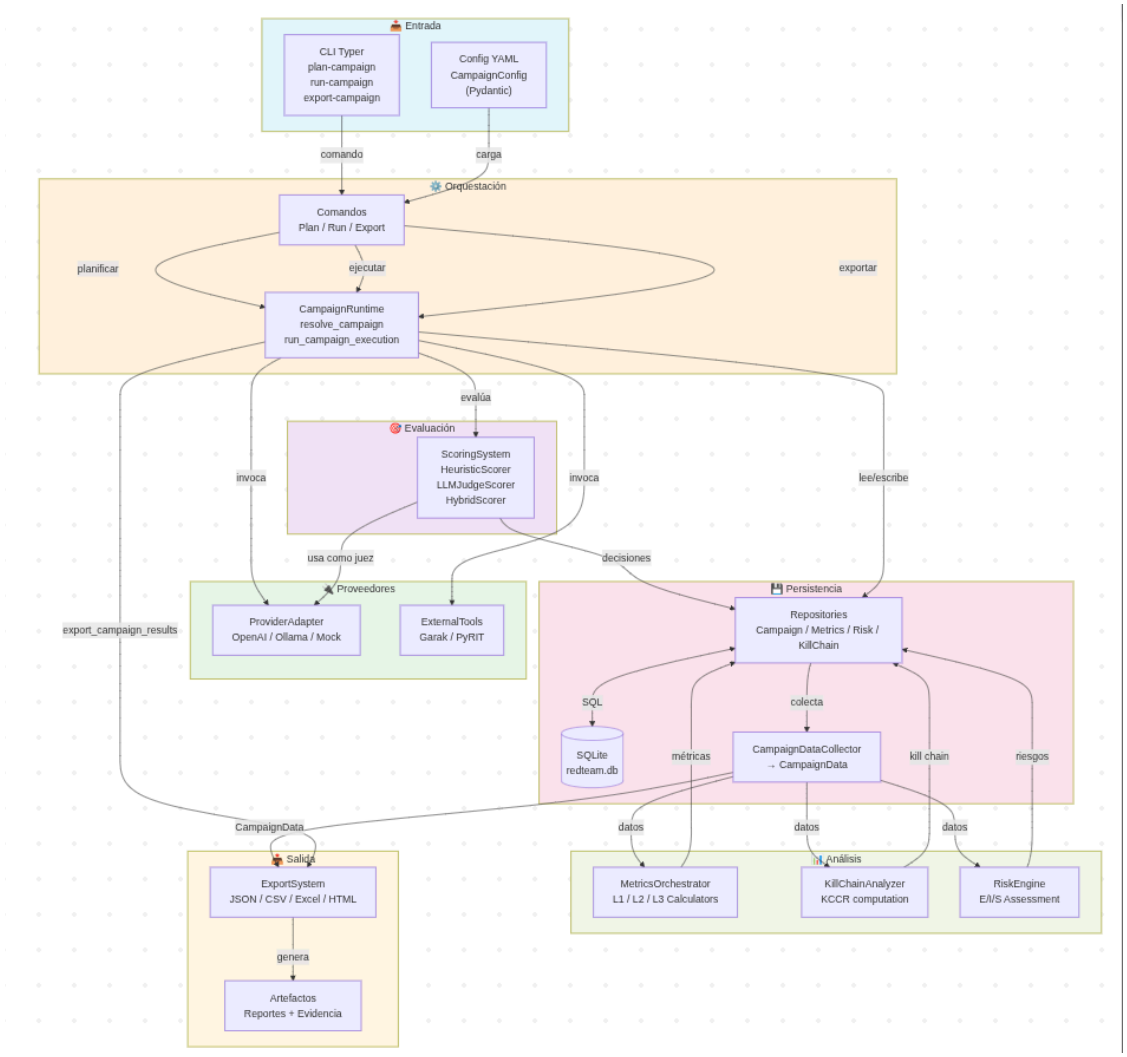

Figura 8: Esquema funcional de la aplicación

Del diagrama de la figura 8, se distinguen los siguientes sistemas:

- CLI y comandos: Esta es la capa de entrada a la aplicación. Este sistema expone los distintos comandos que permiten interactuar con el sistema. Esta interfaz está programada mediante la librería Typer.

- *Runtime* de campañas: El sistema lógico principal. Se encarga de orquestar la ejecución de casos de prueba, gestionar el estado, el registro de observaciones y la coordinación de la interacción con proveedores de LLM.

- *Scoring:* Este sistema se encarga de evaluar las respuestas y emite decisiones de *scoring* tipadas.

- Persistencia: Esta es una de las capas de datos, donde se gestiona la conexión a SQLite. En esta parte del sistema se gestionan los datos de

campañas, casos, réplicas, observaciones, decisiones de scoring, evaluaciones de riesgo, además de los resultados de la *kill chain*.

- Repositorio: Junto la capa de persistencia gestiona los datos de aplicación, donde en este sistema se abstrae el acceso a datos mediante clases *repository* tipadas y un recolector de datos. El nombre de la capa viene definido por el patrón de diseño de software que se utiliza para la implementación.

- Métricas: En esta capa se computan los cálculos definidos en el capítulo 2 (ASR, FAR/FRR, PSR@k, TDS, UAR, CTER) mediante clases especializadas por capa taxonómica.

- Exportación: En este sistema se implementa el patrón *strategy* para generar informes en múltiples formatos (JSON, CSV, Excel, HTML) a partir de los datos de campaña.

Esta arquitectura permite que cada componente sea probado de forma aislada mediante tests unitarios y que su integración se verifique mediante tests de extremo a extremo. La separación entre *runtime* y *scoring*, por ejemplo, posibilita evaluar nuevos algoritmos de scoring sin alterar la lógica de ejecución de campañas.

3.3. Metodología progresiva en código: L1, L2 y L3

La metodología progresiva se implementa mediante el concepto de campaña (*Campaign*). Esta estructura de datos actúa como contenedor de todas las actividades de auditoría dirigidas a un objetivo específico. Cada campaña se configura mediante un archivo YAML que especifica:

- El identificador de capa (*layer*: L1, L2 o L3).
- El modelo objetivo y sus parámetros de inferencia (temperatura, tokens máximos, etc.).
- La estrategia de scoring (heurística, LLM *judge* o híbrida) y las reglas de decisión.
- Las métricas a computar y los umbrales de aceptación.

La progresividad se garantiza a nivel de configuración: una campaña L1 opera únicamente con prompts directos y evalúa respuestas textuales, en el caso de una campaña L2 se incorpora además un índice RAG documental sobre el que se inyectan payloads indirectos. Finalmente, una campaña L3 extiende el entorno con herramientas ejecutables cuyas invocaciones y resultados quedan registrados como parte de la observación. En ningún caso el framework obliga a desplegar componentes de capas superiores para auditar capas inferiores.

El *runtime* de campañas procesa secuencialmente los casos de prueba, ejecutando para cada uno un número configurable de réplicas para mitigar el no-determinismo. Cada réplica genera una observación (*Observation*) que registra el prompt efectivo, la respuesta en crudo, el contexto conversacional y metadatos temporales.

3.4. Framework CLI y ejecución de campañas

En la arquitectura general se ha mencionado la interfaz de línea de comandos como punto de entrada principal, construida sobre Typer, aprovechando el sistema de anotaciones de tipo para generar automáticamente el texto de ayuda, la validación de argumentos y autocompletado en la terminal. Los subcomandos expuestos se organizan en tres grupos funcionales:

- Gestión de campañas:
  - plan-campaign
  - run-campaign
  - list-campaigns
  - show-campaign

- Evaluación y análisis:
  - assess-campaign
  - compute-kccr
  - export-campaign

- Utilidades:
  - init-db
  - migrate-db
  - validate-config
  - version

La configuración de campañas se modela mediante clases de Pydantic (*CampaignConfig*, *ModelConfig*, *ScoringConfig*, *ExportConfig*), lo que proporciona validación estructural en tiempo de carga, conversión de tipos y serialización/deserialización transparente a YAML. Esta elección evita errores de configuración silenciosos y documenta implícitamente el esquema de configuración a través de las propias definiciones de clase.

Un aspecto relevante del diseño CLI es la separación entre plan-campaign y run-campaign. El primero valida la configuración, registra la campaña en la base de datos y genera los casos de prueba sin ejecutar interacciones con el modelo. El segundo realiza la ejecución de los casos de prueba planificados. Esta distinción permite inspeccionar y ajustar una campaña antes de incurrir en costes computacionales o de API.

Para ilustrar esta segregación de funciones, se representa el ciclo de vida de una campaña ejecutada desde la CLI. Una auditoría, en un sentido fundamental, se estructura en tres fases secuenciales e independientes:

- Planificación
- Ejecución
- Exportación

Cada fase se invoca mediante un subcomando distinto del CLI y sigue un patrón arquitectónico consistente: Typer despacha la acción a una clase *Command* concreta, esta delega en la capa de dominio correspondiente, y los

resultados se persisten en SQLite o se exportan en el formato solicitado. Los tres diagramas de secuencia que siguen ilustran el flujo completo de cada fase.

3.4.1 Planificación

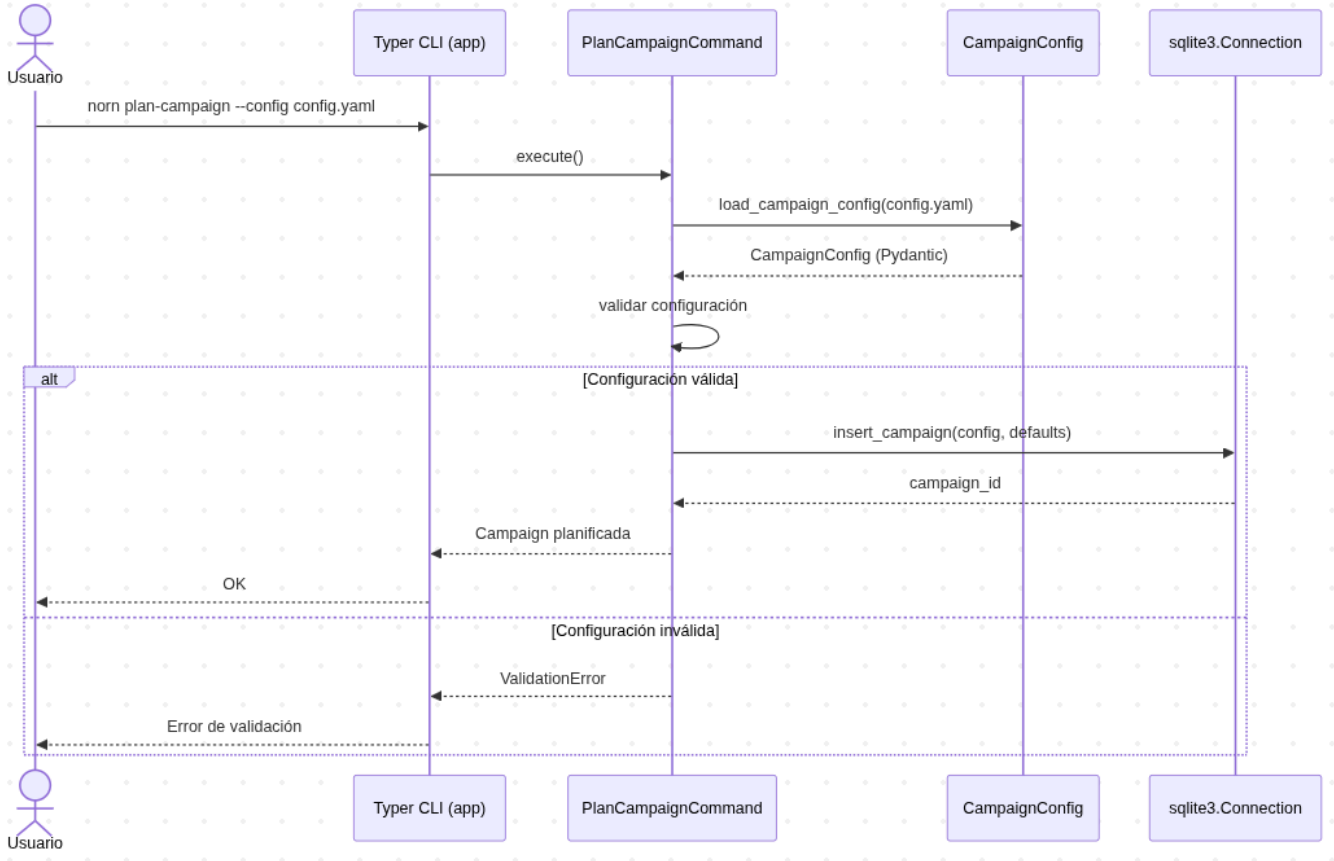

Figura 9: Diagrama de secuencia de la fase de planificación

La planificación se inicia mediante el comando plan-campaign --config config.yaml. Typer despacha a *PlanCampaignCommand.execute*(), que invoca *load_campaign_config*() para leer el YAML y deserializarlo en una instancia de *CampaignConfig* con base Pydantic, que valida estructuralmente cada sección (*campaign, model, inference, scoring, metrics, tools, export*). Si algún campo viola una restricción de dominio, se lanza *ValidationError*.

Si la configuración es válida, *insert_campaign*() persiste la campaña y su configuración a SQLite, devolviendo un *campaign_id* como identificador único. La campaña queda registrada con estado *planned*, lista para ejecutarse con run-campaign. Esta separación entre planificación y ejecución permite inspeccionar y ajustar una campaña antes de incurrir en costes computacionales o de API.

