



**HoneyAgent: Framework configurable de despliegue de  
honeypots distribuidos con análisis de amenazas basado en  
agentes LLM**

**Profesora: Samela, Marcela**  
**Alumno: Romero Mojsa, Gonzalo**

**Trabajo Final de Carrera presentado para obtener el título de Licenciado en Gestión de  
Tecnología Informática  
Julio, 2026**

## Resumen

El presente Trabajo Final de Carrera tiene como objetivo el diseño e implementación de HoneyAgent, un framework configurable para el despliegue de honeypots distribuidos en infraestructuras de Amazon Web Services (AWS), orientado al sector fintech bajo el marco regulatorio del estándar PCI-DSS y las normativas del Banco Central de la República Argentina (BCRA).

La problemática que motiva este trabajo se centra en la creciente exposición de las organizaciones financieras a ataques dirigidos a credenciales e identidades en entornos cloud, combinada con la limitación de los mecanismos de detección tradicionales, que operan de manera reactiva y no ofrecen trazabilidad sobre el proceso de decisión ante cada alerta. En un sector donde la regulación exige justificar las acciones de seguridad, esta brecha resulta especialmente crítica.

Para abordar esta problemática, se propone un enfoque basado en deception technology, en el que un archivo de configuración YAML define el conjunto de honeypots a desplegar: usuarios IAM ficticios, buckets S3 señuelo, endpoints RDS inactivos y secretos falsos en AWS Secrets Manager. La infraestructura se provisiona mediante Terraform o AWS CDK, y la detección de eventos se lleva a cabo a través de un pipeline compuesto por AWS CloudTrail, Amazon EventBridge, AWS Lambda y Amazon GuardDuty. Ante cada activación, un agente basado en modelos de lenguaje de gran escala (LLM), implementado sobre el modelo Claude Sonnet 4.5 a través de Amazon Bedrock con capacidad de uso de herramientas, analiza el evento en contexto, consulta fuentes externas y genera un reporte estructurado que documenta el razonamiento detrás de cada decisión.

En los capítulos siguientes se explicará el marco teórico que sustenta la propuesta, se describe en detalle el diseño e implementación del framework y se presentarán los resultados obtenidos en términos de viabilidad técnica, extensibilidad y alineación regulatoria.

*Palabras clave:* honeypots, agente LLM, AWS, PCI-DSS, canary tokens, deception technology, explainability, fintech.

## Abstract

This Final Degree Project presents the design and implementation of HoneyAgent, a configurable framework for deploying distributed honeypots on Amazon Web Services (AWS) infrastructure, aimed at fintech organizations operating under the PCI-DSS standard and the regulations established by the Banco Central de la República Argentina (BCRA).

The problem that motivates this work stems from the growing exposure of financial organizations to attacks targeting cloud identities and credentials, combined with the limitations of traditional detection mechanisms, which operate reactively and lack traceability over the decision-making process behind each alert. In a sector where regulations require organizations to justify their security actions, this gap becomes particularly critical.

To address this, a deception technology approach is proposed. A YAML configuration file defines the set of honeypots to be deployed: fake IAM users, decoy S3 buckets, inactive RDS endpoints, and fictitious secrets stored in AWS Secrets Manager. Infrastructure is provisioned through Terraform or AWS CDK, and event detection is handled by a pipeline built on AWS CloudTrail, Amazon EventBridge, AWS Lambda, and Amazon GuardDuty. When a honeypot is triggered, a large language model (LLM) agent, implemented on top of the Claude Sonnet 4.5 model through Amazon Bedrock with tool use capabilities, analyzes the event in context, queries external sources, and generates a structured report that documents the reasoning behind each decision.

The following chapters cover the theoretical framework supporting the proposal, a detailed description of the framework design and implementation, and the results obtained in terms of technical feasibility, extensibility, and regulatory alignment.

*Keywords:* honeypots, LLM agent, AWS, PCI-DSS, canary tokens, deception technology, explainability, fintech.

## **Dedicatoria**

Dedico este trabajo a mi familia y amigos, por su apoyo incondicional y por ser mi fuente de motivación constante, y a todas las personas que contribuyeron de alguna manera a la realización de este proyecto

## **Reconocimientos**

Agradezco a la Universidad Abierta Interamericana y al cuerpo docente de la Licenciatura en Gestión de Tecnología Informática por la formación recibida a lo largo de la carrera, que sentó las bases necesarias para el desarrollo de este trabajo. Agradezco también a mi familia, cuyo acompañamiento constante fue un sostén fundamental durante todo el proceso. Este trabajo es, en buena medida, resultado de ese acompañamiento.

# Índice General

|                                                                   |           |
|-------------------------------------------------------------------|-----------|
| <b>Resumen</b>                                                    | <b>2</b>  |
| <b>Abstract</b>                                                   | <b>4</b>  |
| <b>Dedicatoria</b>                                                | <b>5</b>  |
| <b>Reconocimientos</b>                                            | <b>6</b>  |
| <b>Índice General</b>                                             | <b>7</b>  |
| <b>Índice de Figuras</b>                                          | <b>10</b> |
| <b>Índice de Tablas</b>                                           | <b>11</b> |
| <b>1 Introducción</b>                                             | <b>12</b> |
| 1.1 Descripción del problema                                      | 12        |
| 1.2 Justificación de la propuesta                                 | 13        |
| 1.3 Marco institucional                                           | 14        |
| 1.4 Objetivos del Trabajo Final                                   | 16        |
| 1.4.1 Objetivo general                                            | 16        |
| 1.4.2 Objetivos específicos                                       | 16        |
| 1.5 Contribuciones principales                                    | 16        |
| <b>2 Estado del Arte</b>                                          | <b>18</b> |
| 2.1 La evolución de los honeypots hacia entornos cloud            | 18        |
| 2.2 Honeytokens de identidad y credenciales                       | 21        |
| 2.3 La convergencia honeypots + LLM: del alerta al razonamiento   | 23        |
| 2.4 Análisis del gap y posicionamiento de la propuesta            | 25        |
| <b>3 Marco Teórico</b>                                            | <b>27</b> |
| 3.1 Honeypots y Deception Technology                              | 27        |
| 3.1.1 Introducción                                                | 27        |
| 3.1.2 Resumen                                                     | 27        |
| 3.1.3 ¿Qué es un honeypot?                                        | 27        |
| 3.1.4 Clasificación de honeypots                                  | 28        |
| 3.1.5 Deception Technology en entornos cloud                      | 29        |
| 3.2 Canary Tokens e Identidad en la Nube                          | 30        |
| 3.2.1 Introducción                                                | 30        |
| 3.2.2 Resumen                                                     | 31        |
| 3.2.3 ¿Qué son los canary tokens?                                 | 31        |
| 3.2.4 Gestión de identidad en AWS: IAM, Secrets Manager, S3 y RDS | 32        |
| 3.2.5 Honeypots de identidad y credenciales                       | 34        |
| 3.3 Detección de amenazas en AWS                                  | 35        |
| 3.3.1 Introducción                                                | 35        |
| 3.3.2 Resumen                                                     | 36        |
| 3.3.3 AWS CloudTrail                                              | 36        |
| 3.3.4 Amazon EventBridge                                          | 37        |
| 3.3.5 AWS Lambda en pipelines de detección                        | 38        |

|                                                                         |           |
|-------------------------------------------------------------------------|-----------|
| 3.3.6 Amazon GuardDuty                                                  | 39        |
| 3.4 Agentes LLM y razonamiento con herramientas                         | 40        |
| 3.4.1 Introducción                                                      | 40        |
| 3.4.2 Resumen                                                           | 40        |
| 3.4.3 ¿Qué es un agente LLM?                                            | 41        |
| 3.4.4 Tool Use: razonamiento con herramientas externas                  | 42        |
| 3.4.5 Explainability en sistemas de IA para ciberseguridad              | 43        |
| 3.5 Marco regulatorio: PCI-DSS aplicado a fintechs latinoamericanas     | 45        |
| 3.5.1 Introducción                                                      | 45        |
| 3.5.2 Resumen                                                           | 45        |
| 3.5.3 Estándar PCI-DSS: alcance y requerimientos relevantes             | 46        |
| 3.5.4 Contexto regulatorio en Argentina: BCRA y fintechs                | 47        |
| 3.5.5 Honeypots como control complementario bajo PCI-DSS                | 48        |
| <b>4 Propuesta de Intervención</b>                                      | <b>50</b> |
| 4.1 Descripción del framework y arquitectura general                    | 50        |
| 4.2 Capa de configuración (honeypots.yaml + Terraform/CDK)              | 52        |
| 4.3 Honeypots en AWS: diseño e implementación                           | 54        |
| 4.4 Pipeline de detección (CloudTrail, EventBridge, Lambda y GuardDuty) | 55        |
| 4.5 Agente LLM: diseño, herramientas y explainability                   | 57        |
| 4.6 Outputs del sistema                                                 | 59        |
| 4.7 Recursos utilizados                                                 | 60        |
| 4.8 Cronograma de tareas                                                | 61        |
| 4.8.1 Diagrama de Gantt                                                 | 62        |
| 4.9 Evaluación del proyecto                                             | 64        |
| 4.9.1 Matriz FODA de la propuesta de intervención                       | 64        |
| 4.9.2 Análisis de costos                                                | 65        |
| <b>5 Factores externos condicionantes</b>                               | <b>67</b> |
| <b>Conclusiones</b>                                                     | <b>69</b> |
| <b>Trabajos futuros</b>                                                 | <b>71</b> |
| <b>Anexo I</b>                                                          | <b>73</b> |
| <b>Anexo II</b>                                                         | <b>77</b> |
| <b>Acrónimos</b>                                                        | <b>86</b> |
| <b>Referencias</b>                                                      | <b>88</b> |



## Índice de Figuras

Figura 4.1. *Arquitectura general del framework HoneyAgent*

Figura 4.2. *Diagrama de Gantt del cronograma de tareas*

## Índice de Tablas

Tabla 1. *Comparación de frameworks de honeypots en entornos cloud*

Tabla 2. *Integración de LLMs en honeypots y pipelines de análisis de amenazas*

Tabla 3. *Análisis FODA de la propuesta de intervención*

Tabla 4. *Matriz de trazabilidad entre requisitos regulatorios y componentes de HoneyAgent*

# 1 Introducción

## 1.1 Descripción del problema

En los últimos años, el sector financiero ha experimentado una transformación profunda hacia la adopción de infraestructuras cloud. Este proceso, lejos de simplificar la superficie de ataque, la ha expandido de forma considerable. Verizon Business (2025, p. 10) señala que el abuso de credenciales continúa siendo el vector de ataque más frecuente en brechas de seguridad a nivel global, y que el 54% de las víctimas de ransomware registradas tenían sus dominios presentes en volcados de credenciales comprometidas (Verizon Business, 2025, p. 12). Por su parte, IBM Security (2025) reporta que el abuso de identidades fue el vector de entrada preferido por los atacantes en el 30% de los casos registrados en 2024, en un contexto de mercado creciente de credenciales robadas: la oferta de credenciales de info stealers a la venta en la dark web aumentó un 12% respecto del año anterior. Esta tendencia afecta de manera directa a las organizaciones que operan en entornos cloud, donde las identidades digitales son, en la práctica, el perímetro de seguridad.

En el caso específico de las fintechs argentinas, el problema adquiere una dimensión adicional. El Banco Central de la República Argentina (2026, p. 1), a través de la Comunicación "A" 8398, establece que los proveedores de servicios de pago y entidades financieras quedan sujetos a requisitos mínimos de gestión de riesgos tecnológicos y ciberseguridad. A nivel internacional, el estándar PCI DSS v4.0.1 (PCI Security Standards Council, 2024, p. 9) exige la existencia de un plan de respuesta ante incidentes aplicable a todas las entidades del ecosistema, junto con la documentación y trazabilidad de cada decisión de seguridad tomada durante un incidente. Cumplir con estas obligaciones demanda no solo detectar las amenazas, sino poder justificar de forma auditada cómo se llegó a cada conclusión.

Sin embargo, los mecanismos de detección tradicionales presentan limitaciones estructurales frente a este escenario. Izhikevich et al. (2023, p. 313) demostraron, a través de un estudio con sensores distribuidos en cinco proveedores de nube y veintitrés países, que los atacantes realizan actividades de reconocimiento sobre infraestructuras cloud de manera sistemática y automatizada, en plazos que superan ampliamente la capacidad de respuesta de los equipos de seguridad que dependen de alertas reactivas. La OWASP Foundation (2025, A09) identifica las fallas en registro y monitoreo como uno de los diez riesgos más críticos en aplicaciones, señalando que la ausencia de visibilidad en tiempo real sobre eventos anómalos es una brecha recurrente en organizaciones de todos los tamaños.

Esta combinación de factores define el problema que motiva el presente trabajo: el crecimiento sostenido de los ataques sobre identidades cloud, las obligaciones regulatorias que exigen trazabilidad y las limitaciones estructurales de los mecanismos de detección existentes.

## **1.2 Justificación de la propuesta**

La propuesta de intervención no busca reemplazar los controles de seguridad existentes, sino complementarlos mediante un enfoque proactivo basado en deception technology. Tal como señalan Moric et al. (2025, pp. 1-2), los honeypots constituyen una técnica de detección que invierte la lógica de los mecanismos tradicionales: en lugar de analizar tráfico legítimo en busca de anomalías, generan activos señuelo que, por definición, no deberían recibir ningún tipo de interacción. Cualquier acceso a estos activos representan, con alto grado de certeza, una actividad maliciosa o no autorizada, lo que los convierte en una herramienta con una tasa de falsos positivos considerablemente menor a la de los sistemas basados en firmas o umbrales de comportamiento.

En el contexto cloud, esta lógica se vuelve especialmente valiosa. Devi Priya y Chakkaravarthy (2023, p. 1) demuestran que los honeypots desplegados en infraestructura

cloud permiten detectar actividad de reconocimiento y movimiento lateral en etapas tempranas del ciclo de ataque, antes de que el atacante alcance activos de mayor valor. HoneyAgent extiende este concepto hacia las identidades y credenciales de AWS, desplegando señuelos en los servicios más frecuentemente abusados según la evidencia empírica disponible: IAM, S3, Secrets Manager y RDS.