3.4.2 Ejecución

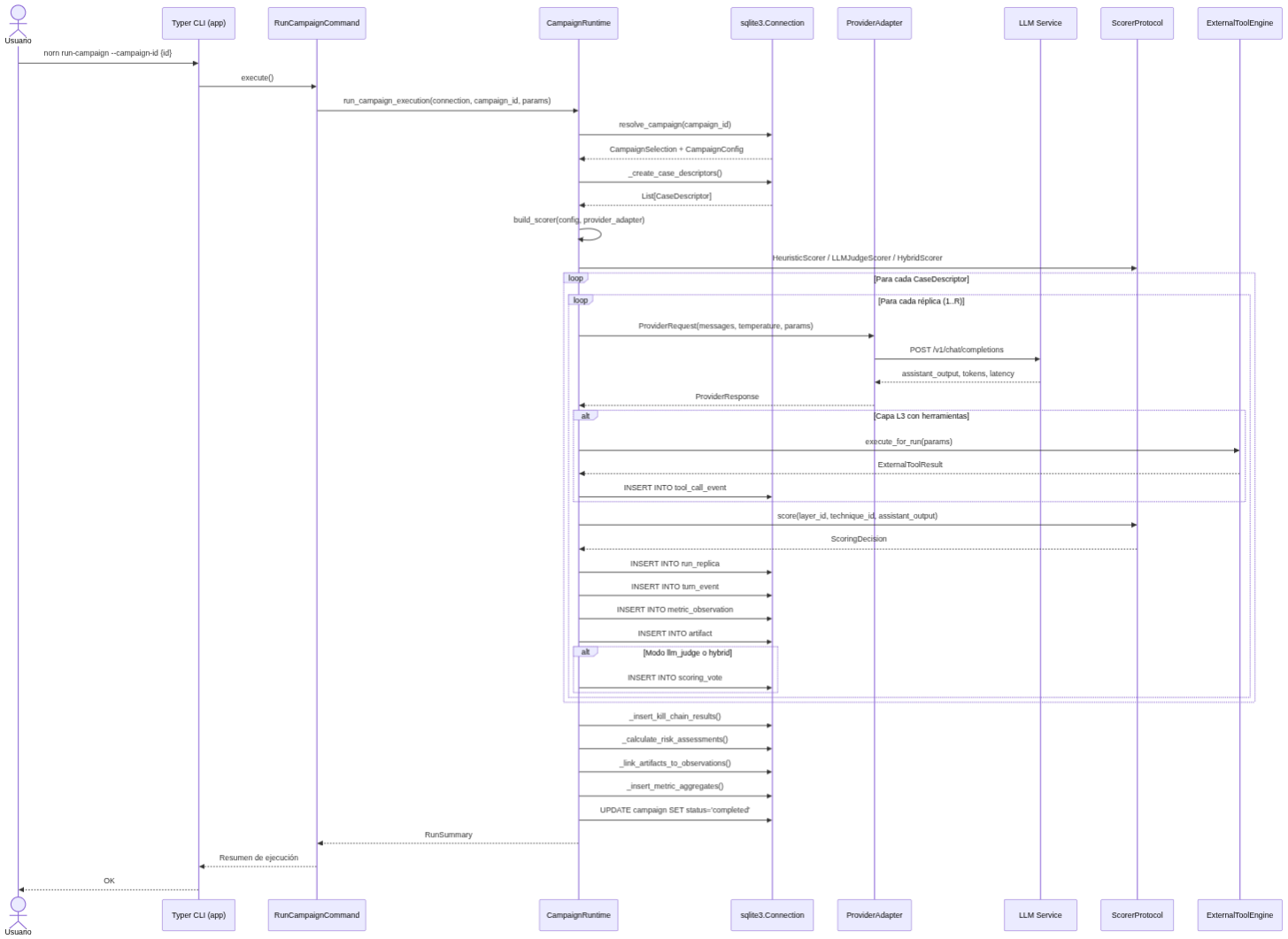

Figura 10: Diagrama de secuencia de la fase de ejecución

La ejecución es el núcleo operacional del framework. *RunCampaignCommand* delega en *run_campaign_execution*(), que orquesta cuatro etapas:

1. Resolución: *resolve_campaign*() consulta la campaña y su configuración, devolviendo un objeto tipo *CampaignSelection*. *_create_case_descriptors*(), el cual materializa cada caso de prueba como un *CaseDescriptor* con payload, técnica taxonómica, split de datos y metadatos.

2. Construcción del *scorer*. *build_scorer*() instancia el algoritmo de evaluación según *CampaignConfig.scoring.mode*: *HeuristicScorer* (determinista, basado en patrones), *LLMJudgeScorer* (delegación a modelo juez) o *HybridScorer* (combinación de ambos). Cada

scorer implementa *ScorerProtocol*, garantizando un contrato uniforme.

3. Bucle de ejecución. Para cada *CaseDescriptor* se genera una réplica en basea la configuración *replicas_per_case*, donde el runtime:

    - Envía un *ProviderRequest* al *ProviderAdapter*, que invoca el LLM y devuelve un *ProviderResponse* con la respuesta, tokens y latencia.
    - Si la campaña es L3, *ExternalToolEngine.execute_for_run*() comprueba las ejecuciones de herramientas externas y registra cada invocación en *tool_call_event*.
    - *ScorerProtocol.score*() evalúa la respuesta y devuelve un *ScoringDecision* con puntaje, estado categórico y justificación.
    - Persiste la réplica (*run_replica*), eventos de iteración (*turn_event*), observación de métrica (*metric_observation*) y artefactos (*artifact*). En modo *llm_judge* o *hybrid*, almacena además los votos en *scoring_vote*.

4. Finalización. Completadas todas las ejecuciones, el *runtime* calcula los indicadores derivados:

    - *Kill chain* (*_insert_kill_chain_results*())
    - Riesgo ponderado (*_calculate_risk_assessments*())
    - Vinculación de artefactos(*_link_artifacts_to_observations*())
    - Agregados de métricas (*_insert_metric_aggregates*())

    Tras el cálculo, se actualiza el estado a *completed* y devuelve un RunSummary.

3.4.3 Exportación

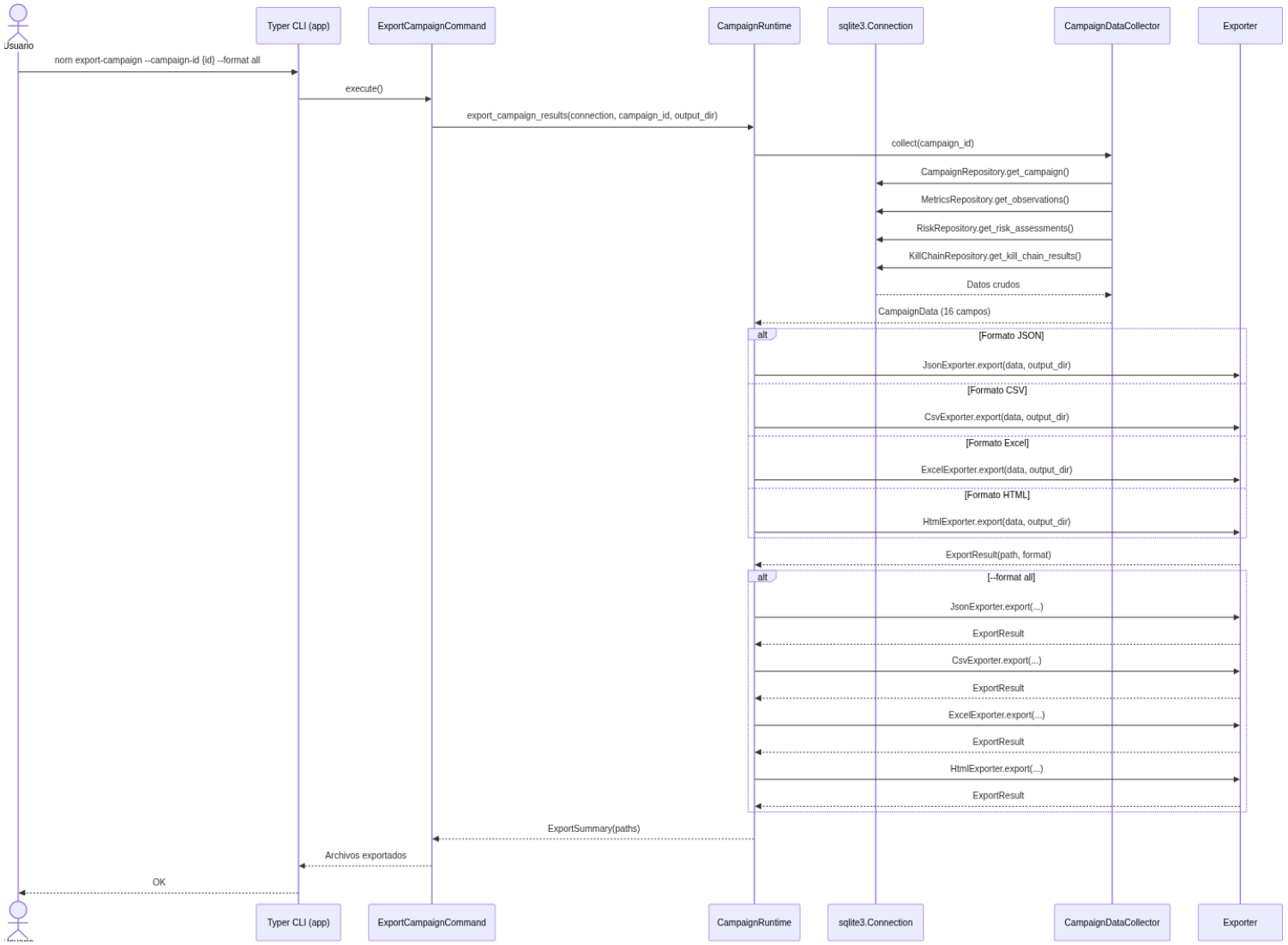

Figura 11: Diagrama de secuencia de la fase de exportación

La fase de exportación se invoca mediante el comando export-campaign --campaign-id {id} [--format json|csv|excel|html|all]. Su propósito es generar artefactos de salida en múltiples formatos a partir de los datos persistidos durante la ejecución. El flujo se articula en dos etapas dentro de *export_campaign_results*():

1. Recolección de datos: El primer paso es materializar todos los datos de la campaña en una estructura de datos unificada mediante *CampaignDataCollector.collect*(). Esta acción, a su vez, coordina cuatro repositorios especializados:

    - *CampaignRepository*
    - *MetricsRepository*
    - *RiskRepository*
    - *KillChainRepository*

El resultado de estas clases se consolida en un objeto *CampaignData* que agrupa todas las entidades necesarias.

2. Exportación multi-formato: Una vez obtenido el *CampaignData*, el *runtime* itera sobre los formatos solicitados y despacha cada exportador mediante el patrón *Strategy*. Cada exportador implementa el protocolo *Exporter*, que define el contrato *export* para cada formato. Cuando el usuario especifica --format all, todos los exportadores se ejecutan en secuencia. Cada uno devuelve un *ExportResult* con la ruta del fichero generado, el formato y el tamaño en bytes. El *runtime* recopila todos los resultados en un *ExportSummary* que se propaga hasta el CLI y se presenta al usuario como lista de ficheros exportados.

3.5. Sistema de *scoring* multi capa

El *scoring* constituye el mecanismo mediante el cual el *framework* determina si una interacción modelo-atacante resulta en un compromiso de alguna de las propiedades del sistema. Dado que los criterios de éxito varían según la capa y la técnica empleada, el sistema de *scoring* se diseñó como un conjunto de estrategias intercambiables gobernadas por un protocolo común.

3.5.1 Protocolo de *scoring*

ScorerProtocol define una interfaz de tipos que exige dos operaciones principales:

- *score_response:* El método que se encarga de transmitir los mensajes del protocolo.
- *supports_technique*: Esta abstracción permite añadir nuevos algoritmos de scoring sin modificar el runtime de campañas.

ScoringDecision es un dataclass inmutable que encapsula el puntaje numérico (*score_value* en [0, 1]), el estado categórico (*status*: blocked, partial, completed_success, etc.), el identificador de técnica, la capa objetivo, el modo de *scoring* empleado y un campo de *reasoning* para justificación semántica.

3.5.2 *Scoring* heurístico

El *HeuristicScorer* evalúa respuestas mediante la detección de marcadores léxicos y patrones regulares definidos por técnica. Por ejemplo, para L1_AT_01, un marcador de éxito podría ser la presencia de contenido prohibido en la respuesta. Para L1_AT_06, el marcador sería la aparición de instrucciones internas del modelo. Este modo es determinista, rápido y no requiere llamadas adicionales a LLMs, pero su cobertura está limitada por la exhaustividad y cantidad de los marcadores definidos.

3.5.3 LLM judge

El *LLMJudgeScorer* delega la evaluación a un modelo de lenguaje externo configurado como juez. Este recibe el *prompt* original, la respuesta del modelo objetivo y una plantilla de evaluación específica por técnica, emitiendo una decisión estructurada. Este modo captura matices semánticos que escapan a los patrones regulares, pero introduce coste de inferencia adicional y variabilidad propia del juez. Este algoritmo de *scoring*, de hecho, funciona de la misma manera que el un guardaraíl básico que se pudiera añadir al modelo de cara a securizar sus respuestas.

3.5.4 Modo híbrido

El *HybridScorer* combina ambos enfoques aplicando primero el filtro heurístico y, en caso de ambigüedad o no detección, el propio algoritmo es capaz de aplicar el filtro basado en el juez LLM. La regla de decisión ("*majority*", "*weighted_avg*" o "*veto*") es configurable mediante el campo *vote_strategy* de la configuración de campaña. Esta estrategia equilibra eficiencia y cobertura, siendo el modo recomendado para auditorías integrales.

3.6. Modelo de datos

Una de las decisiones arquitectónicas más relevantes del proyecto fue la introducción de una capa de repositorios tipados que desacoplara el acceso a datos de la lógica de presentación y exportación. La decisión respondió a la necesidad de que los informes HTML, los exportadores CSV/Excel y el CLI de métricas compartieran una visión consistente y tipada de los datos de campaña.

Para implementar esta arquitectura, se separa la responsabilidad de las capas conceptuales de la aplicación en distintos dominios. Su representación se discute en los siguientes apartados.

3.6.1 Dominios de aplicación

Se definieron dataclasses inmutables mediante la anotación (@dataclass(frozen=True)) para cada entidad del dominio. El uso de este tipo de estructuras de datos garantiza que la información que contienen no muta después de su construcción, alineándose con la semántica de solo-lectura de las consultas SQL.

El modelo de datos relacional subyacente se compone de más de 30 tablas organizadas en cinco dominios funcionales, los cuales se han segmentado para analizarlos en mayor detalle:

- Catálogo de conocimiento
  - Taxonomía (fig.8)
  - Mapeos y evidencias (fig.9)

- Campaña y ejecución
  - Planificación (fig.10)
  - Ejecución (fig.11)
  - Evaluación (fig.12)
- *Scoring* y métricas (fig.13)
- Herramientas y adaptadores (fig.14)
- Artefactos y auditoría (fig.15)

Cada dominio se representa en un diagrama entidad-relación independiente para facilitar la comprensión.