Sin embargo, la detección por sí sola no es suficiente en entornos regulados. Tanto el estándar PCI DSS v4.0.1 (PCI Security Standards Council, 2024, p. 331) como la Comunicación "A" 8398 del BCRA (Banco Central de la República Argentina, 2026, p. 1) exigen que las decisiones de seguridad sean documentadas y trazables. Bridges et al. (2025, preprint de arXiv aún no sometido a revisión por pares, Secc. 3) identifican precisamente esta brecha en los sistemas de honeypots tradicionales: generan alertas, pero no razonamiento. La incorporación de un agente LLM con capacidad de uso de herramientas permite cerrar ese espacio, dado que el agente no solo clasifica el evento sino que documenta paso a paso el proceso de análisis que lo llevó a su conclusión, produciendo un reporte auditable en cada activación.

Por último, la decisión de implementar el framework de forma declarativa mediante un archivo YAML responde a una necesidad práctica. Yu et al. (2024, p. 2) destacan que las arquitecturas de honeynet existentes presentan limitaciones en escala y profundidad, y que la complejidad operativa es uno de los principales obstáculos para su adopción. Un esquema de configuración simple y extensible permite que equipos de seguridad con recursos acotados puedan desplegar y mantener honeypots sin depender de conocimiento especializado en cada servicio de AWS involucrado.

### **1.3 Marco institucional**

Las organizaciones del sector fintech en Argentina operan bajo un esquema regulatorio dual que combina normativa local e internacional. En el plano local, el Banco

Central de la República Argentina (2026, p. 1), mediante la Comunicación "A" 8398, incorporó formalmente a los Proveedores de Servicios de Pago (PSP) inscriptos en su registro como sujetos obligados al cumplimiento de los Requisitos Mínimos para la Gestión y Control de los Riesgos de Tecnología y Seguridad de la Información, otorgándoles un plazo de 180 días corridos desde la publicación de la norma para implementar las adecuaciones correspondientes. Esta incorporación equipara, en materia de ciberseguridad, a los PSP con las entidades financieras tradicionales, extendiendo a las fintechs obligaciones que hasta entonces les resultaban ajenas.

En el plano internacional, las organizaciones que procesan, transmiten o almacenan datos de tarjetas de pago quedan alcanzadas por el estándar PCI DSS v4.0.1. De acuerdo con el PCI Security Standards Council (2024, p. 8), el estándar aplica a todas las entidades que participan en el procesamiento de datos de cuentas, incluyendo comercios, procesadores, adquirentes, emisores y otros proveedores de servicios (PCI Security Standards Council, 2024, p. 8). Sus controles abarcan desde la segmentación de redes y el control de acceso hasta la gestión de incidentes y la trazabilidad de las decisiones de seguridad. En el contexto argentino, las fintechs que operan como emisores de tarjetas, procesadores de pago o facilitadores de cobros se encuentran, por tanto, sujetas de manera simultánea a ambos marcos.

La convergencia entre la Comunicación "A" 8398 del BCRA y PCI DSS define el espacio institucional en el que se enmarca HoneyAgent. Ambos marcos comparten el requisito de documentar y auditar las decisiones de seguridad tomadas ante incidentes: el Banco Central de la República Argentina (2026, p. 6) exige procedimientos coordinados para la gestión de ciberincidentes, mientras que PCI DSS (PCI Security Standards Council, 2024, p. 331) requiere la existencia de un plan de respuesta activable ante cualquier evento sospechoso o confirmado que pueda comprometer el entorno de datos de los titulares de

tarjetas. El framework propuesto en este trabajo busca cubrir precisamente esa brecha: generar no sólo alertas de detección, sino registros auditables del razonamiento que llevó a cada decisión de seguridad.

## **1.4 Objetivos del Trabajo Final**

### **1.4.1 Objetivo general**

Diseñar e implementar HoneyAgent, un framework de honeypots distribuidos en AWS con un agente de inteligencia artificial, para mejorar la detección y el análisis de ataques sobre identidades y credenciales en organizaciones fintech.

### **1.4.2 Objetivos específicos**

1. Diseñar la arquitectura de honeypots de identidad en AWS (IAM, S3, Secrets Manager, RDS), configurable de forma declarativa mediante un archivo YAML.
2. Implementar el pipeline de detección basado en AWS CloudTrail, Amazon EventBridge y AWS Lambda para la captura y el enrutamiento de eventos generados por los señuelos.
3. Desarrollar un agente LLM con capacidad de uso de herramientas que analice eventos de seguridad y genere reportes con razonamiento explícito y trazable.
4. Validar la capacidad de detección del framework frente a escenarios de ataque simulados sobre los señuelos de identidad.
5. Evaluar la alineación de los reportes generados con los requisitos de documentación y trazabilidad de PCI DSS v4.0.1 y la Comunicación "A" 8398 del BCRA.

## **1.5 Contribuciones principales**

El presente trabajo realiza las siguientes contribuciones:

Framework configurable de honeypots orientado a identidades AWS. HoneyAgent es el primer framework de código abierto que centraliza el despliegue de señuelos sobre los cuatro servicios de AWS más frecuentemente abusados en ataques de identidad: IAM, S3,

Secrets Manager y RDS. Su configuración declarativa en YAML permite definir, modificar y ampliar el conjunto de señuelos sin necesidad de modificar el código subyacente, reduciendo la barrera operativa para equipos con recursos acotados.

Pipeline de detección nativo en AWS sin agentes adicionales. El sistema de detección se apoya exclusivamente en servicios administrados de AWS: CloudTrail como fuente de auditoría, EventBridge como motor de enrutamiento y Lambda como ejecutor del agente. Esta decisión arquitectónica elimina la necesidad de instalar y mantener agentes de terceros, reduciendo la superficie de ataque y el costo operativo del framework.

Agente LLM con razonamiento explícito y trazable. El componente central del framework es un agente basado en el modelo Claude Sonnet 4.5, invocado a través de Amazon Bedrock con capacidad de uso de herramientas, capaz de correlacionar eventos de CloudTrail, consultar reputación de IPs y generar un reporte estructurado que documenta cada paso del proceso de análisis. A diferencia de los sistemas de honeypots tradicionales que producen únicamente alertas, HoneyAgent produce razonamiento auditable, lo que permite a los equipos de seguridad reconstruir la cadena de decisiones ante una revisión regulatoria.

Alineación demostrable con PCI DSS v4.0.1 y la Comunicación "A" 8398 del BCRA. El framework es evaluado explícitamente contra los requisitos de documentación y trazabilidad de ambos marcos regulatorios, aportando evidencia concreta de que un sistema de honeypots con análisis LLM puede contribuir al cumplimiento normativo en el sector fintech argentino.

## 2 Estado del Arte

La ciberseguridad ha experimentado una transformación profunda en la última década, impulsada por la migración masiva de servicios e infraestructuras hacia entornos cloud, la proliferación de amenazas basadas en identidad y credenciales, y la irrupción de la inteligencia artificial como herramienta tanto ofensiva como defensiva. Dentro de este contexto, los honeypots, sistemas señuelo diseñados para atraer y analizar actividad maliciosa, han recorrido un largo camino desde sus primeras implementaciones en redes físicas hasta convertirse en componentes nativos de plataformas cloud. Al mismo tiempo, la aparición de los grandes modelos de lenguaje (Large Language Models, LLMs) ha abierto una nueva dimensión para el análisis y la respuesta automatizada a amenazas. Este capítulo revisa la literatura existente en torno a tres ejes temáticos convergentes: la evolución de los honeypots hacia entornos cloud, la emergencia de los honeytokens de identidad como vector de detección, y la integración de agentes LLM en pipelines de análisis de amenazas. El objetivo es identificar el estado actual del conocimiento, los avances más significativos y la brecha que motiva la propuesta de HoneyAgent.

### 2.1 La evolución de los honeypots hacia entornos cloud

Los honeypots tienen como función esencial atraer y registrar la actividad de actores maliciosos mediante la simulación de sistemas o servicios vulnerables. Morić et al. (2025, pp. 1-2) definen un honeypot como un sistema señuelo o aplicación diseñada para atraer y examinar actividades maliciosas, que funciona como trampa para atacantes al tiempo que proporciona inteligencia valiosa sobre amenazas emergentes. En términos prácticos, un honeypot permite a los equipos de seguridad identificar vulnerabilidades del sistema, analizar el comportamiento de los adversarios y fortalecer las defensas de red contra ataques como el phishing, los ataques de denegación de servicio distribuido (DDoS, por sus siglas en inglés) y otras intrusiones (Morić et al., 2025, p. 2).

La literatura clasifica los honeypots según su nivel de interacción. Los honeypots de baja interacción (Low-Interaction Honeypots, LIHs) emulan servicios y protocolos de forma limitada, lo que los hace más seguros y fáciles de desplegar, aunque con menor capacidad para capturar comportamientos complejos del atacante. Los honeypots de alta interacción (High-Interaction Honeypots, HIHs) replican sistemas completos y ofrecen un alcance de observación más amplio, pero a costa de un riesgo operacional y una carga de gestión significativamente mayores (Morić et al., 2025, p. 8). Esta dicotomía ha definido el trade-off central en el diseño de honeypots durante décadas: a mayor fidelidad, mayor riesgo y complejidad operativa.

Con la consolidación del cloud computing como modelo predominante de infraestructura empresarial, los honeypots debieron adaptarse a entornos dinámicos, multi-tenant y distribuidos globalmente. Devi Priya y Chakkaravarthy (2023, p. 1) señalan que los honeypots en entornos cloud ofrecen mayor visibilidad que los sistemas de detección de intrusiones (Intrusion Detection Systems, IDS) tradicionales, al reducir los falsos positivos mediante la simulación de dispositivos y aplicaciones reales en la red. Los mismos autores documentan que el costo promedio de una brecha de datos creció a USD 4,35 millones en 2022, un incremento del 12,7% respecto al año anterior, lo que sitúa la detección temprana como una prioridad económica además de técnica (Devi Priya & Chakkaravarthy, 2023, p. 1).

La containerización emergió como un habilitador clave para los honeypots cloud-native. Yu et al. (2024, p. 1) plantean que los honeypots sirven como mecanismo de alerta temprana que proporciona información sobre el comportamiento del atacante durante la explotación, y que la tecnología de contenedores permite construir arquitecturas de honeynet más flexibles y escalables. Sin embargo, los mismos autores identifican cuatro limitaciones persistentes en las arquitecturas existentes: la incapacidad de simular ataques multihop sobre múltiples dispositivos, el bajo nivel de simulación de redes protegidas, la imposibilidad de

capturar ataques en profundidad y la ausencia de métricas de prueba para efectos de deception (Yu et al., 2024, p. 2). Para abordar estas limitaciones proponen HoneyFactory, una arquitectura basada en contenedores Docker que genera honeynets dinámicos alineados con la topología real de la red protegida.

Otros trabajos han explorado la integración de honeypots con servicios cloud nativos. Chin et al. (2025, preprint de arXiv aún no sometido a revisión por pares, Secc. I) señalan que los firewalls basados en reglas estáticas resultan insuficientes en entornos cloud, donde la elasticidad y el multi-tenancy incrementan la superficie de ataque y dificultan las respuestas manuales. Su propuesta integra honeypots de mediana y alta interacción (Cowrie) con Azure Sentinel, Logic Apps y MITRE ATT&CK en un pipeline de respuesta en tiempo real. Los resultados experimentales demuestran un tiempo medio de bloqueo (Mean Time to Block, MTTB) promedio de 0,86 segundos para más de 12.000 intentos SSH (Chin et al., 2025, Secc. V). Machmeier (2024, p. 3) complementa esta perspectiva al sostener que los servicios de internet enfrentan más ataques que nunca y que los honeypots representan una herramienta fundamental para aprender del atacante directamente en el entorno cloud.

La Tabla 1 sintetiza los trabajos más relevantes en esta área, comparando sus características técnicas y alcance. El análisis revela un patrón recurrente: la mayoría de los frameworks cloud-native se construyen sobre honeypots de red (SSH, HTTP, Telnet) y carecen de cobertura sobre activos de identidad cloud, como claves de acceso IAM, secretos en servicios de gestión de credenciales o endpoints de base de datos señuelo. Esta brecha es particularmente relevante para organizaciones que operan sobre AWS, donde las credenciales de acceso constituyen el principal vector de compromiso.

**Tabla 1***Comparación de frameworks de honeypots en entornos cloud*

| Trabajo                            | Año         | Tipo de honeypot                    | Plataforma        | Det. automática                      | Identidad | Regulatorio               |
|------------------------------------|-------------|-------------------------------------|-------------------|--------------------------------------|-----------|---------------------------|
| Devi Priya & Chakkaravarthy (2023) | 2023        | Contenedorizado (red)               | Cloud genérico    | No                                   | No        | No                        |
| Machmeier (2024)                   | 2024        | Red (SSH, HTTP)                     | Cloud genérico    | No                                   | No        | No                        |
| Yu et al. / HoneyFactory (2024)    | 2024        | Contenedorizado, honeynet           | Docker/genérico   | No                                   | No        | No                        |
| Chin et al. (2025)                 | 2025        | Red (SSH, Cowrie)                   | Azure             | Sí (Sentinel)                        | No        | No                        |
| Morić et al. (2025)                | 2025        | Múltiples (LI/HI)                   | On-premise/cloud  | Parcial                              | No        | No                        |
| <b>HoneyAgent [propuesta]</b>      | <b>2026</b> | <b>Identidad (IAM, SM, S3, RDS)</b> | <b>AWS nativo</b> | <b>Sí (CloudTrail + EventBridge)</b> | <b>Sí</b> | <b>Sí (PCI-DSS, BCRA)</b> |

*Nota.* Elaboración propia basada en los trabajos analizados. LI/HI = Low-Interaction/High-Interaction. IAM = Identity and Access Management. SM = Secrets Manager.

## 2.2 Honeytokens de identidad y credenciales

Mientras los honeypots de red simulan servicios y hosts completos, los honeytokens representan un enfoque más granular: artefactos individuales como contraseñas, archivos, registros de base de datos o tokens de API, diseñados para funcionar como indicadores de compromiso cuando son accedidos o utilizados por un atacante. Reti et al. (2024, Secc. I) describen los honeytokens como piezas de información falsa que, a diferencia de los honeypots convencionales, pueden colocarse tanto en entornos de decepción como en sistemas de producción reales, funcionando como un sistema de detección de intrusiones específico del host. Esta propiedad los convierte en una herramienta de alta señal: si una credencial falsa es utilizada, esa activación constituye por definición evidencia de compromiso, sin necesidad de cruzar con otros logs ni superar umbrales de detección.

El ciclo de vida de las credenciales robadas ha sido analizado empíricamente por Rabzelj y Sedlar (2025, p. 1), quienes presentan uno de los estudios más exhaustivos sobre la explotación real de credenciales filtradas. Su investigación analiza más de 27.000 millones de credenciales (aproximadamente 4.000 millones únicas) y despliega una red distribuida de 39 honeypots en 9 redes durante un año, capturando cerca de 18 millones de intentos de autenticación SSH y Telnet y más de 750.000 solicitudes HTTP. Sus hallazgos muestran que las credenciales robadas emergen efectivamente en ataques reales, con una superposición significativa entre los intentos de login en honeypots y las wordlists comunes utilizadas por herramientas de reconocimiento masivo (Rabzelj & Sedlar, 2025, p. 2). Estos resultados tienen implicaciones directas para el diseño de honeytokens de identidad: si los atacantes priorizan credenciales provenientes de leaks y wordlists conocidas, diseñar honeytokens con patrones similares maximiza la probabilidad de activación.

El desafío central en el diseño de honeytokens es la escalabilidad: crear manualmente tokens convincentes y contextualmente coherentes para cada servicio o usuario es un proceso laborioso que no escala en organizaciones medianas o grandes. Reti et al. (2024, Secc. I) identifican este obstáculo como el principal freno para la adopción masiva y proponen el uso de LLMs como solución. En su trabajo evalúan 210 estructuras de prompt distintas para generar siete tipos de honeytokens: honeywords, archivos de configuración, bases de datos, archivos de log y robots.txt, entre otros, sobre múltiples modelos (GPT-3.5, GPT-4, LLaMA2 y Gemini). Los honeywords generados con GPT-3.5 resultaron estadísticamente menos distinguibles de contraseñas reales que los producidos por métodos anteriores, con una tasa de éxito en detección del 15,15% frente al 29,29% reportado por enfoques previos (Reti et al., 2024, Secc. III). En términos concretos, los honeytokens generados por LLMs son más convincentes, es decir, más difíciles de identificar como falsos, que los producidos por algoritmos determinísticos.

La convergencia entre honeypots de identidad, análisis de credenciales a escala y generación asistida por LLM configura un espacio de investigación activo pero aún fragmentado. Los trabajos analizados abordan componentes individuales del problema sin integrarlos en un framework cohesivo que opere sobre infraestructura cloud real en contextos regulatorios específicos.

### **2.3 La convergencia honeypots + LLM: del alerta al razonamiento**

La integración de LLMs en los honeypots no se limita a la generación de contenido más convincente: representa un salto cualitativo en la capacidad de análisis y respuesta. Bridges et al. (2025, Secc. 2) sistematizan esta área identificando que los honeypots tradicionales enfrentan una tensión de diseño fundamental. Los sistemas de baja interacción son seguros pero fácilmente detectables, mientras que los de alta interacción son creíbles pero costosos y riesgosos. La promesa de los LLMs es resolver esta tensión: lograr la fidelidad de un sistema de alta interacción con la simplicidad operativa de uno de baja interacción, mediante la generación de respuestas dinámicas y contextualmente coherentes.

Bridges et al. (2025, Secc. 3) proponen tres sistematizaciones para la integración LLM-honeypot: fingerprinting (detección de honeypots orientada a fortalecer su camuflaje), simulación (uso de LLMs para generar respuestas convincentes a comandos del atacante), y análisis de datos (transformación de logs de honeypot en inteligencia de amenazas accionable). Esta última categoría es la más relevante para el presente trabajo: la capacidad de un agente LLM de razonar sobre eventos capturados, identificar patrones de ataque y generar respuestas explicables va más allá de lo que cualquier sistema de alertas basado en reglas puede ofrecer. Sin embargo, los propios autores reconocen que el campo todavía carece de un framework de código abierto cohesivo que integre todos estos componentes, lo que constituye un obstáculo que frena el progreso y la reproducibilidad de la investigación (Bridges et al., 2025, Secc. 3).

En paralelo, la investigación sobre automatización de Centros de Operaciones de Seguridad (Security Operations Centers, SOC) con LLMs ha madurado de forma significativa. Srinivas et al. (2025, p. 1) documentan que los procesos manuales y basados en reglas de los SOC tradicionales resultan insuficientes ante el volumen y la sofisticación de las amenazas actuales, generando alert fatigue, altas tasas de falsos positivos y dependencia excesiva de analistas humanos. Su revisión sistemática, basada en el análisis de 105 publicaciones entre 2022 y 2025, identifica ocho funciones del SOC donde los LLMs y agentes de IA muestran potencial de automatización: síntesis de logs, triage de alertas, inteligencia de amenazas, gestión de tickets, respuesta a incidentes, generación de reportes, descubrimiento de activos y gestión de vulnerabilidades.

Vinay (2025, preprint de arXiv aún no sometido a revisión por pares, Secc. I) traza la evolución arquitectónica de la inteligencia artificial agéntica (agentic AI) en ciberseguridad en cinco generaciones, desde los LLMs puramente textuales hasta los pipelines autónomos con esquemas de herramientas estandarizados. Los sistemas de segunda generación (Gen-2), denominados agentes aumentados con herramientas (tool-augmented agents), son especialmente relevantes para este trabajo: combina el razonamiento del LLM con la capacidad de ejecutar llamadas a APIs, consultar bases de datos de inteligencia de amenazas y generar acciones estructuradas sobre sistemas externos (Vinay, 2025, Secc. II). La arquitectura ReAct, con ciclos iterativos de razonamiento, acción y observación, es el patrón dominante en esta generación y permite que el agente valide sus hipótesis contra datos reales antes de emitir una conclusión. Sin embargo, estos sistemas presentan limitaciones conocidas: la falta de verificación cruzada entre herramientas puede propagar errores en cascada, y la ausencia de memoria persistente dificulta el seguimiento de incidentes de larga duración (Vinay, 2025, Secc. II).

La Tabla 2 compara los trabajos más representativos que integran LLMs con honeypots o pipelines de análisis de amenazas, destacando el alcance de cada propuesta en términos de tipo de honeypot, capacidad de análisis, integración cloud y contexto regulatorio.

**Tabla 2**

*Integración de LLMs en honeypots y pipelines de análisis de amenazas*

| Trabajo                       | Año         | Capacidad LLM                                   | Tipo de honeypot                    | Tool use            | Explainability | Cloud native    | Regulatorio               |
|-------------------------------|-------------|-------------------------------------------------|-------------------------------------|---------------------|----------------|-----------------|---------------------------|
| Reti et al. (2024)            | 2024        | Generación de honeytokens                       | Honeytokens (7 tipos)               | No                  | No             | No              | No                        |
| Bridges et al. / SoK (2025)   | 2025        | Simulación, análisis de logs                    | Shell (SSH, HTTP)                   | Parcial             | No             | No              | No                        |
| Srinivas et al. (2025)        | 2025        | Automatización SOC (8 funciones)                | N/A (SOC general)                   | Sí                  | Parcial        | No              | No                        |
| Vinay (2025)                  | 2025        | Agentes tool-augmented (Gen-2)                  | N/A (SOC/IR)                        | Sí                  | No             | No              | No                        |
| <b>HoneyAgent [propuesta]</b> | <b>2026</b> | <b>Razonamiento + reportes con trazabilidad</b> | <b>Identidad (IAM, SM, S3, RDS)</b> | <b>Sí (4 tools)</b> | <b>Sí</b>      | <b>Sí (AWS)</b> | <b>Sí (PCI-DSS, BCRA)</b> |

*Nota.* Elaboración propia. Tool use = capacidad de invocar herramientas externas durante el razonamiento. Explainability = generación de justificación auditable. SOC = Security Operations Center. IR = Incident Response.

## 2.4 Análisis del gap y posicionamiento de la propuesta

La revisión de la literatura expuesta en las secciones anteriores permite identificar con precisión la brecha que motiva la presente propuesta. La investigación existente ha avanzado de forma relevante en cada uno de los tres ejes analizados, a saber: honeypots cloud-native, honeytokens de identidad y agentes LLM para ciberseguridad; sin embargo, lo ha hecho de manera compartimentada. Ningún trabajo previo integra los tres elementos en un framework único, configurable y operativo sobre infraestructura cloud real.

En términos específicos, el gap se articula en cuatro dimensiones. En primer lugar, los frameworks cloud-native existentes, como los analizados por Chin et al. (2025), Devi Priya y Chakkaravarthy (2023) y Yu et al. (2024), se concentran en honeypots de red (SSH, HTTP) y

carecen de cobertura sobre activos de identidad cloud-native, como usuarios IAM falsos, secretos en AWS Secrets Manager o endpoints de base de datos señuelo. En segundo lugar, los trabajos sobre honeytokens de identidad, como los de Rabzelj y Sedlar (2025) y Reti et al. (2024), abordan la generación y validación de tokens convincentes, pero no los integran en un pipeline de detección cloud ni en un framework de respuesta automatizada. En tercer lugar, los trabajos que combinan LLMs con honeypots, sistematizados por Bridges et al. (2025), se enfocan en honeypots de shell y no en honeypots de identidad sobre AWS. En cuarto lugar, ninguno de los trabajos revisados considera el contexto regulatorio de la industria financiera latinoamericana, en particular el cumplimiento de PCI-DSS v4.0.1 y la Comunicación A 8398 del Banco Central de la República Argentina (BCRA), que establece requisitos específicos de gestión de ciberincidentes para proveedores de servicios de pago.

HoneyAgent se posiciona en esta intersección vacía: un framework cloud-native sobre AWS que despliega honeypots de identidad configurables mediante YAML, integra un pipeline de detección basado en CloudTrail, EventBridge y GuardDuty, y utiliza un agente LLM con tool use para razonar sobre cada activación y generar reportes explicables con trazabilidad regulatoria. El Capítulo 4 presenta la arquitectura completa de la propuesta y los mecanismos que resuelven cada una de las dimensiones del gap identificado.

## **3 Marco Teórico**

### **3.1 Honeypots y Deception Technology**

#### **3.1.1 Introducción**

El diseño e implementación de HoneyAgent descansa sobre un conjunto de conceptos técnicos que requieren una delimitación precisa para comprender las decisiones de arquitectura presentadas en el Capítulo 4. El primero de esos conceptos es el honeypot: la unidad fundamental de deception technology sobre la que el framework opera. Esta sección desarrolla la definición, la taxonomía y las propiedades operativas de los honeypots, con énfasis en su evolución hacia entornos cloud, que constituye el contexto directo de la propuesta.

#### **3.1.2 Resumen**

Los honeypots son sistemas señuelo cuyo valor reside en la ausencia esperada de interacción legítima: cualquier acceso constituye, por definición, una señal de actividad anómala (Morić et al., 2025, p. 2). La literatura distingue entre honeypots de baja, media y alta interacción según el nivel de emulación del sistema objetivo, y entre honeypots de producción e investigación según su propósito operativo (Morić et al., 2025, p. 8). En entornos cloud, los honeypots se integran con servicios administrados que habilitan arquitecturas distribuidas y nativas de la plataforma, superando las limitaciones de las implementaciones on-premise tradicionales (Devi Priya & Chakkaravarthy, 2023, p. 1).

#### **3.1.3 ¿Qué es un honeypot?**

Un honeypot es un recurso computacional desplegado con la intención deliberada de ser comprometido o examinado por actores no autorizados, con el propósito de estudiar su comportamiento, capturar sus técnicas y generar inteligencia de amenazas accionables. Morić et al. (2025, p. 2) definen un honeypot como un sistema señuelo o aplicación diseñada para atraer y examinar actividades maliciosas, que funciona como trampa para atacantes al tiempo

que proporciona inteligencia valiosa sobre amenazas emergentes. En términos prácticos, los honeypots permiten identificar vulnerabilidades del sistema, analizar el comportamiento de los adversarios y fortalecer las defensas de red contra ataques como el phishing, los ataques de denegación de servicio distribuido y otras intrusiones (Morić et al., 2025, p. 2).

La característica definitoria de un honeypot no reside en sus capacidades técnicas sino en su propósito: a diferencia de los activos de producción, no tiene valor operativo legítimo, por lo que cualquier interacción con él es, por definición, sospechosa. Esta propiedad le confiere una tasa de falsos positivos estructuralmente baja, ya que no existe tráfico legítimo que deba ser filtrado ni umbrales de comportamiento que calibrar. La señal que genera un honeypot es cualitativamente distinta a la que producen los sistemas de detección basados en firmas o en anomalías, pues no requiere comparación con líneas de base ni ajuste de sensibilidad, sino simplemente la detección de cualquier tipo de acceso al señuelo (Morić et al., 2025, p. 2).

### **3.1.4 Clasificación de honeypots**

La taxonomía de honeypots más extendida en la literatura distingue entre dos dimensiones ortogonales: el nivel de interacción que ofrecen al atacante y el propósito operativo para el que fueron diseñados (Morić et al., 2025, p. 8).

En términos de nivel de interacción, los honeypots de baja interacción (Low-Interaction Honeypots, LIH) emulan servicios o protocolos de forma limitada, respondiendo a un conjunto predefinido de inputs sin exponer un sistema operativo completo. Su ventaja principal es la simplicidad operativa y el riesgo reducido: si el atacante logra salir del entorno emulado, no tiene acceso a un sistema real subyacente. Los honeypots de alta interacción (High-Interaction Honeypots, HIH), en cambio, replican sistemas completos y reales, lo que los hace más convincentes y capaces de capturar técnicas de ataque sofisticadas, a costa de una carga de gestión y un riesgo operacional significativamente

mayores. Entre ambos extremos, los honeypots de interacción media (Medium-Interaction Honeypots, MIH) combinan elementos de los dos enfoques: ofrecen mayor fidelidad que los LIH mediante la emulación de comportamientos específicos del sistema, pero sin exponer una superficie completa de sistema operativo (Morić et al., 2025, p. 8).