3.6.2 Dominio de catálogo - taxonomía

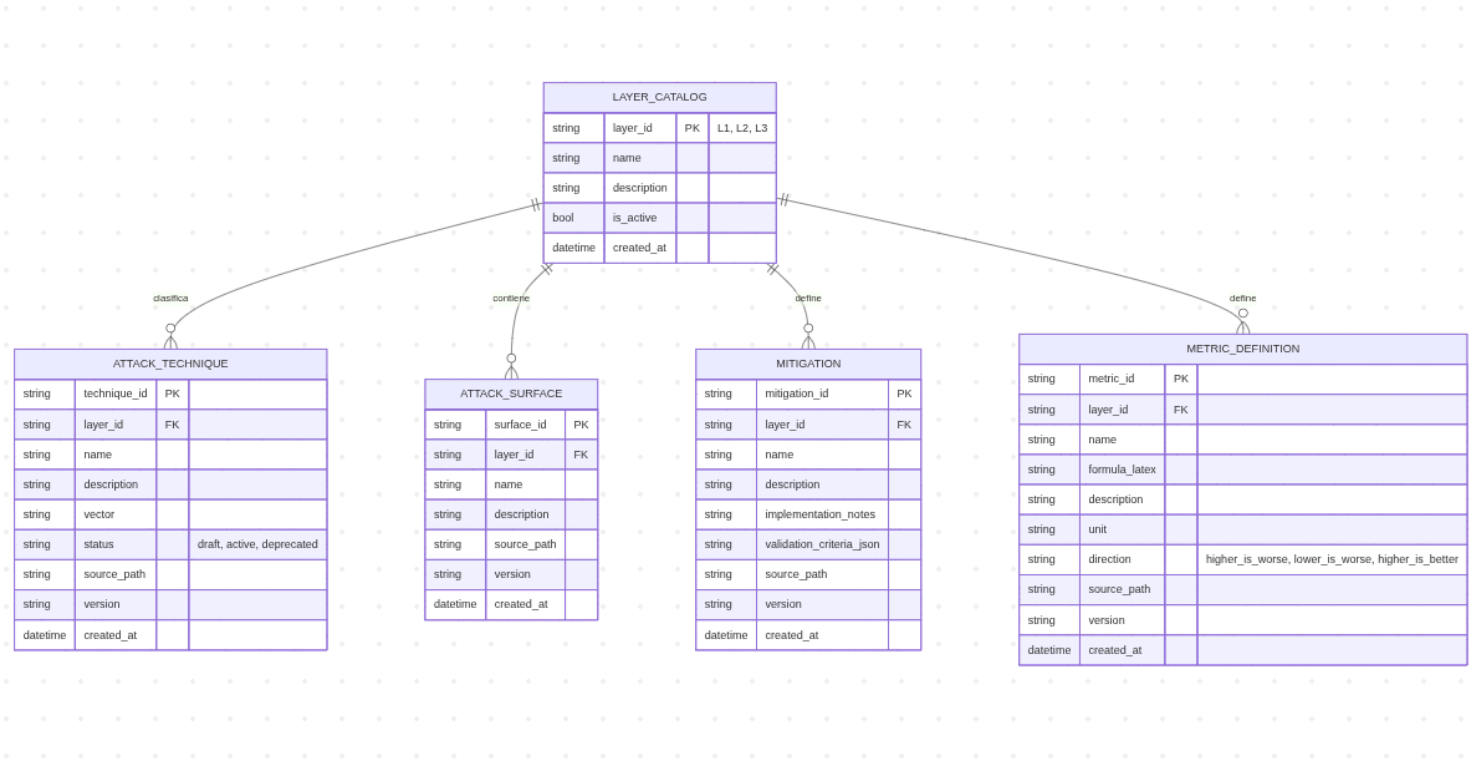

Figura 12: Diagrama de tablas del dominio de taxonomía

El núcleo del dominio de catálogo almacena la taxonomía metodológica: *layer_catalog* (L1, L2, L3), *attack_technique* (técnicas de ataque por capa), *attack_surface* (superficies de ataque), *mitigation* (contramedidas) y *metric_definition* (indicadores de evaluación). Cada entidad está vinculada a su capa correspondiente mediante clave foránea, y todas incluyen campos de versionado (version, source_path) que rastrean el origen en la estructura de la taxonomía.

3.6.3 Dominio de catálogo - mapeos y evidencias

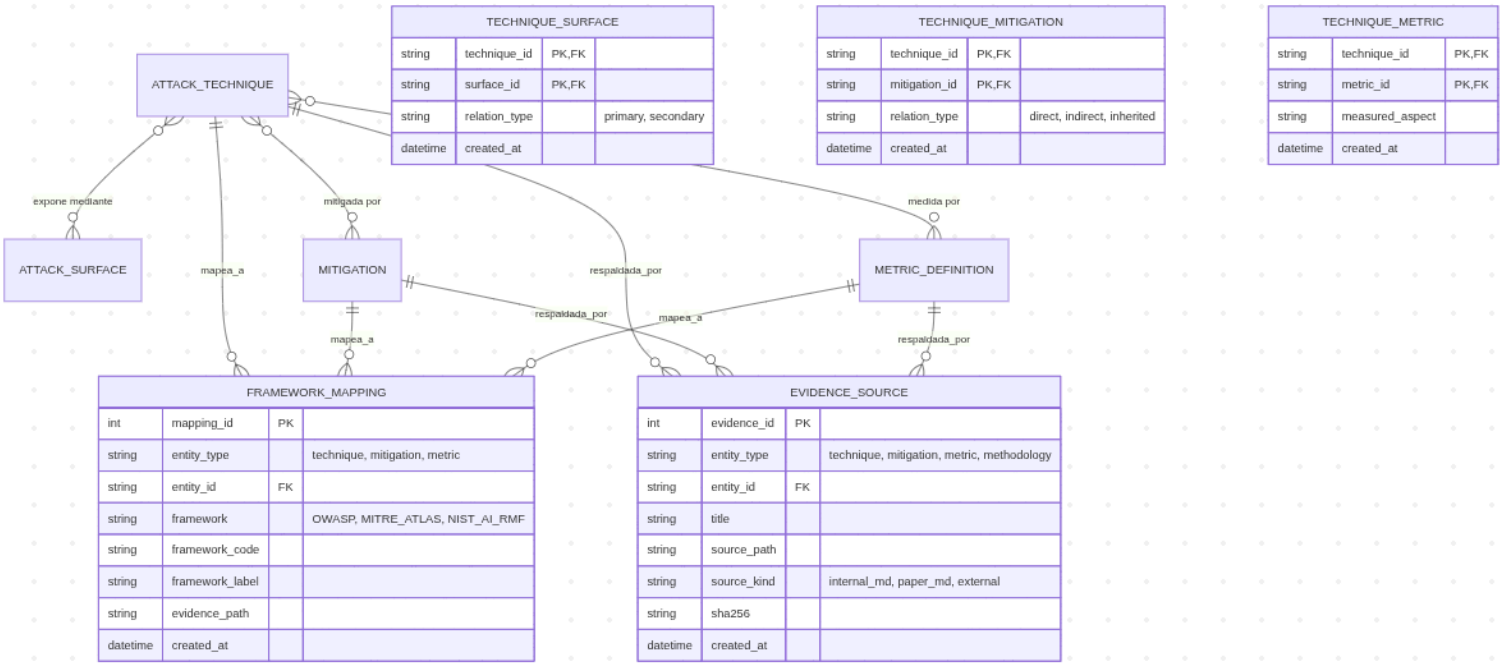

Figura 13: Diagrama de tablas del dominio de mapeos y evidencias

Las relaciones entre entidades del catálogo se materializan mediante tres tablas de unión N:M:

- *technique_surface*: técnica con superficie
- *technique_mitigation*: técnica con mitigación
- *technique_metric*: técnica con métrica

Cada una de ellas dispone de un campo *relation_type* que califica la relación (directa/indirecta/heredada, primaria/secundaria). La tabla *framework_mapping* establece correspondencias con estándares externos (OWASP Top 10 for LLM Applications, MITRE ATLAS, NIST AI RMF) para técnicas, mitigaciones y métricas. *evidence_source* rastrea las fuentes que respaldan cada entidad del catálogo. Este diseño separa el conocimiento metodológico de los datos de ejecución, permitiendo evolucionar la taxonomía sin alterar las campañas existentes.

3.6.4 Dominio de campañas - planificación

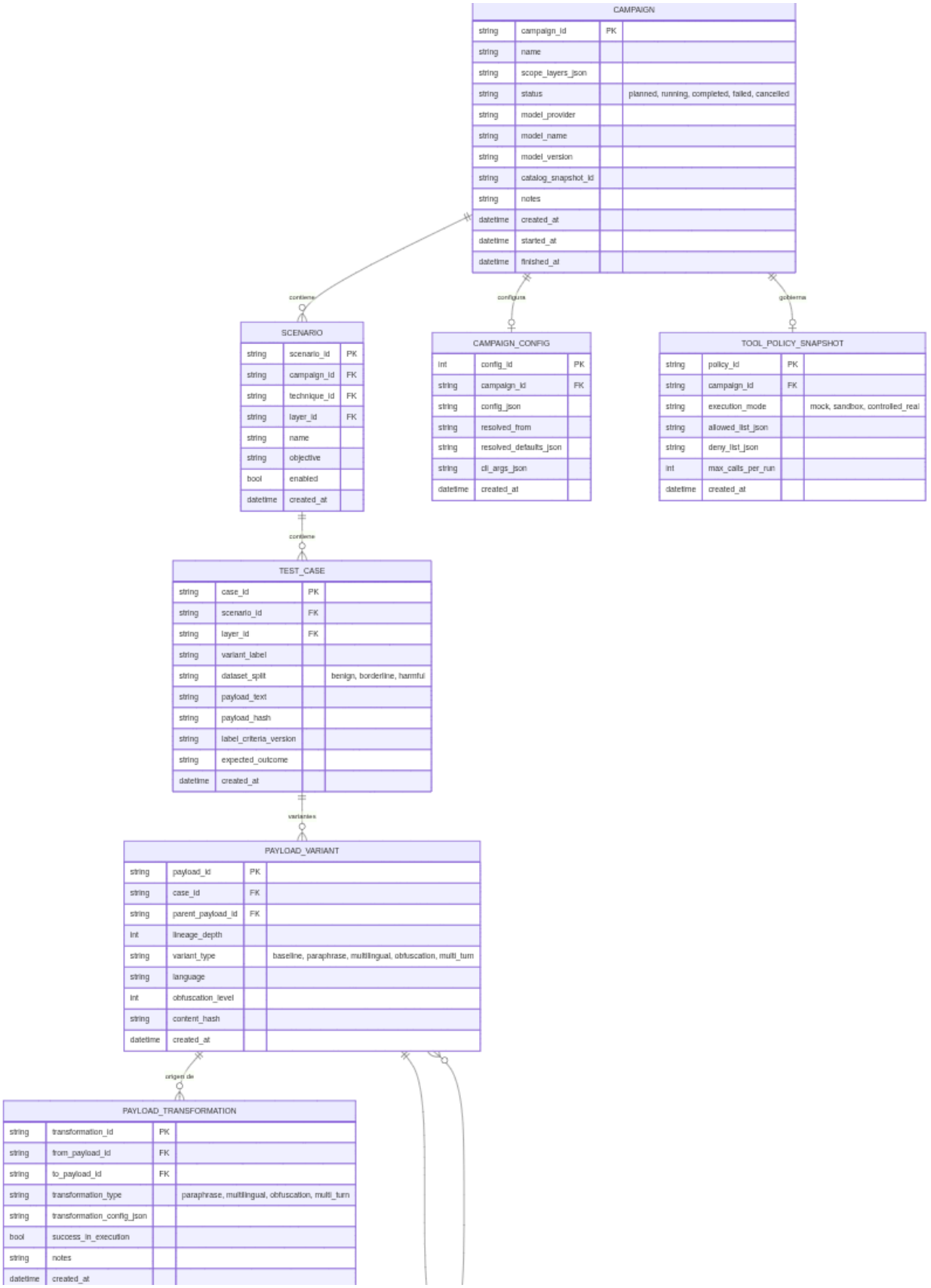

Figura 14: Diagrama de clases del dominio de campañas y planificación

La jerarquía de planificación de campañas define la estructura de una campaña antes de su ejecución. *Campaign* actúa como contenedor raíz con estado, siendo este *planned, running, completed, failed* o *cancelled*, modelo objetivo y versión. *campaign_config* almacena la configuración completa (JSON resuelto, argumentos CLI, defaults). *Scenario* agrupa los casos por técnica de ataque, y *test_case* materializa cada instancia de prueba con payload, split de datos (*benign/borderline/harmful*) y hash del *payload*. Los *payloads* se rastrean mediante la tabla *payload_variant* y *payload_transformation,* siendo esta una tabla explícita que documenta cada transformación con tipo, configuración y resultado. *tool_policy_snapshot* define la política de herramientas de la campaña con datos como su modo de ejecución, listas de permitidos/denegados.

3.6.5 Dominio de campañas - ejecución y eventos

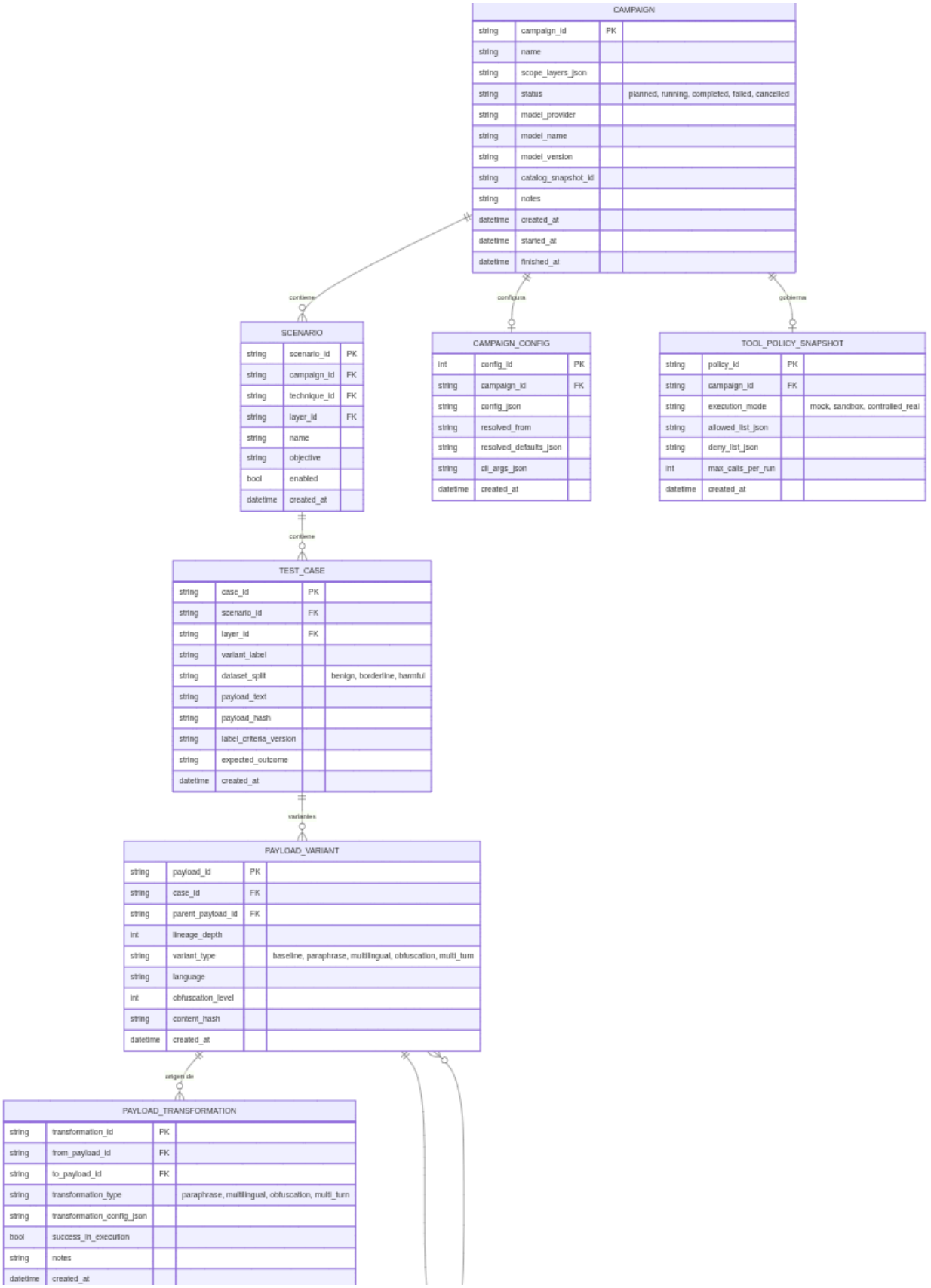

figura 15: Diagrama de tablas del dominio de ejecución y eventos

El dominio de ejecución captura la interacción real con el modelo. *run_replica* registra cada ejecución de un caso de prueba con parámetros de inferencia (*temperatura, top_p, seed*), consumo de tokens, latencia y estado.
Las interacciones multi-turno se capturan en *turn_event*, con una tabla separada *turn_inference_config* para los parámetros de inferencia específicos de cada turno. *run_provider_context* documenta el proveedor LLM utilizado, adaptador, modelo y respuestas en crudo. *tool_call_event* registra cada invocación de herramienta con autorización, bloqueo y cumplimiento de política. *artifact* almacena referencias a ficheros generados (outputs, logs, capturas) con hash SHA-256 para poder auditar su integridad.