La segunda dimensión de clasificación distingue entre honeypots de producción y honeypots de investigación. Los primeros se despliegan dentro de infraestructuras operativas reales, con el objetivo de detectar actividad maliciosa dirigida a los sistemas de la organización. Los segundos se orientan a la captura de datos sobre el comportamiento de los atacantes en general, con fines académicos o de inteligencia de amenazas (Morić et al., 2025, p. 3). HoneyAgent adopta el modelo de producción: sus señuelos se despliegan en el entorno AWS real de la organización y su objetivo es la detección temprana de actividad dirigida a credenciales e identidades cloud, no la investigación general del ecosistema de amenazas.

### **3.1.5 Deception Technology en entornos cloud**

La deception technology es el marco conceptual que engloba el conjunto de técnicas y estrategias orientadas a engañar al atacante, inducirlo a interactuar con activos controlados y, a partir de esa interacción, obtener inteligencia o generar alertas de alta precisión. Los honeypots son la expresión más conocida de este paradigma, pero la deception technology abarca también los honeytokens, las honeyfiles, las honeynets y otros mecanismos que comparten la misma lógica: crear señuelos convincentes cuya activación constituye evidencia directa de compromiso (Morić et al., 2025, p. 2).

En el contexto del cloud computing, este paradigma ha adquirido una dimensión nueva. Devi Priya y Chakkaravarthy (2023, p. 1) señalan que los honeypots desplegados en infraestructura cloud ofrecen mayor visibilidad que los sistemas de detección de intrusiones (Intrusion Detection Systems, IDS) tradicionales, al reducir los falsos positivos mediante la simulación de dispositivos y aplicaciones reales en la red. La elasticidad y la automatización

propias de los entornos cloud permiten, además, escalar el despliegue de señuelos y actualizar su configuración sin interrupciones operativas, algo que no era viable en arquitecturas on-premise.

En plataformas como AWS, la deception technology encuentra en los servicios administrados un conjunto de primitivas nativas sobre las que construir señuelos de alta fidelidad. Un usuario de AWS Identity and Access Management (IAM) ficticio es indistinguible de uno legítimo hasta el momento en que es utilizado; un bucket de Amazon Simple Storage Service (S3) con un nombre plausible no genera ninguna diferencia visible en la consola de administración hasta que recibe una solicitud de acceso no autorizada. Esta propiedad de invisibilidad operativa es central en el diseño de HoneyAgent: los señuelos deben ser suficientemente creíbles para que un atacante que ha obtenido acceso al entorno los considere objetivos válidos, sin que el equipo de seguridad deba invertir recursos significativos en mantener su plausibilidad.

Yu et al. (2024, p. 2) documentan que las arquitecturas de honeynet existentes presentan limitaciones en términos de escala y profundidad, y que la complejidad operativa es uno de los principales obstáculos para su adopción en entornos productivos. El enfoque de configuración declarativa mediante YAML que adopta HoneyAgent responde directamente a este obstáculo: un único archivo define el conjunto completo de señuelos a desplegar, sin necesidad de conocimiento especializado sobre cada servicio de AWS involucrado.

## **3.2 Canary Tokens e Identidad en la Nube**

### **3.2.1 Introducción**

El segundo eje conceptual de HoneyAgent son los canary tokens y, más ampliamente, los honeypots de identidad: señuelos que no simulan servicios de red sino credenciales, usuarios y recursos cloud que, al ser activados, revelan que un atacante ha obtenido acceso al entorno. Esta sección define los canary tokens, describe los servicios de AWS sobre los que

HoneyAgent despliega sus señuelos de identidad, y establece el vínculo entre la naturaleza de esos servicios y el modelo de detección propuesto.

### 3.2.2 Resumen

Los canary tokens son artefactos de información deliberadamente falsos cuya activación constituye evidencia directa de compromiso, sin necesidad de cruzar con otros logs ni superar umbrales de detección (Reti et al., 2024, Secc. 2.1). En el contexto de AWS, los servicios IAM, Secrets Manager, S3 y RDS concentran la mayor parte de los vectores de ataque basados en identidad y credenciales registrados en los principales estudios de inteligencia de amenazas (IBM Security, 2025; Verizon Business, 2025, p. 10). HoneyAgent despliega señuelos sobre cada uno de estos servicios, combinando la lógica de los canary tokens con la infraestructura nativa de AWS para maximizar la probabilidad de detección ante accesos no autorizados.

### 3.2.3 ¿Qué son los canary tokens?

Los canary tokens son piezas de información deliberadamente falsas, diseñadas para parecer legítimas a un atacante pero que, al ser accedidas o utilizadas, generan una notificación que alerta al equipo de seguridad. Reti et al. (2024, Secc. 2.1) los definen como artefactos que, a diferencia de los honeypots convencionales, pueden colocarse tanto en entornos de decepción como en sistemas de producción reales, funcionando como un sistema de detección de intrusiones específico del host. Esta propiedad los distingue funcionalmente de los honeypots de red: mientras estos últimos simulan sistemas completos accesibles desde la red, los canary tokens son objetos discretos que pueden embeberse en cualquier entorno sin alterar el funcionamiento del sistema que los contiene.

El valor de detección de un canary token reside en su especificidad. Si una credencial falsa es utilizada para autenticarse, esa activación constituye por definición evidencia de compromiso, sin necesidad de cruzar eventos ni superar umbrales de anomalía (Reti et al.,

2024, Secc. 2.1). Esta característica los convierte en una herramienta de muy alta señal en entornos donde el volumen de alertas convencionales representa un problema operativo reconocido: Srinivas et al. (2025, p. 1) documentan que los procesos manuales y basados en reglas de los centros de operaciones de seguridad tradicionales resultan insuficientes ante el volumen y la sofisticación de las amenazas actuales, generando alert fatigue y altas tasas de falsos positivos.

El diseño de canary tokens convincentes plantea un desafío de escala. Reti et al. (2024, Secc. 2.1) identifican como principal obstáculo para su adopción masiva la dificultad de generar manualmente tokens contextualmente coherentes para cada servicio o usuario en organizaciones medianas o grandes. HoneyAgent resuelve este problema a través de la configuración declarativa en YAML: el operador define qué tipos de señuelos desplegar y el framework se encarga de provisionarse sobre los servicios de AWS correspondientes, sin requerir intervención manual por cada activo.

### **3.2.4 Gestión de identidad en AWS: IAM, Secrets Manager, S3 y RDS**

En infraestructuras de AWS, el acceso a los recursos se gestiona a través de un conjunto de servicios que concentran tanto el plano de control de la plataforma como los datos más sensibles de las organizaciones. IBM Security (2025) identifica que los ataques basados en identidad representaron el 30% de todas las instrucciones registradas en 2024, en un contexto de mercado creciente de credenciales robadas, con un incremento del 12% en la oferta de credenciales de infostealers a la venta en la dark web respecto del año anterior. Izhikevich et al. (2023, p. 313) complementan este panorama al demostrar que los atacantes realizan actividades de reconocimiento sobre infraestructuras cloud de manera sistemática y automatizada, probando credenciales y explorando recursos en plazos que superan ampliamente la capacidad de respuesta de los equipos que dependen de alertas reactivas.

AWS Identity and Access Management (IAM) es el servicio que controla la autenticación y autorización sobre todos los recursos de la plataforma. Su centralidad en la arquitectura de seguridad de AWS lo convierte en el objetivo prioritario de los atacantes: comprometer una credencial IAM equivale a obtener las llaves del entorno. Verizon Business (2025, p. 10) señala que el abuso de credenciales continúa siendo el vector de ataque más frecuente en brechas de seguridad a nivel global. En ese contexto, un usuario IAM ficticio con permisos limitados actúa como señuelo de alta fidelidad: es indistinguible de un usuario legítimo hasta el momento en que alguien intenta utilizarlo, y cualquier operación realizada bajo su identidad queda registrada en AWS CloudTrail con el contexto completo de la acción.

AWS Secrets Manager es el servicio orientado al almacenamiento y la rotación de secretos como contraseñas de bases de datos, claves de API y tokens de acceso a servicios externos. Desde la perspectiva de la deception technology, un secreto falso almacenado en este servicio representa un señuelo de precisión: cualquier intento de recuperar su valor implica una llamada autenticada a la API, que queda registrada con el identificador del principal que la realizó. Bridges et al. (2025, Secc. 4.3) identifican la generación de inteligencia de amenazas accionable a partir de eventos capturados como una de las aplicaciones más relevantes de los agentes LLM en pipelines de honeypots, y la activación de un secreto señuelo es exactamente el tipo de evento discreto y de alta certeza sobre el que ese razonamiento puede operar.

Amazon Simple Storage Service (S3) es ampliamente utilizado para almacenar datos de aplicaciones, backups y artefactos de despliegue, lo que lo convierte en un objetivo frecuente para atacantes que buscan exfiltrar información sensible tras comprometer credenciales de acceso. Verizon Business (2025, p. 10) reporta que el abuso de credenciales continúa siendo el vector de ataque más frecuente en brechas de seguridad a nivel global, un patrón que también resulta aplicable a los servicios de almacenamiento cloud como S3. Un

bucket S3 con un nombre plausible y objetos con nombres sugestivos actúa como trampa de alta probabilidad: un atacante que explora el entorno tras comprometer una credencial IAM lo identificará como un activo de valor e intentará acceder a él, generando el evento de detección que activa el pipeline de HoneyAgent.

Amazon Relational Database Service (RDS) proporciona bases de datos relacionales administradas, frecuentemente utilizadas para almacenar datos transaccionales y de clientes en organizaciones del sector fintech. Izhikevich et al. (2023, p. 313) documentan que el reconocimiento automatizado sobre infraestructura cloud incluye el escaneo sistemático de servicios y puertos expuestos como parte del ciclo estándar de exploración post-compromiso, un patrón consistente con el escaneo de endpoints de bases de datos administradas como RDS. Un endpoint de RDS ficticio con credenciales de acceso aparentes sirve como señuelo para atacantes que, tras comprometer credenciales IAM o de aplicación, intentan pivotar hacia las bases de datos del entorno, generando eventos de conexión que son capturados y analizados por el framework.

### **3.2.5 Honeypots de identidad y credenciales**

Los honeypots de identidad representan una especialización del paradigma general de los honeypots aplicada específicamente a credenciales, tokens y usuarios: en lugar de simular servicios de red, crean activos de autenticación falsos cuya utilización por parte de un atacante es detectada y registrada. Rabzelj y Sedlar (2025, p. 1) presentan uno de los estudios más exhaustivos sobre el ciclo de vida de las credenciales robadas, analizando más de 27.000 millones de credenciales y desplegando una red de 39 honeypots durante un año, capturando cerca de 18 millones de intentos de autenticación. Sus hallazgos muestran que las credenciales robadas emergen efectivamente en ataques reales, con superposición significativa entre los intentos de login capturados y las wordlists utilizadas por herramientas de reconocimiento masivo (Rabzelj & Sedlar, 2025, p. 2).

La implicación práctica de estos resultados para el diseño de HoneyAgent es directa: si los atacantes priorizan credenciales con patrones similares a los que circulan en leaks y wordlists conocidas, diseñar señuelos con nombres de usuario, nombres de buckets y nombres de secretos plausibles maximiza la probabilidad de activación. Reti et al. (2024, Secc. 6.2) confirman esta intuición al evaluar que los honeytokens generados con patrones estadísticamente similares a credenciales reales tienen tasas de detección por parte del atacante significativamente menores, lo que implica que son más probables de ser utilizados antes de que el atacante sospeche de su naturaleza.

La combinación de canary tokens de identidad con los servicios cloud de AWS configura el núcleo del mecanismo de detección de HoneyAgent. Cada señuelo opera como un indicador de compromiso de alta precisión: su activación no requiere análisis estadístico ni validación contra umbrales, sino simplemente la detección del evento en CloudTrail y su enrutamiento al agente LLM para el análisis de contexto. Esta arquitectura de detección basada en la certeza inherente de los honeypots de identidad es la que permite generar reportes con razonamiento auditable en lugar de simples alertas, cerrando la brecha que Bridges et al. (2025, Secc. 4.3) identifican como el principal déficit de los sistemas de honeypots tradicionales.

### **3.3 Detección de amenazas en AWS**

#### **3.3.1 Introducción**

El pipeline de detección de HoneyAgent está compuesto por cuatro servicios administrados de AWS que operan en secuencia: CloudTrail registra cada evento generado por los señuelos, EventBridge lo enruta mediante reglas configurables, Lambda ejecuta el agente LLM y GuardDuty aporta una capa de análisis de amenazas complementaria basada en machine learning. Esta sección describe cada uno de estos servicios, su función específica

dentro del pipeline y las razones que justifican su elección como componentes de la arquitectura propuesta.

### 3.3.2 Resumen

AWS CloudTrail registra cada acción realizada sobre los recursos del entorno AWS como un evento auditable que incluye la identidad del principal, el servicio involucrado y los parámetros de la operación (Amazon Web Services, 2026c, p. 1). Amazon EventBridge actúa como bus de eventos serverless que recibe esos registros y los enruta a destinos configurables mediante reglas de filtrado basadas en patrones JSON (Amazon Web Services, 2026a, p. 1). AWS Lambda ejecuta código sin necesidad de administrar servidores, siendo invocado por los eventos que EventBridge le entrega (Amazon Web Services, 2026d, p. 1). Amazon GuardDuty complementa el pipeline con detección continua de amenazas basada en inteligencia de amenazas externa y modelos de machine learning sobre las mismas fuentes que CloudTrail registra (Amazon Web Services, 2026b, p. 1).

### 3.3.3 AWS CloudTrail

AWS CloudTrail es el servicio de auditoría central de la plataforma AWS. Amazon Web Services (2026c, p. 1) lo define como un servicio que permite habilitar la auditoría operacional y de riesgos, la gobernanza y el cumplimiento normativo del entorno de AWS, registrando como eventos las acciones realizadas por usuarios, roles y servicios, tanto desde la consola de administración como desde la interfaz de línea de comandos y las APIs. Cada evento contiene el contexto completo de la acción: la identidad del principal que la realizó, el timestamp, la dirección IP de origen, el servicio involucrado y los parámetros de la solicitud junto con su respuesta.

En el contexto de HoneyAgent, CloudTrail es la fuente primaria de señales de detección. Cualquier interacción con un señuelo, sea el uso de un usuario IAM ficticio, el acceso a un bucket S3 señuelo, el intento de recuperar un secreto falso de Secrets Manager o

la conexión a un endpoint de RDS ficticio, genera un evento que CloudTrail registra de forma inmediata y automática. Amazon Web Services (2026c, p. 2) especifica que CloudTrail puede entregar estos eventos a Amazon EventBridge, lo que lo convierte en el punto de integración natural entre los señuelos y el pipeline de análisis del framework.

La elección de CloudTrail como fuente de detección responde a una propiedad arquitectónica fundamental: su cobertura es universal y no requiere agentes adicionales. Chin et al. (2025, Secc. I) señalan que los mecanismos de detección basados en agentes en entornos cloud presentan limitaciones frente a la elasticidad y el multi-tenancy de la plataforma. CloudTrail, al operar como servicio nativo de AWS, registra eventos con independencia del tipo de recurso o de la forma de acceso, sin necesidad de instrumentación específica por cada servicio señuelo.

### **3.3.4 Amazon EventBridge**

Amazon EventBridge es un servicio serverless que utiliza eventos para conectar componentes de aplicaciones, facilitando la construcción de arquitecturas orientadas a eventos a escala (Amazon Web Services, 2026a, p. 1). En su modelo operativo, los buses de eventos actúan como enrutadores que reciben eventos de fuentes diversas y los entregan a cero o más destinos configurables. EventBridge proporciona capacidades de filtrado basadas en patrones JSON, lo que permite que solo los eventos que cumplen criterios específicos activen los destinos registrados.

Dentro del pipeline de HoneyAgent, EventBridge cumple la función de enrutador inteligente entre CloudTrail y el agente Lambda. Una regla configurada para filtrar eventos de CloudTrail que involucren los recursos señuelo del framework activa automáticamente la función Lambda correspondiente cada vez que se detecta una interacción con un activo de decepción. Amazon Web Services (2026a, p. 1) describe además la capacidad de transformación previa a la entrega, que permite reformatear el evento antes de que llegue al

destino, simplificando el procesamiento en la función Lambda y reduciendo la lógica de parseo que el agente debe ejecutar.

### **3.3.5 AWS Lambda en pipelines de detección**

AWS Lambda es un servicio de computación serverless que ejecuta código en respuesta a eventos sin necesidad de administrar la infraestructura subyacente. Amazon Web Services (2026d, p. 1) define Lambda como un servicio de cómputo que gestiona de forma autónoma los recursos necesarios, incluyendo el mantenimiento del sistema operativo, el aprovisionamiento de capacidad y el escalado automático, permitiendo que el operador se concentre exclusivamente en el código de la función. Las funciones Lambda son invocadas por eventos provenientes de otras fuentes, como EventBridge, y ejecutan una lógica definida en respuesta a cada invocación.

En la arquitectura de HoneyAgent, Lambda es el ejecutor del agente LLM: recibe el evento de EventBridge, lo pasa como contexto al modelo Claude Sonnet 4.5 invocado a través de Amazon Bedrock, con sus herramientas configuradas, y persiste el reporte resultante. El modelo de ejecución efímera de Lambda, donde cada invocación corre en un entorno aislado sin estado persistente entre llamadas, es adecuado para el análisis de eventos de honeypot: cada activación de un señuelo es un evento discreto que debe analizarse de forma independiente, sin interferencia con invocaciones anteriores o concurrentes (Amazon Web Services, 2026d, p. 5). La integración nativa de Lambda con EventBridge y con los demás servicios de AWS simplifica además la cadena de autorización del agente.

La elección de Lambda como ejecutor del agente refleja la arquitectura sin componentes adicionales de HoneyAgent. Bridges et al. (2025, Secc. 4.1) señalan que el campo converge hacia una arquitectura canónica multi-componente para integrar honeypots y LLMs, a partir de propuestas hoy dispersas en la literatura. Lambda permite implementar ese

framework sin desplegar ni mantener servidores dedicados, reduciendo la superficie de ataque y el costo operativo del sistema.

### **3.3.6 Amazon GuardDuty**

Amazon GuardDuty es un servicio de detección de amenazas que monitorea, analiza y procesa de forma continua las fuentes de datos y logs del entorno AWS. Amazon Web Services (2026b, p. 1) lo describe como un servicio que utiliza feeds de inteligencia de amenazas, como listas de direcciones IP y dominios maliciosos conocidos, combinados con modelos de machine learning para identificar actividad sospechosa o potencialmente maliciosa. Sus fuentes de datos fundamentales incluyen los eventos de administración de CloudTrail, los registros de flujo de red (VPC Flow Logs) y los logs de consultas DNS, que analiza de forma continua sin necesidad de configuración adicional por parte del operador.

En el pipeline de HoneyAgent, GuardDuty opera como una capa de detección complementaria y ortogonal a los señuelos. Mientras que los honeypots de identidad generan alertas basadas en la certeza de la activación de un activo señuelo, GuardDuty aporta detección basada en el análisis de patrones de comportamiento sobre el conjunto del entorno. Amazon Web Services (2026b, pp. 1-2) describe la capacidad Extended Threat Detection, que identifica ataques multi-etapa correlacionando eventos de múltiples fuentes y tipos de recursos a lo largo del tiempo, incluyendo el uso de credenciales IAM comprometidas y el acceso anómalo a bases de datos RDS, escenarios directamente relevantes para los vectores que HoneyAgent monitorea.

La integración de GuardDuty responde a una lógica de defensa en profundidad. Chauhan y Shiaeles (2023, p. 427) analizan los frameworks de seguridad de los principales proveedores cloud y señalan que AWS combina herramientas nativas de detección con capacidades de automatización de respuesta. Un evento capturado por un señuelo de HoneyAgent puede estar acompañado de actividad adicional que GuardDuty detecta pero que

no involucra directamente los activos señuelo, como intentos de movimiento lateral o consultas de reconocimiento sobre recursos no monitoreados. La combinación de ambas señales enriquece el contexto disponible para el agente LLM al momento de generar el reporte de análisis.

## **3.4 Agentes LLM y razonamiento con herramientas**

### **3.4.1 Introducción**

La evolución de la inteligencia artificial aplicada a ciberseguridad atravesó, en los últimos años, una transformación estructural que va mucho más allá del procesamiento de lenguaje natural. Los primeros sistemas basados en grandes modelos de lenguaje (large language models, LLM) demostraron capacidades notables para resumir alertas, identificar indicadores de compromiso en texto no estructurado y asistir a analistas en la interpretación de reportes técnicos. Sin embargo, su naturaleza estática y su incapacidad de interactuar con fuentes de datos operativas en tiempo real los limitaban como herramientas de automatización autónoma. La respuesta a esas limitaciones fue el surgimiento de los agentes LLM: sistemas que combinan el razonamiento de un modelo de lenguaje con acceso estructurado a herramientas externas, memoria persistente y bucles iterativos de planificación y acción. Esta sección desarrolla los fundamentos teóricos de esa arquitectura; qué es un agente LLM, cómo opera el mecanismo de tool use y por qué la explicabilidad del razonamiento constituye un requisito crítico en entornos de seguridad con exigencias regulatorias.

### **3.4.2 Resumen**

Un agente LLM es un sistema de inteligencia artificial que combina la capacidad de razonamiento lingüístico de un modelo de lenguaje de gran escala con acceso estructurado a herramientas externas, módulos de memoria y ciclos iterativos de planificación y acción. A diferencia de los LLM convencionales, que producen una respuesta estática por cada

consulta, los agentes operan en bucles donde evalúan el estado actual, seleccionan una herramienta apropiada, interpretan el resultado y refinan su plan hasta alcanzar el objetivo definido (Lazer et al., 2026, p. 5). En entornos de operaciones de seguridad, esta arquitectura permite automatizar flujos de trabajo que de otro modo requieren intervención humana en cada etapa: enriquecimiento de alertas, consulta de inteligencia de amenazas, correlación de eventos y generación de reportes (Srinivas et al., 2025, p. 2). La convergencia entre agentes LLM y sistemas de honeypot configura una nueva categoría de soluciones capaces de razonar sobre cada activación, producir análisis contextualizados y generar trazabilidad completa del proceso de inferencia (Bridges et al., 2025, p. 2). En el marco de HoneyAgent, el agente LLM constituye el componente central de análisis: recibe eventos de activación de canary tokens, invoca herramientas especializadas y produce un reporte estructurado con justificación explícita de cada conclusión.

### 3.4.3 ¿Qué es un agente LLM?

La distinción entre un LLM y un agente LLM radica en su relación con el entorno de ejecución. Un LLM opera de forma reactiva: recibe una entrada, aplica su distribución de probabilidad sobre el espacio de tokens y produce una salida textual. Su razonamiento es estático, sin posibilidad de verificar afirmaciones contra datos externos ni de ejecutar acciones con efectos reales en sistemas operativos. Vinay (2025, p. 2) caracteriza estos sistemas de primera generación como herramientas eficaces para el procesamiento de texto, pero fundamentalmente restringidas porque no pueden ejecutar consultas externas, obtener datos de plataformas operativas ni mantener estado entre consultas; cualquier omisión o ambigüedad en la entrada puede generar respuestas plausibles pero incorrectas sin mecanismo de corrección.

Un agente LLM supera esas limitaciones mediante la incorporación de cuatro componentes estructurales: un módulo de razonamiento que descompone objetivos complejos

en subtareas, un sistema de memoria de corto y largo plazo, una capa de herramientas que permite interactuar con APIs y fuentes de datos externas, y un bucle iterativo de planificación, acción y reflexión (Lazer et al., 2026, p. 5). A diferencia del modelo estático, el agente puede suspender su proceso de razonamiento, invocar una herramienta externa, observar el resultado y continuar con información verificada, lo cual reduce drásticamente la dependencia de patrones aprendidos en favor de evidencia observada directamente.

Lazer et al. (2026, preprint de arXiv aún no sometido a revisión por pares, p. 6) formalizan este espectro mediante una taxonomía de cinco niveles de autonomía, desde la inferencia estática sin autonomía (nivel 0) hasta la planificación y aprendizaje continuo sin supervisión humana (nivel 4). Los sistemas de nivel 2 y 3, donde el agente usa herramientas con lógica predefinida y ejecuta flujos de trabajo orientados a objetivos con supervisión parcial, resultan los más relevantes para aplicaciones de seguridad como HoneyAgent, donde la auditabilidad de cada decisión y la posibilidad de revisión humana son requisitos no negociables en contextos de cumplimiento normativo.

#### **3.4.4 Tool Use: razonamiento con herramientas externas**

El mecanismo de tool use es el componente que transforma un LLM estático en un agente operacionalmente útil para ciberseguridad. A través de interfaces estructuradas como llamadas a funciones (function calls) o esquemas JSON predefinidos, el agente puede invocar herramientas externas de forma determinística: consultar registros de auditoría, ejecutar búsquedas en bases de datos de inteligencia de amenazas, realizar lookups de reputación de direcciones IP o despachar alertas hacia sistemas de notificación (Lazer et al., 2026, p. 5). Esta capacidad de interacción con el entorno es la que distingue a los sistemas agénticos del paradigma de IA generativa anterior.

El patrón arquitectónico más extendido para organizar este proceso es ReAct (Reason + Act), en el cual el agente alterna ciclos de razonamiento explícito con la ejecución de

acciones concretas, observa el resultado de cada herramienta invocada y actualiza su plan en consecuencia (Vinay, 2025, p. 3). Este enfoque iterativo permite al agente construir hipótesis verificables: en lugar de inferir si una dirección IP es maliciosa a partir únicamente de su distribución de entrenamiento, consulta activamente servicios de reputación y contrasta el resultado con el historial de actividad en los registros, reduciendo la tasa de alucinaciones sobre datos de estado del sistema.

La cuarta generación de sistemas agénticos descrita por Vinay (2025, pp. 4-5) introduce esquemas estandarizados mediante el Model Context Protocol (MCP), que define contratos de entrada y salida para cada herramienta disponible. Este enfoque garantiza reproducibilidad porque la misma invocación con los mismos parámetros produce el mismo resultado estructurado, y reduce el riesgo de ejecución de comandos arbitrarios fuera del conjunto autorizado. Vinay (2025, p. 5) señala que, aunque los esquemas controlan la estructura de cada llamada, no garantizan la corrección semántica del razonamiento subyacente; por eso la supervisión humana sigue siendo relevante para decisiones de alto impacto en entornos de producción.

En el contexto de HoneyAgent, las herramientas disponibles para el agente incluyen la consulta de registros de AWS CloudTrail, la verificación de historial de actividad de identidades IAM, el enriquecimiento con inteligencia de amenazas sobre IPs involucradas y el despacho de notificaciones por canales configurables. Cada invocación queda registrada como parte de la cadena de razonamiento del agente, lo cual garantiza trazabilidad completa del análisis y facilita la revisión posterior por parte del equipo de seguridad.

### **3.4.5 Explainability en sistemas de IA para ciberseguridad**

La adopción de sistemas de inteligencia artificial en entornos de seguridad enfrenta un desafío estructural que va más allá de la precisión de las inferencias: la interpretabilidad del razonamiento. Los LLM convencionales han sido caracterizados como sistemas de "caja

negra" porque producen respuestas sin explicitar el proceso que las generó (Srinivas et al., 2025, p. 2). Esta opacidad es especialmente problemática en contextos de seguridad donde cada decisión, como bloquear un acceso, escalar un incidente o iniciar una respuesta forense, requiere justificación auditable para cumplir con marcos normativos como PCI-DSS o lineamientos de organismos regulatorios.