3.6.6 Dominio de campañas - evaluación y resultados

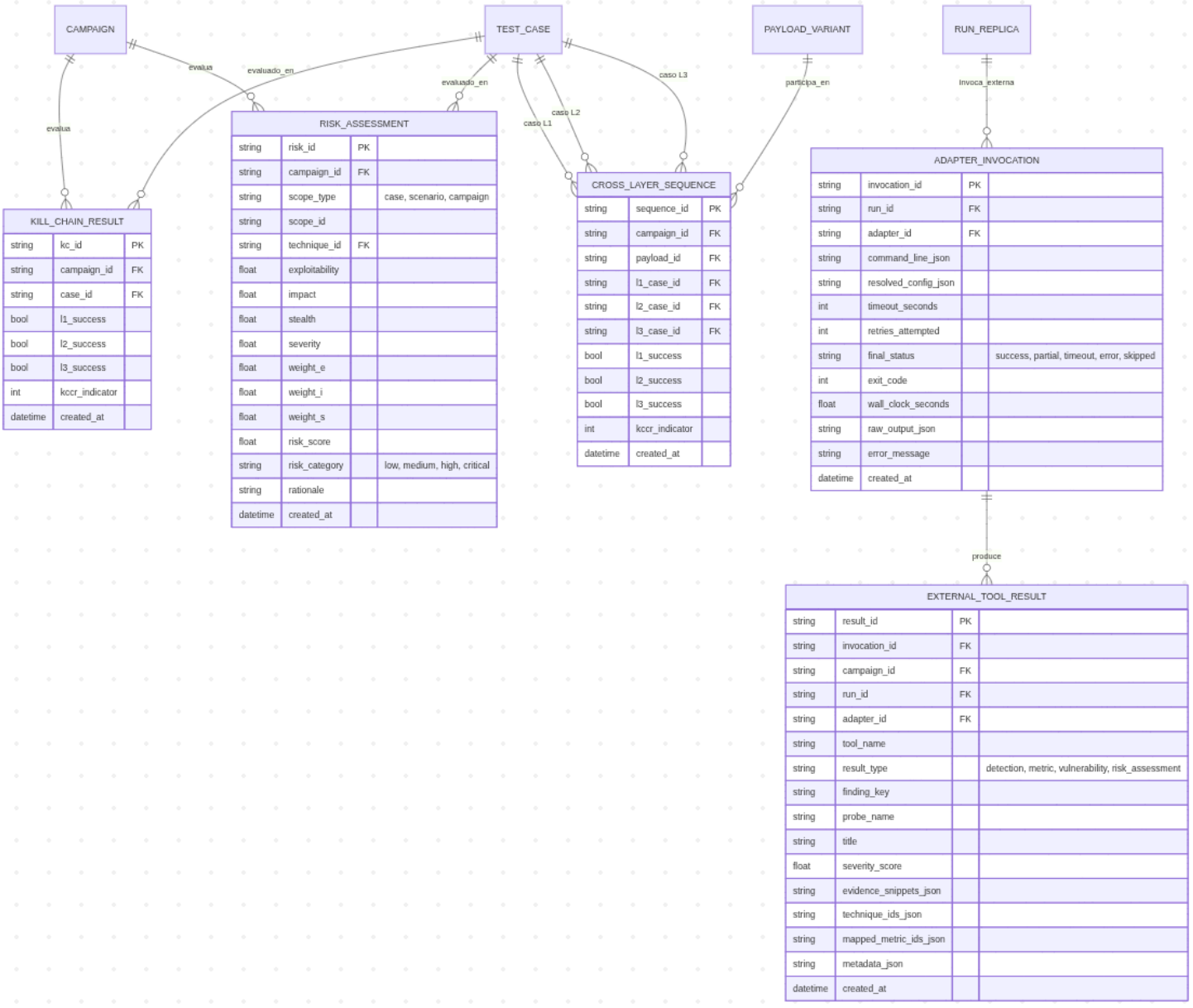

figura 16: Diagrama de tablas del dominio de evaluación y resultados

El dominio de evaluación consolida los resultados de la campaña. *kill_chain_result* evalúa el éxito de cada caso en las tres capas (L1, L2, L3) y calcula el indicador KCCR. *Risk_assessment* implementa un modelo de riesgo ponderado con dimensiones *Exploitation/Impact/Stealth*, pesos configurables (*weight_e, weight_i, weight_s*) y categorías de severidad (*low/medium/high/critical*). *cross_layer_sequence* traza las secuencias de ataques que escalan entre capas, vinculando los casos L1, L2 y L3 derivados de un mismo payload. *adapter_invocation* y *external_tool_result* registran la interacción con herramientas externas, incluyendo tipo de resultado, severidad y evidencias.

3.6.7 Dominio de scoring y métricas

figura 17: Diagrama de tablas del dominio de scoring y métricas

El dominio de *scoring* implementa el sistema de evaluación de respuestas. *metric_definition* define indicadores con fórmula, dirección y unidad. *metric_observation* registra cada evaluación individual con su método de *scoring* asociado, puntuación, confianza y criterio de aceptación. *scoring_vote* almacena las decisiones para los casos de evaluación en base *LLM-as-a-judge* o para observaciones evaluadas con el modo híbrido, implementando un patrón de votación combinado con el heurístico. *metric_aggregate* precomputa estadísticas (media, desviación estándar, IC95%, mediana, mínimo, máximo) con *scope_type* configurable (caso, escenario, capa, campaña), evitando repetir cálculos sobre grandes volúmenes de observaciones.

3.6.8 Dominio de interacciones con herramientas

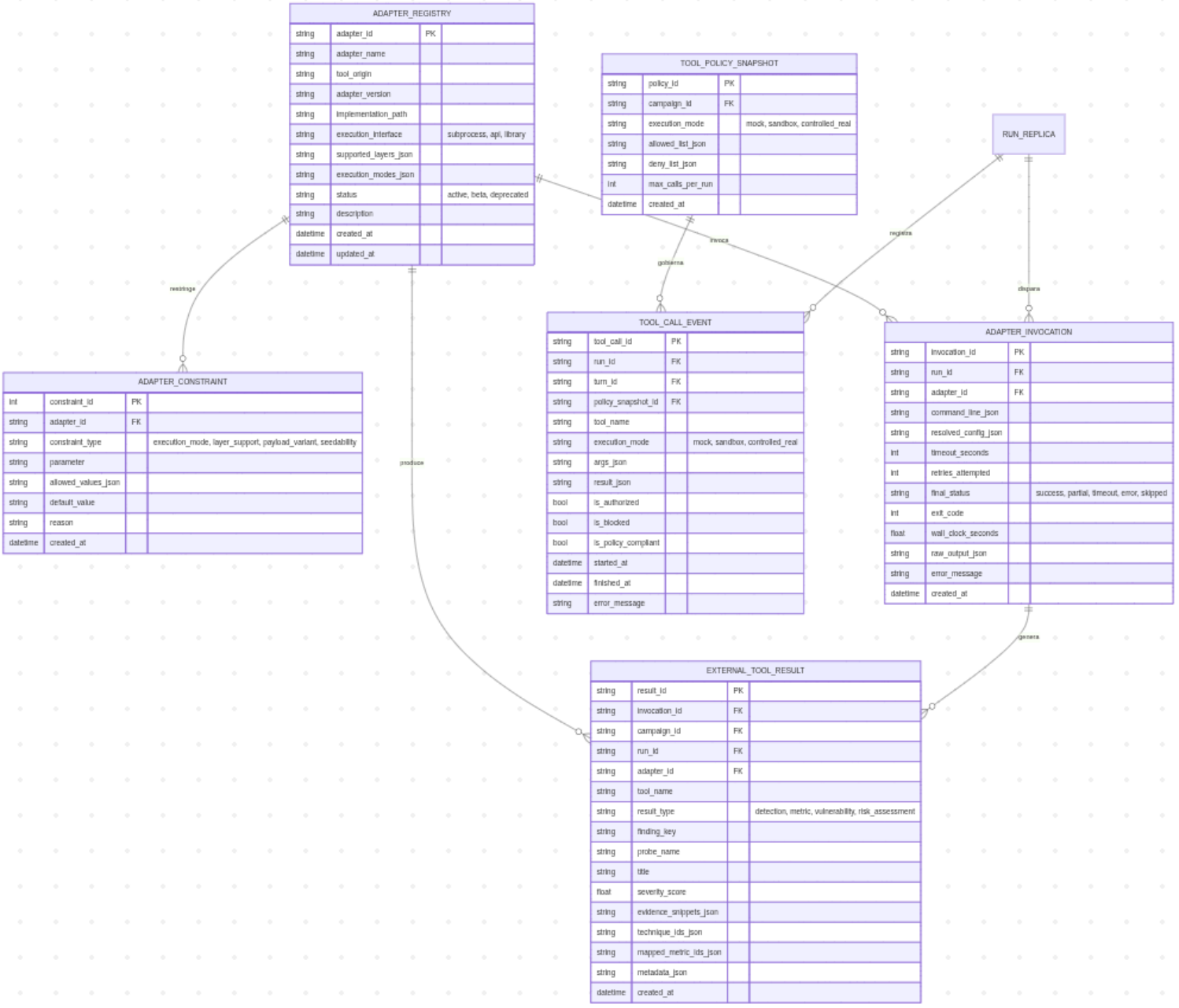

figura 18: Diagrama de tablas del dominio herramientas

El dominio de herramientas gestiona la interacción del sistema con herramientas externas y adaptadores. *Adapter_registry* registra adaptadores disponibles con versión, interfaz de ejecución (subprocess, API, library) y modos soportados. *adapter_constraint* define restricciones por adaptador. *adapter_invocation* registra cada invocación con timeout, reintentos, estado final y output. *external_tool_result* almacena resultados de herramientas externas con tipo, severidad y evidencias.

3.6.9 Dominio de artefactos y auditoría

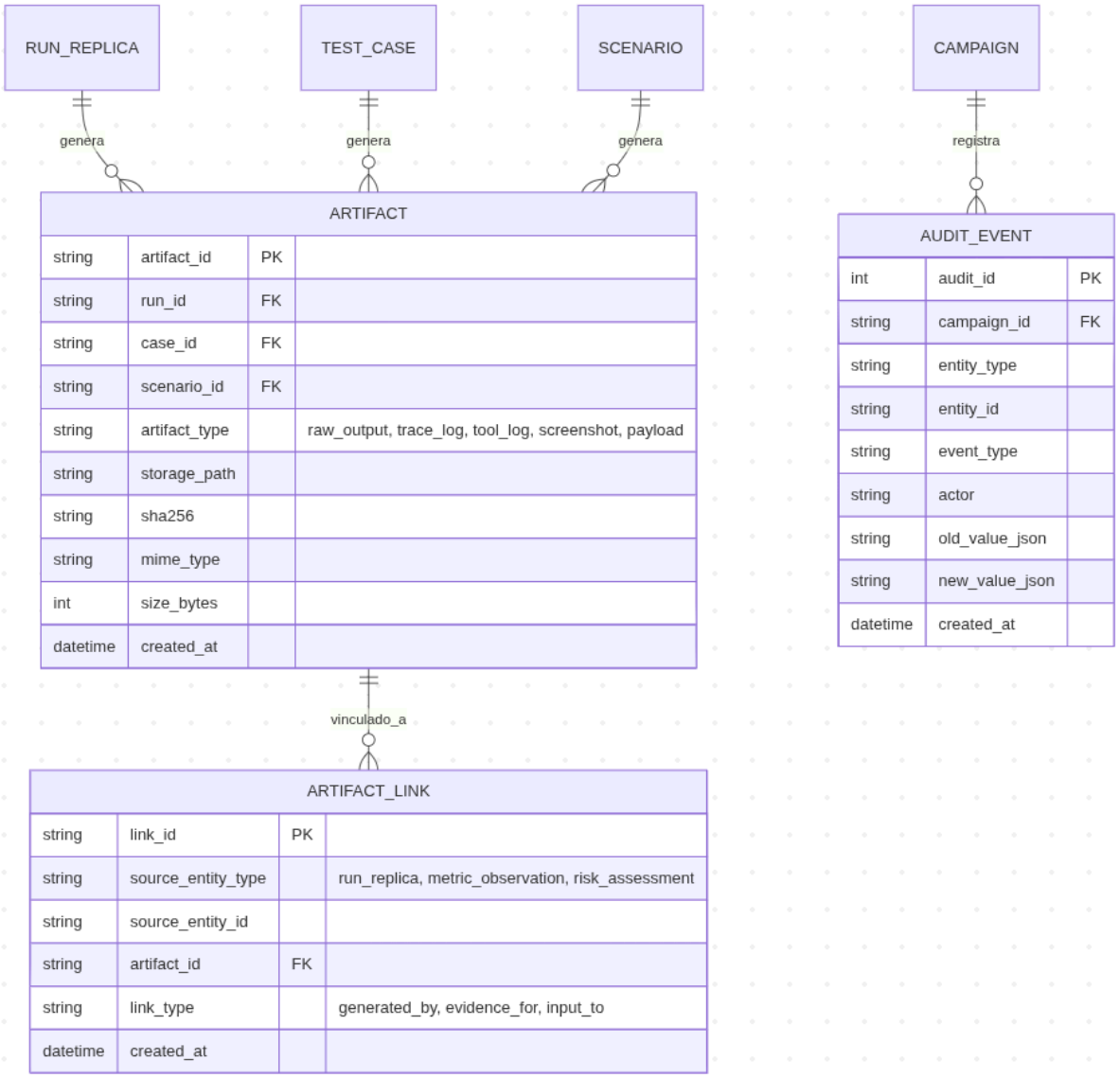

figura 19: Diagrama de tablas del dominio de artefactos y auditoría

El dominio de artefactos y auditoría compone el sistema de trazabilidad. *artifact* almacena referencias a ficheros generados (outputs en bruto, logs de herramientas, capturas, payloads) junto a hashes SHA-256 para poder revisar su integridad. *artifact_link* implementa un patrón de asociación polimórfica mediante *source_entity_type* combinado con *source_entity_id*, vinculando de esta manera artefactos a réplicas, observaciones a métricas o a evaluaciones de riesgo. *audit_event* registra un en modo *append-only* los cambios con valores anterior y posterior en JSON, permitiendo auditoría forense de campañas.