La explicabilidad (explainability) en sistemas de IA para ciberseguridad se refiere a la capacidad del sistema de acompañar cada conclusión con una descripción comprensible del proceso que la sustenta, incluyendo las herramientas invocadas, los datos observados y la cadena de inferencias aplicada. El mecanismo de Chain-of-Thought (CoT), incorporado en los agentes LLM modernos, permite externalizar ese razonamiento como texto estructurado que puede revisarse, auditarse y archivar como parte del registro del incidente (Srinivas et al., 2025, p. 15). Esta característica convierte al agente en un sistema explicable por diseño, no por post-procesamiento.

Vinay (2025, p. 5) advierte que incluso en sistemas de cuarta generación con esquemas estandarizados, las acciones del agente requieren verificación humana porque los errores semánticos pueden producir llamadas sintácticamente correctas pero contextualmente erróneas. En entornos de alta criticidad, como la detección de activaciones de honeytokens en sistemas financieros, esta limitación refuerza la necesidad de que el agente no solo tome decisiones, sino que las fundamente con evidencia directamente observable y referenciable. La explicabilidad no es un atributo deseable sino un requisito funcional en ese contexto.

Bridges et al. (2025, p. 2) identifican la generación automatizada de inteligencia de amenazas a partir de honeypots como un área aún en etapa de prueba de concepto, donde el valor práctico depende de que el sistema produzca análisis que los equipos de seguridad puedan utilizar directamente en sus procesos de respuesta a incidentes. En HoneyAgent, la explicabilidad se implementa a través del reporte estructurado que el agente genera al

finalizar su análisis: cada sección corresponde a un paso razonado del proceso, desde la identificación de la entidad involucrada hasta la clasificación de la amenaza y las acciones recomendadas.

### **3.5 Marco regulatorio: PCI-DSS aplicado a fintechs latinoamericanas**

#### **3.5.1 Introducción**

El contexto regulatorio en el que operan las fintechs latinoamericanas, y particularmente las argentinas, combina estándares internacionales de seguridad de datos con marcos normativos locales que han evolucionado significativamente en los últimos años. Por un lado, el Payment Card Industry Data Security Standard (PCI-DSS) establece los requisitos de seguridad que deben cumplir todas las entidades que almacenan, procesan o transmiten datos de tarjetas de pago. Por otro, el Banco Central de la República Argentina (BCRA) ha expandido sus requisitos mínimos de gestión de riesgos tecnológicos y de seguridad de la información para incluir a los Proveedores de Servicios de Pago (PSP), categoría en la que se encuadran la mayoría de las fintechs del ecosistema local. La intersección entre ambos marcos configura un entorno de cumplimiento exigente que requiere controles técnicos robustos, trazabilidad completa de eventos de seguridad y capacidad de respuesta ante incidentes documentada. Esta sección analiza los requerimientos más relevantes de ese marco y fundamenta el rol de los honeypots como control complementario dentro de su estructura.

#### **3.5.2 Resumen**

PCI-DSS es un estándar desarrollado por el PCI Security Standards Council (PCI SSC) que establece los requisitos técnicos y operativos que deben cumplir las organizaciones que almacenan, procesan o transmiten datos de cuentas de pago (PCI Security Standards Council, 2024, p. 1). En su versión 4.0.1, el estándar organiza sus obligaciones en doce requerimientos agrupados en seis objetivos, entre los cuales el monitoreo continuo de accesos (Requerimiento 10) y las pruebas periódicas de seguridad (Requerimiento 11) son los que

presentan mayor alineación con las capacidades de detección de HoneyAgent. En Argentina, el BCRA extendió mediante la Comunicación "A" 8398 las obligaciones del marco de Requisitos Mínimos para la Gestión y Control de los Riesgos de Tecnología y Seguridad de la Información a los PSP incluidos en el Registro de PSP del BCRA, incorporándose como sujetos obligados con las mismas exigencias que las entidades financieras tradicionales (Banco Central de la República Argentina, 2026, párr. 1). Esta convergencia normativa obliga a las fintechs a implementar controles de seguridad equivalentes a los de la banca, en un ecosistema donde el procesamiento de pagos digitales las convierte en objetivo prioritario para actores maliciosos.

### **3.5.3 Estándar PCI-DSS: alcance y requerimientos relevantes**

PCI-DSS fue desarrollado para promover y fortalecer la seguridad de los datos de cuentas de pago y facilitar la adopción generalizada de medidas de seguridad consistentes a nivel global (PCI Security Standards Council, 2024, p. 1). El estándar aplica a cualquier entidad que almacene, procese o transmita datos de titulares de tarjeta (cardholder data, CHD) o datos de autenticación sensibles (sensitive authentication data, SAD), con independencia de su tamaño o volumen de transacciones (PCI Security Standards Council, 2024, p. 4). Esto incluye a comercios, procesadores, adquirentes, emisores y otros proveedores de servicios; categoría en la que se encuadran las fintechs que operan plataformas de pagos digitales.

Los doce requerimientos del estándar se organizan en seis objetivos de alto nivel: construcción y mantenimiento de redes seguras, protección de datos de cuentas, gestión de vulnerabilidades, implementación de controles de acceso fuertes, monitoreo y prueba regular de redes, y mantenimiento de una política de seguridad de la información (PCI Security Standards Council, 2024, p. 1). Para los propósitos de HoneyAgent, los más relevantes son el Requerimiento 10, que exige registrar y monitorear todos los accesos a componentes del

sistema y datos de titulares de tarjeta, y el Requerimiento 11, que obliga a probar la seguridad de sistemas y redes de manera periódica mediante técnicas de detección de intrusiones.

El Requerimiento 10 establece que los registros de auditoría deben implementarse para soportar la detección de anomalías y actividad sospechosa, además del análisis forense de eventos (PCI Security Standards Council, 2024, p. 236). Específicamente, el subrequerimiento 10.2.1 señala que los registros de auditoría deben emitir alertas al administrador del sistema y proveer datos a otros mecanismos de monitoreo, como los sistemas de detección de intrusiones (PCI Security Standards Council, 2024, p. 239). El subrequerimiento 10.4.1 exige que los eventos de seguridad sean revisados al menos una vez por día para identificar anomalías o actividad sospechosa. El Requerimiento 11.5, a su vez, establece que deben usarse técnicas de detección y prevención de intrusiones comparando el tráfico entrante con firmas conocidas y comportamientos de compromiso para detectar intrusiones en la red (PCI Security Standards Council, 2024, p. 283). Estos requerimientos constituyen el núcleo normativo sobre el cual HoneyAgent opera como control complementario.

### **3.5.4 Contexto regulatorio en Argentina: BCRA y fintechs**

El marco regulatorio de ciberseguridad para el sistema financiero argentino está articulado principalmente por el BCRA a través del texto ordenado sobre Requisitos Mínimos para la Gestión y Control de los Riesgos de Tecnología y Seguridad de la Información. Históricamente este marco aplicaba a entidades financieras autorizadas por el BCRA; sin embargo, la Comunicación "A" 8398, emitida el 5 de febrero de 2026, extendió su alcance para incorporar como sujetos obligados a los PSP incluidos en el Registro de PSP del BCRA (Banco Central de la República Argentina, 2026, párr. 1). Esta ampliación refleja el reconocimiento regulatorio de que las fintechs que operan en el ecosistema de pagos digitales concentran riesgos tecnológicos equivalentes a los de la banca tradicional.

La norma establece requisitos específicos para la gestión de relaciones con terceras partes, incluyendo la definición de roles y responsabilidades, medidas de seguridad acordes a los resultados de la evaluación de riesgos y procedimientos formales para la gestión de ciberincidentes en todas sus etapas (Banco Central de la República Argentina, 2026, Sección 10.2). Adicionalmente, el proceso de control y monitoreo exige procedimientos de auditoría interna sobre los procesos y actividades tercerizadas, con periodicidad definida en función del nivel de riesgo y criticidad de los servicios evaluando su impacto en el sistema de control interno (Banco Central de la República Argentina, 2026, Sección 10.5).

Para las fintechs que además operan en el ecosistema de pagos con tarjeta, la convergencia del marco regulatorio BCRA con los requerimientos de PCI-DSS genera una capa de exigencias que se refuerzan mutuamente: donde PCI-DSS establece requisitos técnicos específicos de seguridad de datos de pago, el marco BCRA exige procesos de gobernanza que aseguren el cumplimiento y la capacidad de respuesta ante incidentes. La ausencia de controles detectivos robustos representa un riesgo de cumplimiento simultáneo en ambos marcos, lo cual eleva el costo potencial de un incidente de seguridad no detectado oportunamente.

### **3.5.5 Honeypots como control complementario bajo PCI-DSS**

Los honeypots y, más específicamente, los canary tokens, no están mencionados de forma explícita en el texto de PCI-DSS. Sin embargo, sus capacidades de detección se alinean directamente con los objetivos de varios requerimientos del estándar, posicionándolos como controles complementarios de alta efectividad en entornos de fintechs. El sub requerimiento 10.2.1 establece que los sistemas de auditoría deben proveer datos a otros mecanismos de monitoreo, como los sistemas de detección de intrusiones (PCI Security Standards Council, 2024, p. 239). Un canary token activado dentro del entorno de datos de titulares de tarjeta (cardholder data environment, CDE) constituye exactamente ese tipo de señal: de alta

fidelidad, sin falsos positivos estructurales, y con trazabilidad inmediata de la identidad o credencial comprometida.

La sección A3.5 del Apéndice A3 del estándar, aplicable a proveedores de servicios designados, exige la implementación de una metodología para la identificación oportuna de patrones de ataque y comportamientos indeseados en todos los sistemas, incluyendo la identificación de anomalías o actividad sospechosa (PCI Security Standards Council, 2024, p. 368). Los honeytokens de identidad, como usuarios IAM ficticios o claves de API monitoreadas, satisfacen ese requisito por definición: cualquier uso de esas credenciales es actividad anómala que no requiere correlación estadística para ser clasificada como incidente, reduciendo el tiempo de detección a segundos desde el momento de la activación.

Bridges et al. (2025, p. 2) identifican la generación automatizada de inteligencia de amenazas a partir de la activación de honeypots como el objetivo de madurez más avanzado en este campo. En el contexto regulatorio de PCI-DSS, esa inteligencia tiene un valor adicional: no solo informa la respuesta al incidente, sino que contribuye al registro documental que las entidades deben mantener para demostrar cumplimiento durante una evaluación formal. HoneyAgent integra ambas dimensiones al generar reportes estructurados con cadena de razonamiento auditable para cada activación, proveyendo simultáneamente capacidad detectiva y evidencia documental alineada con los estándares de registro exigidos por el Requerimiento 10.

## 4 Propuesta de Intervención

### 4.1 Descripción del framework y arquitectura general

El presente capítulo corresponde a un proyecto de diseño e implementación: se desarrolla el framework HoneyAgent como propuesta de intervención y se lo valida mediante un prototipo mínimo viable, sometido a escenarios de ataque simulados sobre infraestructura real de AWS, siguiendo el cronograma de seis etapas detallado en la sección 4.8.

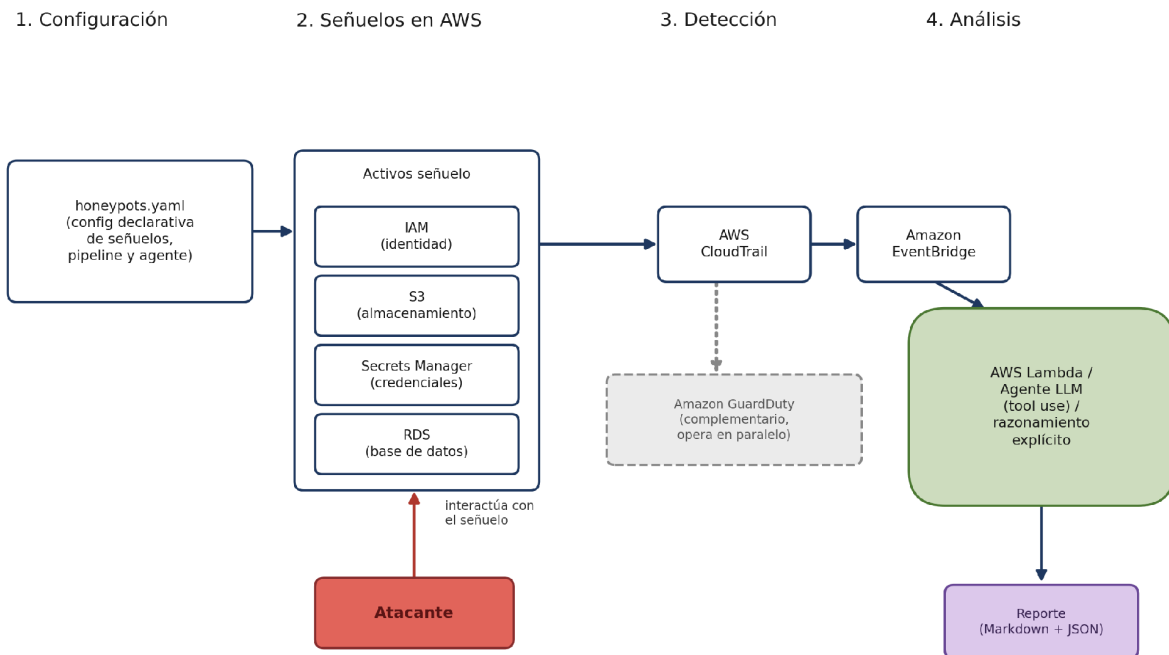
HoneyAgent es un framework configurable para el despliegue de honeypots distribuidos sobre infraestructura de Amazon Web Services, diseñado específicamente para organizaciones del sector fintech que operan bajo el estándar PCI DSS v4.0.1 y la normativa del Banco Central de la República Argentina. El framework integra tres capas funcionales que se corresponden con los objetivos específicos planteados en la sección 1.4: una capa de configuración declarativa, un pipeline de detección nativo de AWS y un agente de inteligencia artificial con capacidad de uso de herramientas.

La arquitectura general se organiza en cuatro etapas secuenciales. En primer lugar, la capa de configuración traduce las decisiones de un equipo de seguridad (qué señuelos desplegar, sobre qué servicios y con qué nivel de alerta) en infraestructura real mediante un archivo de configuración declarativo y plantillas de aprovisionamiento. En segundo lugar, los señuelos quedan desplegados sobre cuatro servicios de AWS frecuentemente atacados en incidentes de identidad: Identity and Access Management (IAM), Simple Storage Service (S3), Secrets Manager y Relational Database Service (RDS). En tercer lugar, cualquier interacción con esos activos señuelo, por definición ilegítima dado que no forman parte de ningún flujo productivo real, es capturada por AWS CloudTrail, enrutada por Amazon EventBridge y procesada por una función de AWS Lambda. Finalmente, esa función aloja al agente LLM, que analiza el evento, enriquece el contexto consultando herramientas externas, razona explícitamente sobre el nivel de riesgo y produce un reporte estructurado y auditable.

Esta organización en capas responde directamente al análisis del estado del arte presentado en el Capítulo 2: mientras que los honeypots tradicionales producen alertas que requieren interpretación manual por parte de un analista (Bridges et al., 2025), y los frameworks que incorporan LLMs suelen acoplar el modelo a un único proveedor de infraestructura o a un conjunto fijo de señuelos (Reti et al., 2024; Yu et al., 2024), HoneyAgent separa explícitamente la definición de los señuelos (capa de configuración), su despliegue (infraestructura como código) y su análisis (agente LLM agnóstico al tipo de señuelo). Esta separación es la que permite que el mismo agente analice eventos provenientes de un usuario IAM señuelo, un bucket S3 señuelo o un secreto de Secrets Manager señuelo sin necesidad de lógica de análisis diferenciada por tipo de activo: el agente recibe siempre la misma estructura de evento de CloudTrail y aplica el mismo proceso de razonamiento, enriquecido con las herramientas que considere pertinentes según el contexto.

La Figura 4.1 ilustra el flujo completo: un atacante que obtiene credenciales filtradas o descubre un activo señuelo interactúa con él de forma indistinguible de como lo haría con un recurso productivo; CloudTrail registra esa interacción como un evento de auditoría; EventBridge la enruta hacia la función Lambda según reglas que identifican a los activos señuelo por su identidad o nombre; y el agente LLM, alojado en esa función, produce un informe que documenta qué ocurrió, por qué se considera sospechoso y qué acción se recomienda, dejando una traza completa y auditable del proceso de detección y análisis.

**Figura 4.1**  
*Arquitectura general del framework HoneyAgent*



*Nota.* Elaboración propia. La flecha punteada indica que Amazon GuardDuty opera como capa de detección complementaria y en paralelo a los honeypots, sin intervenir en la cadena principal CloudTrail-EventBridge-Lambda descrita en la sección 4.4.

## 4.2 Capa de configuración (honeypots.yaml + Terraform/CDK)

El primer objetivo específico del trabajo establece la necesidad de una arquitectura de honeypots "configurable de forma declarativa mediante un archivo YAML" (sección 1.4.2). Esta decisión de diseño busca resolver una limitación práctica observada en el estado del arte: los frameworks de honeypots existentes suelen requerir modificaciones de código para agregar, modificar o retirar señuelos, lo que eleva la barrera operativa para equipos de seguridad con recursos acotados, una situación frecuente en organizaciones fintech medianas y pequeñas del ecosistema argentino (sección 1.3).

HoneyAgent resuelve esto mediante un archivo `honeypots.yaml` que centraliza tres tipos de definiciones: (a) los señuelos a desplegar, especificando su tipo (`iam_identity`, `s3_bucket`, `secrets_manager`, `rds_endpoint`), su configuración particular (nombre, política adjunta, objetos semilla, etc.) y su nivel de alerta;

(b) los parámetros del pipeline de detección (fuente de auditoría, motor de enrutamiento, función ejecutora y capa complementaria de detección); y (c) la configuración del agente LLM (modelo, herramientas habilitadas, formatos y destino de los reportes). Un equipo de seguridad puede así ampliar la superficie de señuelos (por ejemplo, agregando un segundo bucket S3 señuelo con un nombre que imite a un sistema productivo recién anunciado internamente) editando una sección del archivo y volviendo a aplicar la configuración, sin tocar el código del framework ni del agente.

La traducción de ese archivo declarativo a infraestructura real se realiza mediante plantillas de aprovisionamiento (Terraform o AWS CDK, según la preferencia del equipo adoptante): por cada entrada habilitada en `honeypots.yaml` se generan los recursos de AWS correspondientes (el usuario IAM y su política, el bucket y sus objetos semilla, el secreto, etc.) junto con la regla de Amazon EventBridge que vincula ese activo específico con la función Lambda analizadora. Este enfoque de infraestructura como código aporta dos beneficios directamente relevantes para el contexto regulatorio del proyecto: primero, garantiza que el despliegue sea reproducible y versionable, lo cual constituye evidencia documental ante una auditoría de PCI DSS o del BCRA; segundo, permite destruir y volver a desplegar el entorno de señuelos de forma controlada, reduciendo el riesgo de que un activo señuelo quede expuesto más tiempo del previsto o de que su configuración diverge silenciosamente de lo declarado.

Para validar este diseño en la práctica, se construyó un prototipo mínimo viable que implementa el primero de los cuatro tipos de señuelo (el honeypot de identidad) de punta a punta, y documenta los tres restantes como entradas deshabilitadas (`enabled: false`) en el mismo archivo de configuración, mostrando cómo el patrón se extiende sin alterar ni el pipeline ni el agente (ver Anexo correspondiente al prototipo). Esta decisión de recortar el alcance de la prueba de concepto a un único tipo de señuelo, mientras se documenta

explícitamente la extensión a los demás, permite concentrar el esfuerzo de validación en el núcleo arquitectónico común a los cuatro casos: la cadena CloudTrail, EventBridge, Lambda y el agente LLM.

### 4.3 Honeypots en AWS: diseño e implementación

El diseño de cada señuelo sigue un principio común derivado del análisis del marco teórico (sección 3.1.5): un honeypot efectivo en un entorno cloud debe ser indistinguible de un activo productivo para un atacante, pero inequívocamente identificable como ilegítimo para el sistema de detección. HoneyAgent aplica este principio mediante una combinación de verosimilitud superficial (nombres, políticas y metadatos que imitan convenciones reales de nomenclatura del entorno objetivo) y trazabilidad interna (etiquetas, prefijos o patrones reconocibles únicamente por las reglas de EventBridge, nunca expuestos al atacante).

**Honeypot de identidad (IAM).** Consiste en uno o más usuarios de IAM con nombres verosímiles (por ejemplo, `billing-readonly` o `ci-deploy-bot`) y credenciales de acceso de larga duración generadas pero nunca utilizadas legítimamente. La hipótesis de diseño es que estas credenciales pueden "filtrarse" deliberadamente en lugares donde un atacante realista buscaría secretos expuestos (repositorios públicos, archivos de configuración, tickets de soporte), y que cualquier uso posterior de esas credenciales constituye, por definición, una señal de compromiso. Cada usuario señuelo recibe una política de permisos mínima y cuidadosamente acotada, de forma que ninguna acción que el atacante intente pueda producir efectos reales sobre el entorno, y queda asociado a una regla de EventBridge que captura toda actividad de CloudTrail cuya identidad invocadora coincida con ese usuario.

**Honeypot de almacenamiento (S3).** Consiste en buckets con nombres que imitan convenciones de nomenclatura de sistemas productivos (por ejemplo, `statements-prod-backup` o `clientes-export-2026`), poblados con objetos

señuelo que contienen datos sintéticos con apariencia de información sensible (resúmenes de cuenta, exportaciones de clientes). La detección se basa en operaciones de listado, lectura o descarga sobre ese bucket específico, que CloudTrail registra como eventos de datos cuando se habilita esa categoría de auditoría.

**Honeypot de credenciales (Secrets Manager).** Consiste en secretos con nombres que sugieren contener credenciales de bases de datos productivas (por ejemplo, `prod/rds/app-readwrite`), cuyo valor son credenciales sintéticas que no corresponden a ninguna instancia real. Cualquier intento de recuperar el valor de ese secreto (mediante `GetSecretValue`) es, en sí mismo, una señal de que un actor está buscando activamente moverse lateralmente hacia la capa de datos.

**Honeypot de base de datos (RDS).** Consiste en un endpoint o registro DNS que simula la existencia de una instancia de base de datos relacional expuesta, sin una instancia real asociada, lo cual evita los costos de cómputo y almacenamiento que implicaría levantar una instancia RDS funcional únicamente con fines de señuelo. La detección se orienta a los intentos de resolución, conexión o escaneo dirigidos a ese endpoint.

Los cuatro tipos comparten una propiedad de diseño crítica: ninguno requiere que el equipo de seguridad anticipe el vector de ataque específico que un adversario va a utilizar. Basta con que el activo exista y sea descubrible (mediante reconocimiento, filtración de credenciales o enumeración de recursos) para que cualquier interacción con él constituya una señal de alta confianza, a diferencia de los sistemas de detección basados en firmas o umbrales que requieren conocer de antemano el patrón de ataque a identificar (sección 2.1).

#### **4.4 Pipeline de detección (CloudTrail, EventBridge, Lambda y GuardDuty)**

Una de las contribuciones centrales del framework, señalada en la sección 1.5, es que su pipeline de detección se apoya exclusivamente en servicios administrados de AWS, sin requerir la instalación ni el mantenimiento de agentes de terceros. Esta decisión

arquitectónica reduce tanto la superficie de ataque del propio framework (un agente de monitoreo de terceros mal configurado podría convertirse, paradójicamente, en un nuevo vector de compromiso) como su costo operativo, dado que los tres servicios involucrados (CloudTrail, EventBridge y Lambda) cuentan con niveles de uso incluidos en el free tier de AWS suficientes para el volumen de eventos que genera un entorno de señuelos.

El pipeline se compone de tres etapas:

**AWS CloudTrail** actúa como fuente de auditoría. Registra cada llamada a la API de AWS realizada sobre el entorno como un evento que documenta la identidad del principal invocador, el servicio y la operación involucrados, los parámetros de la solicitud, la dirección IP de origen y el resultado (sección 3.3.3). Esta es la fuente primaria de señales del framework: cualquier interacción con un activo señuelo (uso de un usuario IAM ficticio, acceso a un bucket S3 señuelo, intento de recuperar un secreto falso o conexión a un endpoint RDS ficticio) queda registrada automáticamente, sin que HoneyAgent deba instrumentar el activo en sí.

**Amazon EventBridge** cumple la función de enrutador inteligente entre CloudTrail y la función Lambda. Por cada señuelo habilitado en `honeypots.yaml`, la capa de configuración genera una regla de EventBridge cuyo patrón de evento identifica unívocamente a ese activo: por el nombre de usuario invocador en el caso de los honeypots de identidad, por el nombre del bucket o del secreto en los demás casos. Esta granularidad permite que el sistema reaccione exclusivamente ante interacciones con activos señuelo, descartando automáticamente el enorme volumen de actividad legítima que genera cualquier cuenta de AWS en producción, una propiedad que resulta indispensable para mantener la tasa de falsos positivos cercana a cero, a diferencia de los sistemas de detección de anomalías que deben calibrarse contra una línea base de comportamiento normal (sección 2.1).

**AWS Lambda** es el ejecutor del agente. Recibe el evento ya filtrado y enriquecido por EventBridge, lo traduce a un resumen estructurado (identidad invocadora, operación, origen, momento, resultado) y lo entrega como contexto al agente LLM. El modelo de ejecución efímera de Lambda (donde cada invocación corre en un entorno aislado que se aprovisiona y se destruye automáticamente) resulta particularmente adecuado para esta arquitectura: el framework no necesita mantener un proceso corriendo de forma continua a la espera de eventos, lo que reduce tanto el costo como la superficie expuesta del propio sistema de detección.

**Amazon GuardDuty** opera como capa de detección complementaria y ortogonal a los señuelos. Mientras que los honeypots generan alertas basadas en la certeza absoluta de que cualquier interacción con ellos es ilegítima (no hay tráfico "normal" posible sobre un activo que no cumple ninguna función productiva), GuardDuty aporta una capa de detección basada en el análisis de patrones de comportamiento, fuentes de inteligencia de amenazas (listas de IPs y dominios maliciosos conocidos) y aprendizaje automático sobre la totalidad del entorno, incluyendo los recursos productivos reales (sección 3.3.6). La combinación de ambas capas permite que HoneyAgent cubra tanto los escenarios donde el atacante interactúa directamente con un señuelo como aquellos donde su actividad sobre recursos productivos exhibe patrones anómalos detectables por análisis estadístico, sin que ambas capas compitan ni se superpongan: los honeypots aportan certeza con bajo volumen de señales, GuardDuty aporta cobertura amplia con un perfil de confianza probabilístico.

#### **4.5 Agente LLM: diseño, herramientas y explainability**

El componente central del framework (y su principal diferencial respecto de los sistemas de honeypots tradicionales analizados en el Capítulo 2) es un agente basado en el modelo Claude Sonnet 4.5, invocado a través de Amazon Bedrock con capacidad de uso de herramientas (tool use), alojado en la función Lambda que ejecuta el pipeline de detección. El

acceso al modelo se realiza mediante el cliente AnthropicBedrock del SDK de Anthropic, autenticado con las credenciales de AWS (rol de IAM) de la propia función Lambda, sin depender de una clave de API externa ni de una cuenta de facturación separada de la de AWS. Su diseño responde directamente al tercer objetivo específico del trabajo: desarrollar un agente que "analice eventos de seguridad y genere reportes con razonamiento explícito y trazable" (sección 1.4.2).

El agente recibe como entrada un resumen estructurado del evento capturado (qué identidad realizó qué operación, desde qué dirección IP, en qué momento y con qué resultado) junto con un conjunto de instrucciones de sistema que establecen su rol, el contexto regulatorio en el que opera (PCI DSS, normativa del BCRA) y el formato esperado del informe de salida. A partir de allí, el agente sigue un ciclo de razonamiento con uso de herramientas: puede decidir, de forma autónoma, invocar una o más herramientas externas para enriquecer su análisis antes de producir una conclusión.