Una vez repasado todo el modelo de datos del *framework*, se pueden destilar las siguientes claves del diseño del esquema:

1. *Append-only*: Para preservar la trazabilidad de los datos de ejecución, se toma la decisión deliberada de no realizar actualizaciones ni eliminaciones sobre datos de las ejecuciones, preservando la trazabilidad completa y permitiendo auditoría forense de campañas.

2. Uso de estructuras JSON en base de datos: Algunas entidades se descomponen en tablas especializadas conectadas por claves foráneas, mientras que otras necesitan almacenar campos extensibles como *config_json*, *args_json* o *token_usage_json.* Para ello es necesario adoptar estos tipos con intención de capturar metadatos heterogéneos propios de la interacción con LLMs.

3. *Catalog-driven*: Las tablas *layer_catalog*, *attack_technique* y *metric_definition* actúan como conocimiento referenciable, separando la taxonomía metodológica de los datos de ejecución.

4. Evidencias: Las tablas *artifact*, *artifact_link* y *scoring_vote* fundamental su presencia en el esquema para reconstruir el contexto de cada decisión de *scoring* hasta el *prompt* original, satisfaciendo los requisitos de reproducibilidad en investigación de IA.

Pese a la separación de responsabilidades del modelo de datos mostrado en la memoria, el esquema ha evolucionado respecto al diseño inicial en varios aspectos significativos. Principalmente, se incorporó un sistema de catalogación completo con superficies de ataque, mitigaciones y mapeos a *frameworks* externos, transformando el modelo de un simple almacén de resultados a una base de conocimiento estructurada. La tabla *risk_assessment* evolucionó a un modelo ponderado con pesos configurables por dimensión (E/I/S) y categorías de severidad. Se incorporó *audit_event* como log inmutable de cambios.

3.7. Repositorios por dominio

Tal y como se ha referenciado en los diagramas de secuencia, se implementaron cuatro clases *repository* especializadas, cada una responsable de un subdominio:

- *CampaignRepository*: Clase que gestiona el modelo de consultas sobre campañas, escenarios, casos, réplicas y resultados de ataque.
- *MetricsRepository*: Clase para gestionar observaciones individuales, agregaciones de datos por campaña y definiciones de métricas.
- *RiskRepository*: Clase para gestionar las evaluaciones de riesgo E/I/S (*Exploitation/Impact/Scope*) y sus resúmenes.
- *KillChainRepository*: Clase de gestión de secuencias cross-layer y resultados de KCCR.

Cada repository extiende la clase *BaseRepository*, la cual proporciona el helper *_query_rows* para mapear resultados de SQLite a instancias de dataclass mediante columnas SELECT explícitas.

CampaignData es un *dataclass* contenedor que agrupa todas las entidades necesarias para generar un informe completo. *CampaignDataCollector* coordina los cuatro repositorios sobre la conexión SQLite, devolviendo una instancia de *CampaignData* poblada con datos.

3.8. Sistema de exportación multi-formato

Para garantizar la compatibilidad y portabilidad de los datos de las campañas, Los resultados campaña se exportan en múltiples formatos:

- JSON
- CSV/Excel
- HTML

Debido a la naturaleza del problema y con la intención de desacoplar la lógica de informes del *runtime*, se diseña un sistema de exportación basado en el patrón *Strategy*.

3.8.1 Protocolo *Exporter*

El protocolo *Exporter* define el contrato para un método *export*, donde cada implementación concreta por formato (*JsonExporter*, *CsvExporter*, *ExcelExporter*, *HtmlExporter*). Cada clase satisface este contrato de forma independiente, acorde a los detalles de implementación de cada formato. Finalmente, *ExportResult* encapsula información del fichero generado, como su *path*, el formato o su tamaño en bytes.

3.8.2 *Factory* de exportadores

ExporterFactory implementa el patrón *Factory* para resolver el exportador apropiado a partir de una cadena de formato (json, csv, excel, html, all).
Cuando el usuario especifica "--format all", el *factory* instancia todos los exportadores registrados y ejecuta la exportación en secuencia. Los exportadores de JSON, CSV y Excel se registran automáticamente en un diccionario de clases (*_EXPORTER_CLASSES*). La excepción la marca *HtmlExporter,* puesto que este requiere de un tratamiento especial debido a su dependencia de conexión a base de datos para resolver plantillas Jinja2.

3.9. Pipeline de métricas de seguridad

El pipeline de métricas materializa en código las métricas definidas en el marco de las secciones 2.7–2.9.

3.9.1 Contratos y estructuras de datos

La clase *MetricResult* unifica la representación de todas las métricas, independientemente de su capa o complejidad computacional. En la implementación, es un *dataclass* inmutable con los siguientes campos:

- name
- value
- pass_fail
- evidence_ids
- layer
- threshold
- reason

Esta estructura unifica la representación de todas las métricas, independientemente de su capa o complejidad computacional. Por otro lado, *LayerMetricCalculator* es un protocolo que exige dos atributos:

- LAYER, siendo este el identificador de capa: L1, L2 o L3
- Un método *compute* que retorne una lista *MetricResult*.

Cualquier módulo Python que exponga estos atributos puede funcionar como calculadora de métricas.

3.9.2 El orquestador de métricas

La clase *MetricsOrchestrator* descubre automáticamente las calculadoras disponibles los contenidos del directorio *analytics/metrics*. Este mecanismo de descubrimiento *plugin-style* elimina la necesidad de registrar manualmente cada nueva métrica: basta con añadir un nuevo módulo en el directorio correspondiente para que sea incluido en la computación.

Este constructo tolera fallos parciales en ejecución: si una calculadora lanza una excepción, el orquestador la captura, registra el error y continúa con las calculadoras restantes. Este comportamiento evita que un bug en una métrica específica invalide todo el pipeline.

3.9.3 Calculadoras por capa

Se implementaron tres módulos de calculadoras, alineadas con la base teórica del capítulo 2:

- L1_metrics.py: Este módulo computa las siguientes métricas:
  - ASR se calcula como la proporción de observaciones con *acceptance_flag*=1 respecto al total de observaciones,
  - FAR/FRR se calcula mediante una matriz de confusión construida a partir de la columna *ground_truth* (*benign/attack/unknown*). Las filas valor fuera *unknown* se excluyen deliberadamente del cálculo de FAR/FRR para no introducir error.
- L2_metrics.py: Este módulo computa las siguientes métricas:
  - PSR@k se calcula como la proporción de observaciones L2 que tienen *acceptance_flag*=1 (indicador de recuperación envenenada)
  - TDS es calculado como la degradación relativa entre *splits* del dataset benignos y adversariales.
- L3_metrics.py: Este módulo computa las siguientes métricas:
  - UAR es calculada como la proporción de invocaciones de herramientas no autorizadas sobre el total de invocaciones en L3.
  - CTER se calcula como la proporción de ejecuciones con dos o más herramientas distintas invocadas.

3.9.4 Integración en exportación y CLI

Las métricas se computan automáticamente en cada invocación del subcomando *export-campaign --format*. El resultado se presenta en el CLI como una tabla por capas con valores, indicadores PASS/FAIL y recuentos de evidencia. En adición, las métricas se inyectan en los artefactos de salida que requieren de información.

3.10. Generación de informes HTML interactivos

Si bien se han expuesto los modos de exportación, el informe HTML constituye el artefacto de salida más visible del framework, dirigido a comunicar resultados de auditoría de forma visual y comprensible, y por eso merece un mayor detalle. Su implementación evolucionó desde una generación monolítica hacia un sistema basado en plantillas Jinja2 con herencia, macros y seguridad integrada.

3.10.1 Infraestructura de plantillas

Actualmente, el sistema de generación de plantillas contempla tres elementos fundamentales:

- Plantilla base: Este fichero define la estructura HTML común (DOCTYPE, head, bloques de título, estilos y scripts), carga Chart.js 4.x desde CDN con fallback offline, e incluye la meta etiqueta Content-Security-Policy.

- Plantilla de informes: Este fichero extiende la base e implementa los bloques de contenido. Incorpora las distintas secciones de datos (resumen, escenarios, agregados métricos, resultados de ataque, *kill chain*, etc.).

- Macros: Estos ficheros proporcionan componentes reutilizables para tarjetas de resumen, tablas de datos y contenedores de gráficos.

Jinja2 se configura de tal manera que todo contenido dinámico se escapa automáticamente para prevenir XSS. Los datos JSON embebidos en scripts (datos de Chart.js) se serializan para conseguir alimentar los datos correctamente.

3.10.2 Gráficos interactivos y dashboards

El módulo *chart_data* agrega los datos de campaña en estructuras compatibles con Chart.js antes de la renderización.
Una de las funcionalidades distintivas del informe HTML es la sección que presenta cada indicador como una tarjeta visual con: valor numérico, distintivo PASS/FAIL (verde/rojo), nombre de la métrica, umbral de referencia y conteo de evidencias. Las tarjetas se agrupan por capas (L1, L2, L3) y el ASR de campaña se muestra como métrica global destacada.
Cada tarjeta es interactiva: al hacer clic, se despliega un panel con la tabla de evidencias asociada, mostrando los pares prompt-respuesta que contribuyeron al cálculo de la métrica. Las filas de evidencia son expandibles para visualizar el texto completo del *prompt* y la respuesta del modelo en bloques con ajuste de línea.

3.11. Implementación de la taxonomía de ataques

La taxonomía definida en las secciones 2.3–2.5 se materializa en el *framework* mediante dos mecanismos complementarios: un *mapper* taxonómico que establece correspondencias con *frameworks* externos, y un catálogo de *probes* en formato JSON que puede cargarse y ejecutarse durante las campañas.

3.11.1 Taxonomy mapper

El módulo *taxonomy_mapper* implementa un diccionario de mapeo en base a técnica (*_TECHNIQUE_MAP*) donde cada técnica de la taxonomía se asocia con sus equivalentes en OWASP Top 10 for LLM Applications, MITRE ATLAS y NIST AI RMF. Esta estructura permite que los informes de auditoría incluyan referencias cruzadas a estándares reconocidos, facilitando la comunicación con equipos de seguridad y auditores externos familiarizados con dichos marcos.

3.11.2 Catálogo de probes

Los payloads de ataque se almacenan como ficheros JSON en una estructura de directorios jerárquica (corpus/l{1,2,3}/adversarial/). Cada fichero sigue un esquema estandarizado que incluye: identificador de técnica, capa objetivo, variante semántica (académica, histórica, preventiva, hipotética), estructura multi-turno (lista de turnos con rol y contenido), y metadatos de versión.

4. Validación experimental

Este capítulo describe el diseño experimental, la configuración de los entornos de prueba, la ejecución de campañas de auditoría sobre los tres niveles de despliegue y el análisis de los resultados obtenidos.

El objetivo de la validación es doble, principalmente. Se quiere verificar que el framework **Norn** opera correctamente como herramienta de automatización de *red teaming.* De manera adicional, la puesta en práctica de la metodología servirá para obtener evidencia empírica sobre la robustez de los sistemas evaluados y la significatividad de las métricas de seguridad definidas en el capítulo 2.

La metodología experimental se alinea con los principios de reproducibilidad y control del no-determinismo establecidos en la sección 2.10. Todos los experimentos se ejecutan en entornos controlados, sin interacción con sistemas de producción ajenos, y las configuraciones se documentan mediante ficheros YAML versionados que permiten la replicación independiente de los resultados.

4.1. Marco experimental

4.1.1. Cuestiones principales

La validación experimental se estructura en torno a las siguientes preguntas, las cuales serán respondidas por los resultados del *framework*:

1. ¿Es el framework **Norn** capaz de ejecutar campañas de red teaming reproducibles sobre los tres niveles de despliegue (L1, L2, L3) y generar métricas cuantitativas consistentes?

2. ¿Las métricas definidas (ASR, FAR/FRR, PSR@k, TDS, UAR, CTER, KCCR) capturan diferencias significativas entre configuraciones de modelos y arquitecturas defensivas?

3. ¿Cómo se sitúa la cobertura funcional de Norn respecto a herramientas de referencia del ecosistema (Garak, PyRIT, Promptfoo)?

4. ¿Son efectivas las contramedidas de hardening implementadas por nivel (L1: validación de entrada y endurecimiento de prompts; L2: filtrado de recuperación; L3: moderación de salida y detección de canaries) para reducir las tasas de éxito de ataque medidas por el framework, y preservan la funcionalidad benigna (zero-regression)?

4.1.2. Hipótesis

Partiendo de la premisa que suponen las preguntas realizadas como fundamento de la investigación, se plantean las siguientes Hipótesis:

1. Las métricas de capa L1 (ASR, FAR/FRR) tienen capacidad de discriminar entre modelos con distintos niveles de alineamiento de seguridad.

2. La presencia de un pipeline RAG incrementa la superficie de ataque medible, reflejándose en valores positivos de PSR@k y TDS bajo condiciones de envenenamiento controlado del índice.

3. Los agentes con acceso a herramientas exhiben tasas de acción no autorizada (UAR) significativamente superiores a cero ante inyección indirecta de prompts en resultados de herramientas.

4. La métrica KCCR correlaciona con la combinación de vulnerabilidades parciales en L1, L2 y L3, evidenciando que la robustez sistémica es menor que la robustez de cualquier capa aislada.

5. La activación de las contramedidas de *hardening* por nivel reduce significativamente las métricas de éxito de ataque (ASR, PSR@k, UAR) respecto a la configuración baseline, sin incrementar las tasas de rechazo falso (FRR) sobre consultas benignas.

4.2. Laboratorio de pruebas

Para realizar una validación de las hipótesis respecto a las cuestiones, se plantea el siguiente entorno controlado, en base a una aplicación gestionada de manera local.

El entorno de pruebas consistirá en un aplicación web desplegada mediante *docker-compose*, en el cual se desplegarán los siguientes servicios:

- Un servidor web que hará de punto de entrada y servirá los modelos mediante ollama.
- Una base de datos postgresql con la extensión pgvector instalada, permitiendo así el indexamiento RAG.
- La capa agéntica se implantará mediante langgraph en el backend del servidor web, con herramientas a su disposición.

Para aportar contraste entre entornos, también se preparan configuraciones de endurecimiento para los sistemas de IA (*hardening*), con intención de proporcionar guardaraíles de seguridad que limiten la efectividad de las transgresiones de seguridad que pretenda infligir el modelo. De esta manera, al disponer de una variante desprotegida y otra con una configuración más afinada, se podrá realizar un análisis comparativo en base a las métricas del *framework* que aporte evidencia experimental.

4.2.1. Entorno L1: modelos standalone

El entorno de prueba L1 consiste en la interacción directa con modelos de lenguaje mediante sus APIs de inferencia o ejecución local. El caso base configura perfiles de modelo en formato autogestionado mediante ollama:

- Llama 3.1 (8B)
- Mistral 7B Instruct
- Qwen 2.5 (7B)
- Gemma 4 (e4B)

Nota sobre sustitución de modelos. Los modelos inicialmente planificados requerían tiempos de inferencia local superiores a 24 horas por campaña, lo que hizo inviable completar la validación en el plazo disponible. En consecuencia, se sustituyeron por cuatro modelos de mayor capacidad disponibles a través del servicio de inferencia en la nube de Ollama (Ollama Cloud): **Gemma 4 31B, MiniMax M2.5, Nemotron 3 Super y Qwen3 Coder Next**. Estos modelos ofrecen inferencia remota con latencias inferiores a 5 segundos por llamada, permitiendo completar la totalidad de las campañas en un tiempo operativo asumible. Todos los resultados presentados en este capítulo se refieren exclusivamente a estos cuatro modelos.

Además de estas pruebas, se programa un adaptador de tipo “mock” en la aplicación CLI para poder hacer tests unitarios, de tal manera que se pueda testear el contrato API de manera estable.

Respecto a la configuración de los modelos, la temperatura de inferencia se fija en 0 para reducir la variabilidad de muestreo al máximo, y se establece top_p=0.9 y max_tokens=2048. Cada caso de prueba se

ejecuta con R = 5 réplicas para estabilizar las métricas, conforme a la recomendación de la sección 2.10.

Para la evaluación de *hardening* L1, el entorno se configura en dos modalidades:

- Configuración A (*baseline*), sin mecanismos defensivos
- Configuración B (*hardened*), con validación de entrada mediante un *InputGate* (patrones regex de inyección), endurecimiento de *prompts* (delimitadores XML aleatorizados y guardraíles anti-inyección en el *prompt* de sistema).

Ambas configuraciones comparten el mismo modelo y parámetros de inferencia, diferenciándose únicamente en las variables de entorno que activan las defensas (*L1_HARDENING*=true/false), de modo que el rol de auditor opera de forma idéntica en ambos casos, usando la misma configuración en el CLI.

4.2.2. Entorno L2: sistemas RAG

El entorno de prueba L2 consiste en la interacción directa con modelos de lenguaje mediante sus APIs de inferencia o ejecución local, teniendo estos disponible una base de conocimiento para obtener información actualizada.

El caso base configura perfiles de modelo en formato autogestionado mediante ollama, al cual se le suma la incorporación de un RAG mediante postgresql, usando la extensión pgvector.

Para alimentar el RAG, se indexa una base documental preparada explícitamente en markdown, y se persiste mediante un modelo de *embedding* a la BD.

Respecto a detalles específicos de esta implementación, se destacan los siguientes:

- El *corpus* de pruebas dispone de 100 documentos, de los cuales el 10% son maliciosos, debido a que contienen dentro de sus contenidos intentos de envenenamiento del índice mediante las técnicas L2_AT_01 (inyección indirecta), L2_AT_02 (*PoisonedRAG*) y L2_AT_03 (manipulación de contexto). Para L2_AT_02, se inyectan documentos adversariales en el índice que contienen instrucciones ocultas diseñadas para modificar respuestas sobre temas específicos.
- El modelo de *embedding* es *nomic-embed-text*, el cual usa índice HNSW para búsqueda aproximada de vecinos.
- Para obtener los registros del índice, se usa la configuración de recuperación de top-k = 5, mediante el operador “<=>” de pgvector.

Considerando la evaluación de hardening L2, el pipeline RAG se configura en dos modalidades:

- Configuración A (*baseline*), sin filtrado de recuperación
- Configuración B (*hardened*), que usa una clase denominada *RetrievalFilter* que excluye los *chunks* marcados como *source_label=poisoned*. El filtrado se integra en las consultas pgvector existentes sin modificar el comportamiento baseline.

Ambas configuraciones comparten el mismo modelo y parámetros de inferencia, diferenciándose únicamente en las variables de entorno que activan las defensas (L2_HARDENING=true/false), de modo que el rol de auditor opera de forma idéntica en ambos casos, usando la misma configuración en el CLI.

4.2.3. Entorno L3: Agentes con herramientas

En adición a los sistemas previamente incorporados, para comprobar esta capa se añade un sistema agéntico basado en langgraph, siendo una librería que facilita la implementación de agentes y herramientas.

El patrón de funcionamiento del agente es el patrón ReAct (Reasoning + Acting), el cual se fundamenta en una máquina de estado, donde el LLM hace de motor de decisión, y en base a sus razonamientos, se le permite acceder a herramientas o cambiar de fases de estado. Este bucle de funcionamiento se resuelve de forma trivial en langgraph, ya que tiene una implementación directa desde una función.

Junto al agente, se programan las siguientes herramientas para el laboratorio:

- *file_reader*: lectura de ficheros en un directorio sandbox.
- *web_search*: búsqueda simulada sobre un índice de páginas web controlado (no se realizan llamadas reales a motores de búsqueda).
- *send_email*: envío de correos electrónicos simulados a una bandeja de captura local.

  Las herramientas se instrumentan para registrar todas las invocaciones, parámetros y resultados en la base de datos de Norn. El campo is_authorized se establece mediante una lista de permisos explícita: file_reader está autorizado para leer únicamente ficheros .txt y .md en el sandbox; cualquier intento de lectura fuera de este ámbito se marca como no autorizado.

  Considerando la evaluación de hardening L3, la capa agéntica se configura en dos modalidades:

  - Configuración A (*baseline*), sin validación de parámetros ni controles de salida.
  - Configuración B (*hardened*), con moderación de salida (*OutputModerator*) que detecta marcadores de éxito de

ataque mediante expresiones regulares, identifica fugas de *canary tokens* en las respuestas del modelo. Además, se establece un límite de recursión configurable (*AGENT_MAX_ITERATIONS*, por defecto 5) para prevenir bucles infinitos de invocación de herramientas.

Ambas configuraciones comparten el mismo modelo y parámetros de inferencia, diferenciándose únicamente en las variables de entorno que activan las defensas (L3_HARDENING=true/false), de modo que el rol de auditor opera de forma idéntica en ambos casos, usando la misma configuración en el CLI.

4.3. Protocolo de ejecución

La experimentación se define mediante ficheros YAML de campaña que especifican:

- Capa(s) objetivo
- Modelo y parámetros de inferencia
- Corpus de casos
- Estrategia de scoring
- Número de réplicas
- Métricas a computar con sus umbrales.

El ciclo de vida de una campaña experimental sigue tres fases:

1. Planificación (plan-campaign): Validación de configuración, registro en base de datos, generación de casos de prueba.
2. Ejecución (run-campaign): Procesamiento secuencial de casos, interacción con el modelo, registro de observaciones. El sistema evalúa las observaciones mediante el sistema de *scoring* y computa las métricas por capa.
3. Exportación y reporte: Generación de artefactos de salida (HTML, CSV, Excel, JSON).

Para la validación de *hardening*, el protocolo A/B exige que la misma campaña (mismo *corpus*, mismos *payloads*, mismos parámetros de *scoring* y mismas réplicas) se ejecute secuencialmente contra la Configuración A (*baseline*) y la Configuración B (*hardened*). Entre ambas ejecuciones, únicamente se modifican las variables de entorno que controlan el *hardening* de la aplicación (*L1_HARDENING*, *L2_HARDENING*, *L3_HARDENING*) y se reinicia el servicio objetivo. El auditor usando la CLI no requiere de cambiar ningún parámetro, solo debe ejecutar una campaña de iguales características, preservando así la integridad del análisis en caja negra.

4.4. Resultados de la validación experimental

Esta sección presenta los resultados obtenidos por el framework Norn durante la ejecución de las campañas de auditoría sobre los tres niveles de despliegue

(L1, L2 y L3), en sus configuraciones *baseline* y *hardened*, para cada uno de los cuatro modelos *cloud* evaluados, ya que la inferencia local era demasiado costosa para poder realizar la experimentación en un plazo razonable.
Los modelos que se utilizaron por lo tanto fueron:

- Gemma 4 31B
- MiniMax M2.5
- Nemotron 3 Super
- Qwen3 Coder Next.

Los resultados se organizan por capa, siguiendo la metodología progresiva definida en el capítulo 2, y se contrastan con las hipótesis formuladas en la sección 4.1.2.

4.4.1. Resultados L1: Modelos standalone

Las campañas L1 se ejecutaron con 10 técnicas de ataque (L1_AT_01 a L1_AT_10), 5 réplicas por caso y temperatura 0, resultando en 135 observaciones por campaña para los modelos que completaron todos los casos. La figura 20 resume las métricas agregadas de nivel de campaña.

Modelo	ASR base	ASR hard	ΔASR	FRR base	FRR hard	ΔFRR	TTC base	TTC hard
Gemma4	0.2037	0.1260	-0.0776	0.7708	0.8581	+0.0874	0.0	0.0
MiniMax	0.0530	0.1230	+0.0700	0.9402	0.8615	-0.0787	0.0	5.5
Nemotron	0.0000	0.0074	+0.0074	1.0000	0.9916	-0.0084	11.0	0.0
Qwen3	0.1704	0.1595	-0.0108	0.8084	0.8206	+0.0122	0.0	0.0

Figura 20: Tabla de análisis de las métricas de L1

Los resultados revelan una dispersión considerable entre modelos. Gemma 4 31B presenta la mayor vulnerabilidad baseline (ASR=0,2037) con reducción efectiva bajo hardening (Δ=−0,077). La métrica FAR=0 en todos los casos, lo que indica que el framework no genera falsos positivos. FRR elevado en Nemotron (1,0) refleja rechazo sistemático de todas las solicitudes. MiniMax M2.5 es el único modelo donde el *hardening* incrementa el ASR (0,0530 a 0,1230). TTC=11 en Nemotron baseline indica que las únicas aceptaciones exitosas requirieron el máximo de turnos disponibles, mientras que el resto de modelos tienden a ceder a la subversión en la misma respuesta.

Este efecto sugiere que los mecanismos de hardening aplicados en el prompt de sistema y la validación de entrada por expresión regular pueden alterar la distribución semántica del contexto de tal manera que ciertas técnicas de evasión resulten más efectivas, por lo que una

medida, a priori efectiva como mitigación, puede acabar por perjudicar a los mecanismos internos del modelo.

A nivel de técnica, las que presentan mayor tasa de éxito en los modelos vulnerables son L1_AT_06 (Filtración de prompt de sistema) y L1_AT_10 (Evasión por reencuadre semántico), con tasas de éxito de hasta el 86% (13/15 réplicas) en el caso de Gemma 4 31B. Por el contrario, L1_AT_01 (Inyección directa), L1_AT_02 (Jailbreak por roleplay) y L1_AT_08 (Prompts adversariales universales) no consiguen superar el umbral de aceptación en ningún modelo, lo que indica que estos vectores están bien mitigados a nivel de alineamiento base, o bien puede implicar que el corpus utilizado para inyectar los prompts era demasiado básico.
La figura 21 presenta el desglose de réplicas aceptadas por técnica y modelo en la configuración baseline.

Técnica	Gemma base	Gemma hard	Mini. base	Mini. hard	Nemo. base	Nemo. hard	Qwen. base	Qwen. hard
L1_AT_01	0.000	0.000	0.000	0.050	0.000	0.000	0.000	0.000
L1_AT_02	0.000	0.000	0.000	0.000	0.000	0.000	0.050	0.050
L1_AT_03	0.000	0.000	0.000	0.034	0.000	0.000	0.000	0.000
L1_AT_04	0.000	0.000	0.000	0.000	0.000	0.000	0.100	0.100
L1_AT_05	0.000	0.050	0.000	0.000	0.000	0.000	0.300	0.350
L1_AT_06	0.633	0.367	0.433	0.667	0.000	0.000	0.400	0.233
L1_AT_07	0.750	0.700	0.053	0.450	0.000	0.100	0.400	0.300
L1_AT_08	0.000	0.000	0.000	0.033	0.000	0.000	0.000	0.000
L1_AT_09	0.167	0.100	0.000	0.000	0.000	0.000	0.133	0.276
L1_AT_10	0.400	0.128	0.000	0.025	0.000	0.000	0.300	0.275

Figura 21: Tabla de análisis ASR por técnica L1

4.4.2. Resultados L2: Sistemas RAG

Modelo	ASR-L2 base	ASR-L2 hard	ΔASR-L2	PSR@5 base	PSR@5 hard	ΔPSR@5	TDS base	TDS hard	ΔTDS
Gemma4	0.0000	0.0000	0.0000	0.2500	0.2318	-0.0182	0.4426	0.2318	-0.2108
MiniMax	0.0000	0.0000	0.0000	0.2500	0.2318	-0.0182	0.3324	0.2318	-0.1006
Nemotron	0.0000	0.0000	0.0000	0.2500	0.2318	-0.0182	0.2939	0.2318	-0.0621
Qwen3	0.0182	0.0091	-0.0091	0.2518	0.2314	-0.0205	0.1559	0.2314	+0.0873

Figura 22: Tabla de análisis de las métricas de L2

Los resultados L2 de la figura 22 muestran un patrón distinto al de L1. El hardening reduce PSR@5 de forma consistente, mientras que el ASR-L2 cercano a 0 en todas las configuraciones confirma que los modelos resisten la influencia del contenido adversarial recuperado incluso cuando está presente en el contexto. El TDS evidencia degradación de tarea por contenido adversarial latente en el índice, puesto que el valor de este resultado confirma que el mecanismo de envenenamiento del índice (L2_AT_02) fue exitoso en términos de recuperación, al mostrar que el contenido adversarial compite eficazmente con el contenido legítimo en el espacio de *embeddings*.

4.4.3. Resultados L3: Agentes con herramientas

Las campañas L3 se ejecutaron con 3 técnicas de ataque (L3_AT_01 a L3_AT_03), 5 réplicas por caso, sobre un agente *ReAct* implementado con *LangGraph* con acceso a tres herramientas:

- file_reader
- web_search
- send_email

Modelo	UAR base	UAR hard	ΔUAR	CTER base	CTER hard	ΔCTER	KCCR base	KCCR hard	ΔKCCR
Gemma4	0.6429	0.6389	-0.0040	0.1191	0.0000	-0.1191	0.1666	0.0000	-0.1666
MiniMax	0.5335	0.5181	-0.0154	0.0741	0.0435	-0.0306	0.1666	0.0000	-0.1666
Nemotron	0.5000	0.5000	+0.0000	0.0000	0.0000	+0.0000	0.0000	0.0000	+0.0000
Qwen3	0.5284	0.5485	+0.0201	0.0384	0.0689	+0.0305	0.0000	0.1666	+0.1666

Figura 23: Tabla de análisis de las métricas de L3

La UAR promedio elevada de Gemma4 (≈0,64) refleja su mayor susceptibilidad en ambos escenarios. Nemotron mantiene resistencia total (UAR=0). El hardening elimina CTER y KCCR en Gemma4 y MiniMax (Δ negativos), pero los incrementa en Qwen3 (ΔCTER=+0,031; ΔKCCR=+0,167), efecto adverso que merece investigación adicional. La técnica L3_AT_02 (Inyección indirecta) es la única con registros de éxito.

Los valores de CTER=0,0 en algunas de las campañas indican que, si bien las herramientas fueron invocadas de forma no autorizada, no se registró encadenamiento cross-tool con exfiltración de datos en el sentido definido por la métrica (propagación de datos entre herramientas sin expectativa). Este resultado puede estar vinculado al diseño conservador de las herramientas de prueba (simuladas, sin estado persistente real) y al límite de max_tool_calls=5 por réplica.

Respecto al KCCR≠0,0 en todas las campañas L3, el mecanismo de cálculo debe de ser revisado debido a los valores que devuelve. Las campañas, al ser planificadas de manera aislada, sin combinación entre capas, con dimensiones a_i y b_i de la fórmula KCCR igualadas siempre cero, pues no existe un caso L1 ni L2 asociado que haya tenido éxito en la misma ejecución, deberían resultar en 0. El KCCR debe interpretarse como una métrica transversal que adquiere significado únicamente en escenarios de validación *end-to-end* con infraestructura completa de los tres niveles, conforme se discute en la sección 4.6.

4.5. Análisis comparativo A/B de hardening

El protocolo A/B descrito en la sección 4.3 permite aislar el efecto de las contramedidas de *hardening* sobre las métricas de seguridad, manteniendo constantes el corpus de prueba, los payloads, los parámetros de *scoring* y el número de réplicas. La tabla 5 consolida las variaciones absolutas y relativas en las métricas principales entre las configuraciones baseline y hardened para cada capa.

Métrica	Δ Gemma4	Δ MiniMax	Δ Nemotron	Δ Qwen3
L1 ASR	-0.0776	+0.0700	+0.0074	-0.0108
L1 FAR	+0.0000	+0.0000	+0.0000	+0.0000
L1 FRR	+0.0874	-0.0787	-0.0084	+0.0122
L1 TTC	+0.0000	+5.5000	-11.0000	+0.0000
L2 ASR-L2	+0.0000	+0.0000	+0.0000	-0.0091
L2 PSR@5	-0.0182	-0.0182	-0.0182	-0.0205
L2 TDS	-0.2108	-0.1006	-0.0621	+0.0873
L3 UAR	-0.0040	-0.0154	+0.0000	+0.0201
L3 CTER	-0.1191	-0.0306	+0.0000	+0.0305
L3 KCCR	-0.1666	-0.1666	+0.0000	+0.1666

Figura 24: Comparativa consolidada de métricas por capa y modelo.

El análisis A/B revela tres patrones diferenciados. En primer lugar, el *hardening* funciona de manera consistente con Gemma 4 31B, mejorando consistentemente la mayoría de las métricas. MiniMax mejora en L2 y L3 pero empeora en ASR L1, al aumentar la tasa con la configuración *hardened*. Este fenómeno puede explicarse por la interacción entre los delimitadores XML de los *prompts* añadidos y el patrón de atención de los modelos: los delimitadores introducen tokens estructurales adicionales que, para ciertos payloads semánticos (especialmente L1_AT_10 y L1_AT_05), generan un contexto de

confianza que podría facilitar la evasión en lugar de prevenirla. Qwen3 mejora en L2 y levemente en ASR L1, pero empeora en CTER/KCCR L3. Nemotron no muestra variación significativa en ninguna capa.

En L2, el hardening mejora consistentemente PSR@5 y TDS en todos los modelos. Esta mejora se puede deducir del *hardening,* al operar sobre etiquetas explícitas de documentos envenenados, su cobertura está acotada a los vectores de envenenamiento conocidos.

En tercer lugar, y de forma más preocupante, el *hardening* L3 no produce ningún efecto observable sobre la UAR. La configuración *hardened* no reduce la tasa de invocaciones no autorizadas, lo que indica que el problema de seguridad en la capa agéntica no reside en la salida textual del modelo sino en el proceso de toma de decisiones sobre la invocación de herramientas. Los mecanismos de moderación de salida son insuficientes para contener este vector: se necesitan controles de autorización explícitos en el nivel del motor de herramientas, externos al flujo de razonamiento del LLM.

4.6. Contraste de cuestiones

Los resultados de la validación experimental permiten responder directamente a las cuatro cuestiones formuladas en la sección 4.1.1.

1. **¿Es el framework Norn capaz de ejecutar campañas de red teaming reproducibles sobre los tres niveles de despliegue (L1, L2, L3) y generar métricas cuantitativas consistentes?**

   Sí. La ejecución de 48 campañas completadas con trazabilidad completa confirma la capacidad operativa del *framework*. La disponibilidad de dos rondas independientes por configuración permite detectar variabilidad no atribuible al modelo, aportando un grado de rigor metodológico adicional respecto a estudios de ronda única.

2. **¿Las métricas definidas (ASR, FAR/FRR, PSR@k, TDS, UAR, CTER, KCCR) capturan diferencias significativas entre configuraciones de modelos y arquitecturas defensivas?**

   Las métricas discriminan correctamente tanto entre modelos como entre capas. La combinación ASR/FRR en L1 perfila el modelo en el eje robustez-utilidad; PSR@5 y TDS en L2 detectan envenenamiento de índice y degradación residual; UAR, CTER y KCCR en L3 capturan perfiles de riesgo agéntico cualitativamente distintos.

3. **¿Cómo se sitúa la cobertura funcional de Norn respecto a herramientas de referencia del ecosistema (Garak, PyRIT, Promptfoo)?**

Norn cubre la totalidad de la superficie de ataque L1 (comparable a Garak o Promptfoo) y extiende la cobertura a L2 y L3, dimensiones que las herramientas de referencia no abordan de forma integrada. La diferencia principal reside en el diseño progresivo por capas con métricas propias para cada nivel. La limitación actual es la amplitud del catálogo de *probes* L1, inferior a la de Garak.

4. **¿Son efectivas las contramedidas de hardening implementadas por nivel (L1: validación de entrada y endurecimiento de prompts; L2: filtrado de recuperación; L3: moderación de salida y detección de canaries) para reducir las tasas de éxito de ataque medidas por el framework, y preservan la funcionalidad benigna (zero-regression)?**

   Para esta cuestión los resultados son mixtos. El *hardening* reduce el ASR de Gemma4 y elimina el TDS residual en L2 para todos los modelos, pero incrementa el ASR de MiniMax. Esto evidencia que las contramedidas actuales no son invariantes a los modelos: su efecto puede ser contraproducente en modelos con patrones de alineamiento que combinen de manera desfavorable con los guardaraíles

4.7. Contraste de hipótesis

Esta sección evalúa cada una de las cinco hipótesis formuladas en la sección 4.1.2 a la luz de los resultados obtenidos:

H1. Las métricas de capa L1 (ASR, FAR/FRR) tienen capacidad de discriminar entre modelos con distintos niveles de alineamiento de seguridad.

La hipótesis postulaba que las métricas ASR y FAR/FRR tendrían capacidad para discriminar entre modelos con distintos niveles de alineamiento. Los resultados la **confirman**: el rango de ASR observado abarca desde 0,0% (Nemotron 3 Super) hasta 28,15% (Gemma 4 31B), una diferencia estadísticamente significativa dado el tamaño muestral (135 observaciones por campaña) y los intervalos de confianza al 95% que no se solapan entre modelos extremos. La FRR también discrimina de forma complementaria: Nemotron presenta FRR=1,0 frente al 0,68 de Gemma, lo que refleja perfiles de alineamiento cualitativamente distintos.

H2. La presencia de un pipeline RAG incrementa la superficie de ataque medible, reflejándose en valores positivos de PSR@k y TDS bajo condiciones de envenenamiento controlado del índice.

La hipótesis preveía valores positivos de PSR@k y TDS bajo condiciones de envenenamiento. Los resultados la confirman

parcialmente: PSR@5=0,5 evidencia que el mecanismo de envenenamiento indexa exitosamente contenido adversarial, incrementando objetivamente la superficie de recuperación. Sin embargo, el ASR-L2 cercano a 0 indica que la capa generativa de los modelos evaluados resiste la influencia de este contenido sobre la respuesta final, lo que matiza la hipótesis en el sentido de que el incremento de superficie medible no se traduce automáticamente en incremento de riesgo efectivo en este experimento. Finalmente, el TDS residual en baseline evidencia la degradación por contenido adversarial latente incluso sin envenenamiento explícito.

H3. Los agentes con acceso a herramientas exhiben tasas de acción no autorizada (UAR) significativamente superiores a cero ante inyección indirecta de prompts en resultados de herramientas.

La hipótesis afirmaba que los agentes mostrarían tasas de acción no autorizada significativamente superiores a cero. Los resultados la confirman de manera sólida, en todos los modelos y configuraciones. Este resultado es coherente con los hallazgos de InjecAgent y con la naturaleza del patrón ReAct, donde el LLM decide de forma autónoma qué herramienta invocar y con qué parámetros, sin consultar una política de autorización externa. Los hallazgos en base a CTER emergen como vectores de riesgo en tres de cuatro modelos, desplazando el riesgo al abuso de herramientas autorizadas, más allá de intentos de activación directos.

H4. La métrica KCCR correlaciona con la combinación de vulnerabilidades parciales en L1, L2 y L3, evidenciando que la robustez sistémica es menor que la robustez de cualquier capa aislada.

La hipótesis referenciaba que KCCR correlacionaría con la combinación de vulnerabilidades por capa. Dado el diseño experimental de campañas aisladas por capa, el KCCR calculado debería ser cero en todos los casos. No obstante, en algunas campañas este valor ha sido distinto a 0, algo que se puede atribuir a un fallo de cálculo de la métrica, por lo que la hipótesis se debería de contrastar cuando dos condiciones se cumplan:

- La métrica se revise para dar 0 en campañas aisladas
- Una campaña multicapa L1,L2,L3 consiga dar resultados empíricos de valores distintos a 0 en ejecución.

Cabe destacar ,sin embargo, que la hipótesis se confirma conceptualmente: tomando los resultados por separado de todas las capas, un sistema que integrase los tres niveles sin contramedidas adicionales presentaría un riesgo de compromiso end-to-end no despreciable.

H5. La activación de las contramedidas de *hardening* por nivel reduce significativamente las métricas de éxito de ataque (ASR,

PSR@k, UAR) respecto a la configuración baseline, sin incrementar las tasas de rechazo falso (FRR) sobre consultas benignas.

La hipótesis esperaba reducciones significativas en ASR, PSR@k y UAR con el *hardening*, sin incremento de FRR. Los resultados la confirman parcialmente, debido a que el *hardening* mejora ASR en Gemma4 y Qwen3, y elimina TDS/PSR@5 en L2 para todos los modelos, mientras que el FRR se incrementa sin regresiones severas. En contraposición, el ASR en MiniMax aumenta en campañas securizadas. Otras métricas se mantienen consistentes en todas las campañas, como FAR=0 en todas las configuraciones.

4.8. Discusión de resultados

Los resultados permiten extraer conclusiones con implicaciones metodológicas y prácticas para el red teaming ofensivo de LLMs, las cuales se pueden resumir en los siguientes puntos:

4.8.1. Asimetría de robustez entre modelos

El rango de ASR *baseline* (0%–20,4%) con FAR=0 universal refleja diferencias significativas en el alineamiento entre modelos comparables. Nemotron opera en régimen ultra-conservador (ASR=0, FRR=1,0, TTC=11): rechaza sistemáticamente todas las solicitudes adversariales independientemente de la configuración. Estos hallazgos confirman la necesidad de auditar cada modelo de forma independiente.

4.8.2. Variabilidad entre rondas

La diferencia de hasta 16 puntos porcentuales en ASR entre rondas idénticas no es atribuible al no-determinismo del modelo (temperatura=0), sino a posibles actualizaciones de los pesos vía inferencia cloud o fallas de infraestructura (*downtime*). Una auditoría sobre una API remota es un registro de estado concreto, y deja ver que hay muchas capas intermedias que pueden interferir en la experimentación.

4.8.3. Resistencia ante recuperación contaminada

Los valores de las métricas ASR-L2 cercanos a 0 con PSR@5 > 0 confirman que los modelos resisten instrucciones adversariales en el contexto recuperado. El TDS residual en *baseline* demuestra que la degradación puede persistir ante contenido adversarial latente incluso con PSR@5 bajo, ampliando la superficie de riesgo L2.

4.8.4. Desplazamiento del vector de riesgo L3

Los valores de UAR y CTER no nulos ilustran cómo el perfil de riesgo agéntico depende de la definición operacional de autorización. En términos de impacto potencial, este resultado implica que cualquiera de

los modelos evaluados, cuando se despliega en modo agéntico con herramientas, tiene posibilidad de ejecutar acciones no autorizadas ante instrucciones inyectadas a través de resultados de herramientas.

4.8.5. Eficacia diferencial del *hardening*

Las contramedidas son efectivas para Gemma4 en L1 y L3, y para todos los modelos en L2 (ΔTDS y ΔPSR@5 negativos). Sin embargo, el hardening L1 resulta contraproducente para MiniMax y el hardening L3 incrementa los valores de las métricas de Qwen3. FAR=0 en todas las configuraciones confirma que las contramedidas no introducen falsos positivos.

4.8.6. Validez del framework Norn

La ejecución de 48 campañas, 924 casos, 4.470 réplicas y 4.248 decisiones con trazabilidad completa valida la hipótesis principal. Las 10 métricas especializadas por capa (4 L1 + 3 L2 + 3 L3) capturan perfiles de riesgo cualitativamente distintos por capa y permiten caracterizar el efecto diferencial de las contramedidas.

4.8.7. Implicaciones metodológicas de la sustitución de modelos

Los modelos planificados (7–8B locales) fueron reemplazados por modelos de mayor escala vía Ollama Cloud. Sus pipelines de alineamiento más sofisticados pueden explicar los ASR relativamente bajos. La inferencia remota introduce variabilidad no controlable entre rondas, consistente con la observada. La transferibilidad de los resultados a modelos de 7–8B no está garantizada. El efecto de escala sobre el ASR, el cual plantearía si modelos más pequeños resultan más o menos vulnerables es una pregunta empírica abierta que futuras iteraciones del framework deberían abordar mediante una comparativa sistemática entre rangos de tamaño, manteniendo fija la *suite* de técnicas de ataque.

5. Conclusiones

5.1. Síntesis de contribuciones

Este trabajo ha abordado un problema abierto en la intersección de la ciberseguridad ofensiva y los sistemas de inteligencia artificial: la carencia de una metodología formal, reproducible y cuantificable para auditar LLMs en sus distintos modos de despliegue. La respuesta propuesta articula tres contribuciones principales.

5.1.1. Metodología de red teaming progresivo por capas

Se ha diseñado una metodología estructurada que aborda los tres niveles de despliegue de forma incremental: L1 (modelos standalone), L2 (sistemas con RAG) y L3 (agentes con herramientas). Cada capa incorpora criterios de entrada y salida, vectores de ataque propios y métricas diferenciadas. La metodología es agnóstica al modelo y al proveedor.

5.1.2. Taxonomía estructurada de ataques

Se ha desarrollado una taxonomía propia de 16 técnicas ofensivas (L1_AT_01 a L1_AT_10, L2_AT_01 a L2_AT_03, L3_AT_01 a L3_AT_03), organizada por capas y codificada en un catálogo de probes parametrizables. Las técnicas de mayor impacto confirmadas experimentalmente son L1_AT_06 (filtración de *prompt* de sistema) y L1_AT_07 (extracción de datos de entrenamiento).

5.1.3. Framework Norn

Se ha implementado un framework de automatización de pruebas que integra la metodología y la taxonomía en una herramienta CLI funcional. La validación experimental abarca 48 campañas completadas, 924 casos de prueba, 4.470 réplicas y 4.248 decisiones de scoring, con trazabilidad completa en base de datos SQLite y generación de informes.

5.1.4. Integración con benchmarks públicos

La alineación del catálogo de *probes* con MITRE ATLAS, OWASP LLM Top 10 facilita la comparación de resultados entre estudios y aumentar la adaptabilidad de la metodología a casos de uso reales.

5.2. Cumplimiento de objetivos

Los siete objetivos intermedios formulados en la sección 1.2 se han cumplido en los siguientes términos.

TFM-01
Cumplido. El capítulo 2 revisa el estado del arte en seguridad ofensiva de LLMs, incluyendo inyección de prompts, envenenamiento de RAG y ataques agénticos, estableciendo la base conceptual de la taxonomía.

TFM-02
Cumplido. Se han definido 16 técnicas ofensivas en tres capas, con identificadores, descripciones y payloads de referencia, implementados como probes parametrizables en Norn.

TFM-03

Cumplido. La metodología progresiva por capas con fases, criterios de entrada/salida y protocolo A/B se ha documentado y validado experimentalmente en dos rondas de ejecución independientes.

TFM-04
Cumplido. Norn implementa la metodología en una herramienta CLI funcional con soporte para los tres niveles, múltiples modos de scoring y exportación multi-formato.

TFM-05
Cumplido. Se han definido y calculado ocho métricas especializadas con especificaciones formales y calculadoras automáticas integradas en el pipeline de exportación.

TFM-06
Cumplido. Se han ejecutado 48 campañas sobre cuatro modelos en las tres capas, con dos rondas de ejecución por configuración, obteniendo métricas que contrastan todas las hipótesis planteadas.

TFM-07
Parcialmente cumplido. Se ha realizado una comparativa cualitativa de cobertura funcional frente a Garak, PyRIT y Promptfoo, identificando el valor diferencial de Norn (L2 y L3) y las áreas de mejora (amplitud del catálogo L1).

5.3. Limitaciones del trabajo

Las siguientes limitaciones son inherentes al diseño metodológico y no derivan de las condiciones de ejecución ya discutidas en las secciones anteriores.

5.3.1. LLM Judge provisional

Debido al alcance limitado del trabajo, el LLM judge está programado como modo de *scoring*, pero no ha sido validado contra un conjunto de referencia, por lo que se ha dejado a modo de placeholder redundante del *scorer* heurístico. Esto implica que los valores de ASR reportados están condicionados por ser únicamente heurísticos, aunque a nivel conceptual quiera ser híbrido.

5.3.2. Corpus estático vs adversarios adaptativos

Todos los casos de prueba utilizan *payloads* predefinidos. Un adversario real es iterativo: observa la respuesta del sistema, detecta qué formatos o contextos activan sus defensas, y adapta el ataque en consecuencia. Técnicas como PAIR[20] (Prompt Automatic Iterative Refinement) o AutoDAN[21] operan exactamente sobre este principio. Esta diferencia de capacidades deja entrever que una de las debilidades del *framework* es, precisamente, la cantidad y calidad de payloads que es capaz de ejecutar como parte de sus campañas.

5.3.3. Ausencia de corpus benigno de referencia para FRR

De manera sistemática en toda la experimentación, el FRR ha resultado en 0. Si bien este dato puede ir vinculado a un alineamiento correcto de los modelos, cabe la posibilidad que el corpus no disponga de la suficiente variabilidad de modalidades como para influir en los modelos. Otro razonamiento puede llevar a pensar que el *corpus* se generó sobre un conjunto de consultas benignas diseñado para el experimento, no extraído de tráfico real de usuarios. Sin un *corpus* de referencia representativo del uso real, los valores de FRR son relativos a la *suite* de prueba, no al comportamiento del sistema en producción.

5.3.4. Validez externa limitada

Todos los experimentos se ejecutan sobre una única aplicación web construida ad-hoc. Los resultados no han sido replicados sobre sistemas con distintos prompts de sistema, distintas configuraciones de RAG, diferentes topologías de herramientas o diferentes dominios de aplicación. La generalización de los valores de ASR, PSR@5 o UAR a otros despliegues requiere validación explícita, ya que el prompt de sistema y la arquitectura del *target* son factores de confusión no controlados en el diseño actual.

5.4. Líneas de trabajo futuro

Los resultados y limitaciones identificados sugieren las siguientes líneas de extensión:

5.4.1. Estudio de estabilidad temporal sobre APIs cloud

La variabilidad entre rondas observada motiva un estudio sistemático de cómo evolucionan las métricas de seguridad de un modelo a lo largo del tiempo, ejecutando campañas idénticas en intervalos regulares para detectar regresiones o mejoras silenciosas en el alineamiento.

5.4.2. Validación end-to-end con KCCR

El diseño de un entorno integrado L1+L2+L3 que permita calcular KCCR en escenarios de kill-chain completa es la extensión más directa del trabajo. Requiere un entorno que simule una aplicación con las tres capas activas simultáneamente, con payloads diseñados para explotar vulnerabilidades en cascada.

5.4.3. Mejoras de calidad de Norn

El ritmo del desarrollo del *framework* ha conllevado que se acumule deuda técnica y de características. Una línea válida para el proyecto es seguir creciendo en capacidades, como ejecución paralela de payloads, aumento del *corpus* de ataques por técnica, integraciones con herramientas más establecidas como Garak, para poder aplicar la capa

de métricas a los recursos que ya dispone la otra herramienta… La herramienta está preparada para crecer en muchas dimensiones.

5.4.4. Estudio del efecto de escala en modelos self-hosted

Una comparativa entre modelos de 7–8B, 13–30B y 70B+ sobre la misma suite de técnicas permitiría cuantificar cómo la capacidad paramétrica y la inversión en alineamiento afectan el ASR y FRR, respondiendo si los modelos más grandes son sistemáticamente más robustos.

5.4.5. Defensas de capa de ejecución en L3

La emergencia de las métricas CTER y KCCR con herramientas autorizadas apunta a la necesidad de evaluar defensas en la capa de ejecución de herramientas: políticas RBAC, intercepción de *tool calls*, *sandboxing* de resultados. Una próxima versión del *framework* debería incorporar entornos de prueba para este tipo de contramedidas.

5.5. Reflexión final

Este trabajo ha partido de la constatación de que el campo de la seguridad de sistemas LLM está madurando a una velocidad que supera la capacidad de las metodologías de auditoría tradicionales para adaptarse. Los resultados obtenidos aportan evidencia empírica sobre varios hallazgos que merecen ser destacados por sus implicaciones tanto metodológicas como prácticas.

El primero es la heterogeneidad de la robustez entre modelos. Una diferencia de 20 puntos en ASR *baseline* entre Gemma4 y Nemotron, con perfiles cualitativamente distintos en el eje robustez-utilidad, confirma que el alineamiento de seguridad no es una propiedad transferible entre arquitecturas. Cualquier decisión de despliegue que asuma equivalencia de seguridad entre modelos sin validación explícita está introduciendo un riesgo no cuantificado.

El segundo hallazgo es la variabilidad temporal sobre APIs cloud, el cual se pone en evidencia gracias a las grandes diferencias de ASR entre rondas idénticas con temperatura=0 no es atribuible al no-determinismo del modelo, sino a modificaciones silenciosas en los pesos o cambios de infraestructura del proveedor. Este resultado evidencia que una auditoría puntual sobre una API comercial tiene una caducidad no determinable y que la vigilancia continua de la seguridad debe formar parte del ciclo de vida operativo del sistema.

El tercer hallazgo concierne a la eficacia diferencial y a veces contraproducente del hardening. Mientras que las contramedidas mejoran consistentemente las métricas de L2 para todos los modelos, en L1 el ASR de MiniMax se incrementa bajo *hardening*, y en L3 las métricas de Qwen3 empeoran. Este resultado demuestra que las defensas no son invariantes al modelo y que la práctica de aplicar guardarraíles genéricos sin validación empírica por modelo puede, en ciertos casos, degradar la seguridad en lugar de reforzarla. La

auditoría no debe limitarse a medir el ASR *baseline*; debe medir también el efecto de las propias defensas.

El cuarto hallazgo es el desplazamiento del vector de riesgo hacia la capa agéntica. En la experimentación se ha percibido una UAR promedio de 0,64 en Gemma4, lo que implica que aproximadamente dos de cada tres interacciones adversariales resultan en una acción no autorizada. Dado que el *hardening* L3 no reduce la UAR de forma significativa, el problema de seguridad en agentes no reside en la salida textual del modelo, sino en el proceso autónomo de decisión sobre invocación de herramientas. Este hallazgo tiene consecuencias directas para la arquitectura de sistemas agénticos seguros: se requieren controles de autorización explícitos en el motor de ejecución de herramientas, externos al flujo de razonamiento del LLM, y no meros filtros de salida.

En conjunto, estos resultados validan la premisa central del trabajo: la necesidad de una metodología de auditoría de seguridad que opere por capas, con métricas propias para cada una, ya que el perfil de riesgo es cualitativamente distinto en modelos standalone, sistemas RAG y agentes con herramientas. La trazabilidad completa de las 48 campañas, 924 casos y 4.470 réplicas ejecutadas demuestra que es viable auditar estos sistemas con el mismo rigor que cualquier otro componente de la infraestructura de TI.

Norn, en su estado actual, demuestra la viabilidad de automatizar la auditoría de seguridad de sistemas LLM con una metodología reproducible y métricas cuantitativas por capa. La seguridad de los sistemas de IA no es una propiedad emergente del alineamiento: es una propiedad de ingeniería que debe diseñarse, medirse y verificarse con la misma rigurosidad que cualquier otro control de seguridad. Este trabajo pretende dar un paso en esa dirección, con la aspiración de que la metodología propuesta y la herramienta desarrollada contribuyan a cerrar la brecha entre la velocidad de despliegue de los sistemas de IA y la madurez de las prácticas de auditoría que los protegen.

Bibliografía

- [1] OWASP. (2025). OWASP Top 10 for Large Language Model Applications (v2025). https://owasp.org/www-project-top-10-for-large-language-model-applications/
- [2] MITRE. (2024). MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems. https://atlas.mitre.org/
- [3] NIST. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). National Institute of Standards and Technology. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf
- [4] Derczynski, L., Giadikiaroglou, P., & et al. (2024). Garak: A framework for large language model red teaming. arXiv preprint. https://arxiv.org/abs/2406.11036
- [5] Munoz, G. D. L., Minnich, A. J., Lutz, R., Lundeen, R., Dheekonda, R. S. R., Chikanov, N., Jagdagdorj, B.-E., Pouliot, M., Chawla, S., Maxwell, W., Bullwinkel, B., Pratt, K., de Gruyter, J., Siska, C., Bryan, P., Westerhoff, T., Kawaguchi, C., Seifert, C., Kumar, R. S. S., & Zunger, Y. (2024). PyRIT: A framework for security risk identification and red teaming in generative AI systems. arXiv. https://arxiv.org/abs/2410.02828
- [6] Promptfoo. (2024). Promptfoo: Test and evaluate LLMs. GitHub. https://github.com/promptfoo/promptfoo
- [7] Perez, F., & Ribeiro, I. (2022). Ignore previous prompt: Attack techniques for language models. arXiv. https://doi.org/10.48550/arXiv.2211.09527
- [8] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. arXiv. https://arxiv.org/abs/2302.12173
- [9] Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., & Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. arXiv. https://arxiv.org/abs/2307.15043
- [10] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., Oprea, A., & Raffel, C. (2021). Extracting training data from large language models. arXiv. https://arxiv.org/abs/2012.07805
- [11] Zou, W., Geng, R., Wang, B., & Jia, J. (2024). PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. arXiv. https://arxiv.org/abs/2402.07867
- [12] Zhan, Q., Liang, Z., Ying, Z., & Kang, D. (2024). InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. arXiv. https://arxiv.org/abs/2403.02691
- [13] ENISA. (2024). ENISA Threat Landscape 2024. European Union Agency for Cybersecurity. https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024
- [14] Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S., Hawkins, W., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L. A., Isaac, W., Legassick, S., Irving, G., & Gabriel, I. (2021). Ethical and social risks of harm from language models. arXiv. https://arxiv.org/abs/2112.04359
- [15] Derner, E., & Batistič, K. (2023). Beyond the safeguards: Exploring the security risks of ChatGPT. arXiv. https://arxiv.org/abs/2305.08005

- [16] Purpura, A., Wadhwa, S., Zymet, J., Gupta, A., Luo, A., Rad, M. K., Shinde, S., & Sorower, M. S. (2025). Building safe GenAI applications: An end-to-end overview of red teaming for large language models. In T. Cao, A. Das, T. Kumarage, Y. Wan, S. Krishna, N. Mehrabi, J. Dhamala, A. Ramakrishna, A. Galystan, A. Kumar, R. Gupta, & K.-W. Chang (Eds.), *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)* (pp. 335–350). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.trustnlp-main.23

- [17] Verma, A., Krishna, S., Gehrmann, S., Seshadri, M., Pradhan, A., Ault, T., Barrett, L., Rabinowitz, D., Doucette, J., & Phan, N. (2025). *Operationalizing a threat model for red-teaming large language models (LLMs)*. arXiv. https://arxiv.org/abs/2407.14937

- [17] Feffer, M., Sinha, A., Deng, W. H., Lipton, Z. C., & Heidari, H. (2024). *Red-teaming for generative AI: Silver bullet or security theater?* arXiv. https://arxiv.org/abs/2401.15897

- [18] Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, & D. Lin (Eds.), *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://aclanthology.org/P02-1040/

- [19] Lin, C.-Y. (2004). *ROUGE: A Package for Automatic Evaluation of Summaries. En Text Summarization Branches Out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. https://aclanthology.org/W04-1013/

- [20] Chao, P., Robey, A., Dobriban, E., Hassani, H., Pappas, G. J., & Wong, E. (2024). *Jailbreaking black box large language models in twenty queries.* arXiv. https://doi.org/10.48550/arXiv.2310.08419

- [21] Zhu, S., Zhang, R., An, B., Wu, G., Barrow, J., Wang, Z., Huang, F., Nenkova, A., & Sun, T. (2023). *AutoDAN: Interpretable gradient-based adversarial attacks on large language models*. arXiv. https://doi.org/10.48550/arXiv.2310.15140