La herramienta implementada en el prototipo, `lookup_ip_reputation`, consulta información pública de geolocalización y reputación sobre la dirección IP de origen del evento (país, proveedor de conectividad, organización asociada al rango de direcciones). Esta capacidad ilustra el principio de diseño general: el agente no se limita a clasificar el evento en función de los datos que ya posee, sino que puede ampliar activamente el contexto disponible antes de emitir su veredicto, de forma análoga a como lo haría un analista humano que consulta una base de inteligencia de amenazas durante una investigación. El conjunto de herramientas está diseñado para crecer de forma modular: agregar una nueva fuente de inteligencia (por ejemplo, una consulta a una lista de IPs maliciosas conocidas, o a una base de vulnerabilidades) implica únicamente declarar la herramienta y su esquema de entrada, sin modificar la lógica central del agente.

El aspecto que la sección 3.4.5 identificó como una brecha relevante en el estado del arte (la falta de explainability en los sistemas de detección automatizados basados en IA) constituye el eje central del diseño del agente. En lugar de producir únicamente una clasificación binaria o un puntaje de riesgo, el agente está instruido para exponer su proceso de razonamiento de forma explícita: qué observó en el evento, qué información adicional buscó y por qué, cómo esa información modificó (o no) su evaluación inicial, y qué lo llevó a la conclusión final. Esta traza de razonamiento queda documentada íntegramente en el informe de salida, lo que permite que un analista (incluso uno junior, con conocimiento limitado del contexto previo) pueda revisar, validar o cuestionar cada paso del análisis, en lugar de verse forzado a confiar ciegamente en un veredicto opaco. Esta propiedad es, además, la que habilita la alineación del framework con los requisitos de documentación y trazabilidad de PCI DSS v4.0.1 y de la Comunicación "A" 8398 del BCRA, que se desarrolla en la sección 4.9.

## 4.6 Outputs del sistema

La salida del agente adopta dos formas complementarias, ambas generadas a partir del mismo proceso de análisis y persistidas automáticamente por la función Lambda en un repositorio centralizado (un bucket de Amazon S3 dedicado a reportes).

El **informe en formato Markdown** está orientado a la lectura humana. Se estructura en secciones fijas (resumen del evento, enriquecimiento mediante herramientas externas, razonamiento, veredicto y recomendación) que permiten a un analista revisar rápidamente qué ocurrió y por qué el agente llegó a determinada conclusión, sin necesidad de interpretar el evento de CloudTrail original ni de reconstruir manualmente el contexto. Este formato es el que se proyectará, por ejemplo, durante una reunión de respuesta a incidentes o se adjuntará a un ticket de seguimiento.

El **registro estructurado en formato JSON** está orientado a la integración y a la trazabilidad a largo plazo. Contiene tanto el resumen del evento como el evento original de CloudTrail sin procesar y el informe completo generado por el agente, lo que permite que el reporte funcione como evidencia autocontenida (no depende de que el evento original siga disponible en el histórico de CloudTrail, cuya retención por defecto es acotada) y que pueda integrarse en herramientas de gestión de incidentes (SIEM, sistemas de tickets) mediante procesamiento automatizado.

Ambos artefactos se nombran y organizan siguiendo una convención que incorpora la marca temporal del evento y la identidad involucrada, y se conservan con una política de retención alineada a los plazos de trazabilidad que exigen los marcos regulatorios bajo análisis (sección 3.5). De esta manera, el conjunto de reportes generados por HoneyAgent constituye, en sí mismo, un repositorio de evidencia auditable: cada interacción con un activo señuelo (por definición, cada intento de compromiso detectado) queda documentada con el detalle suficiente para reconstruir, meses después, qué ocurrió, cómo fue detectado y qué razonamiento llevó a clasificarlo como se clasificó.

## 4.7 Recursos utilizados

La implementación del prototipo y, por extensión, del framework completo, se apoya en los siguientes recursos:

**Infraestructura cloud.** Una cuenta de Amazon Web Services con acceso a los servicios IAM, S3, Secrets Manager, RDS, CloudTrail, EventBridge, Lambda, GuardDuty y CloudWatch Logs. El volumen de uso que genera un entorno de señuelos (por diseño, mínimo, dado que un activo señuelo no debería recibir tráfico legítimo) se mantiene dentro de los niveles incluidos en el free tier de AWS, lo cual se desarrolla con mayor detalle en el análisis de costos (sección 4.9.2).

**Modelo de lenguaje.** Acceso al modelo Claude Sonnet 4.5 (familia de modelos de Anthropic) a través de Amazon Bedrock, invocado como inference profile de alcance multirregional, con soporte para tool use, utilizado como motor de razonamiento del agente. El acceso se autentica con las credenciales de AWS (rol de IAM) de la función Lambda, sin requerir una clave de API de Anthropic; el consumo se factura por token procesado dentro de la misma facturación de AWS, y el volumen que genera el framework (acotado al número de eventos efectivamente capturados por los señuelos) resulta igualmente reducido.

**Herramientas de infraestructura como código.** Terraform o AWS CDK para la traducción de `honeypots.yaml` en recursos despleables, y el SDK de AWS para Python (boto3) para los componentes de orquestación y para la propia función Lambda.

**Entorno de desarrollo y control de versiones.** Python 3.12 como lenguaje de implementación del agente y de las herramientas de despliegue, un repositorio de control de versiones para gestionar la evolución del framework y de su configuración declarativa, y un entorno de integración continua para automatizar el empaquetado y despliegue de la función Lambda, incluyendo sus dependencias externas, un aspecto que requiere atención particular dado que el entorno de ejecución de Lambda no incluye por defecto bibliotecas de terceros como el SDK de Anthropic (en su variante de integración con Amazon Bedrock).

**Recursos humanos.** El desarrollo fue realizado por el autor del presente trabajo, con la orientación de la dirección académica del Trabajo Final de Carrera, en el marco institucional descrito en la sección 1.3.

## 4.8 Cronograma de tareas

El desarrollo del proyecto se organizó en seis etapas secuenciales, con superposiciones parciales entre etapas consecutivas para permitir la incorporación temprana de hallazgos de una etapa en el diseño de la siguiente:

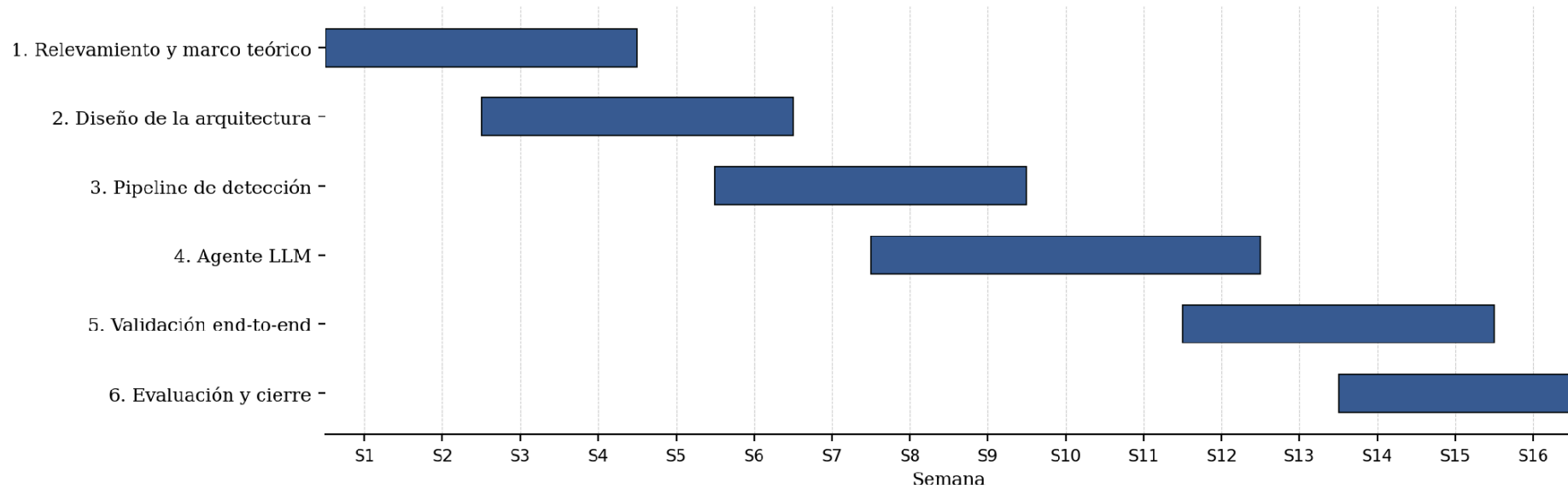
- 1. Relevamiento y marco teórico** (semanas 1 a 4): revisión del estado del arte, selección de fuentes, redacción de los Capítulos 2 y 3.
- 2. Diseño de la arquitectura** (semanas 3 a 6): definición de la capa de configuración, los tipos de señuelo, el pipeline de detección y la interfaz del agente; redacción de las secciones 4.1 a 4.3.
- 3. Implementación del pipeline de detección** (semanas 6 a 9): aprovisionamiento de la infraestructura base (CloudTrail, EventBridge, Lambda, IAM), desarrollo de las reglas de enrutamiento y validación del flujo desde el evento hasta la función, con datos sintéticos.
- 4. Implementación del agente LLM** (semanas 8 a 12): integración con el modelo Claude Sonnet 4.5 a través de Amazon Bedrock, diseño de las instrucciones de sistema, desarrollo e incorporación de herramientas externas, ajuste del formato de los informes de salida.
- 5. Validación end-to-end y escenarios de ataque simulados** (semanas 12 a 15): despliegue del prototipo de honeypot de identidad, ejecución de escenarios de reconocimiento simulados, medición de la tasa de detección y revisión de la calidad y trazabilidad de los informes generados por el agente, correspondiente al cuarto objetivo específico del trabajo.
- 6. Evaluación, documentación y cierre** (semanas 14 a 16): análisis de alineación con PCI DSS v4.0.1 y la Comunicación "A" 8398 del BCRA (quinto objetivo específico), elaboración de la matriz FODA y el análisis de costos, y preparación de la documentación final y del material de defensa.

#### 4.8.1 Diagrama de Gantt

La Figura 4.2 muestra la distribución temporal de las seis etapas descritas.

**Figura 4.2**

*Diagrama de Gantt del cronograma de tareas*



*Nota.* Elaboración propia. Las barras representan la duración en semanas de cada una de las seis etapas descritas en la sección 4.8, con las superposiciones parciales indicadas entre etapas consecutivas.

## 4.9 Evaluación del proyecto

### 4.9.1 Matriz FODA de la propuesta de intervención

**Tabla 3**

*Análisis FODA de la propuesta de intervención*

|                | Positivo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Negativo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Interno</b> | <ul style="list-style-type: none"> <li>• Arquitectura desacoplada en capas (configuración, infraestructura, análisis), que facilita el mantenimiento, la extensión a nuevos tipos de señuelo y la auditoría independiente de cada componente.</li> <li>• Pipeline de detección basado exclusivamente en servicios administrados de AWS, sin dependencias de agentes de terceros, lo que reduce tanto la superficie de ataque como el costo operativo.</li> <li>• Razonamiento explícito y trazable del agente LLM, que convierte al framework en una fuente de evidencia auditable y no solo de alertas, una propiedad directamente alineada con los requisitos documentales de PCI DSS y del BCRA.</li> <li>• Configuración declarativa que reduce significativamente la barrera operativa para equipos de seguridad con recursos acotados, un perfil frecuente entre las fintechs medianas y pequeñas del ecosistema argentino.</li> </ul> | <ul style="list-style-type: none"> <li>• Dependencia de un proveedor de modelos de lenguaje (Anthropic), cuyo modelo se accede a través de Amazon Bedrock: si bien esto evita depender de una clave de API o una cuenta de facturación externas a AWS, persiste una dependencia de disponibilidad, latencia y precio del modelo subyacente para el funcionamiento del componente central del framework.</li> <li>• Cobertura inicial acotada a cuatro tipos de señuelo orientados a identidad y datos; vectores de ataque que no involucren a esos activos (por ejemplo, explotación de vulnerabilidades en aplicaciones expuestas) quedan fuera del alcance directo de los honeypots, aunque sí podrían ser cubiertos por la capa complementaria de GuardDuty.</li> <li>• El prototipo validado de punta a punta cubre un único tipo de señuelo (identidad); la validación de los tres restantes queda como trabajo futuro inmediato (sección de Trabajos futuros).</li> </ul> |
| <b>Externo</b> | <ul style="list-style-type: none"> <li>• Posibilidad de ampliar el catálogo de herramientas del agente (listas de IPs maliciosas, bases de vulnerabilidades, sistemas de tickets) sin modificar su lógica central, lo que permite que el framework evolucione junto con el panorama de amenazas.</li> <li>• Posibilidad de extender el modelo a otros proveedores de nube (Azure, Google Cloud Platform) replicando el mismo patrón arquitectónico sobre sus respectivos servicios de auditoría y orquestación de eventos.</li> <li>• Creciente adopción de agentes de IA en operaciones de seguridad (sección 2.3), lo que sugiere una ventana de oportunidad para que frameworks como HoneyAgent se integren en flujos de trabajo de SOCs que ya están adoptando estas tecnologías.</li> </ul>                                                                                                                                             | <ul style="list-style-type: none"> <li>• Cambios en las políticas de precios o en las condiciones de acceso a la API del proveedor del modelo de lenguaje, que podrían afectar la viabilidad económica del framework a escala.</li> <li>• Evolución de las técnicas de reconocimiento de los atacantes hacia métodos capaces de distinguir activos señuelo de activos productivos, lo que requeriría una inversión continua en el realismo y la rotación de los honeypots desplegados.</li> <li>• Posibles modificaciones futuras al estándar PCI DSS o a la normativa del BCRA que alteren los requisitos de documentación y trazabilidad sobre los que se fundamenta la propuesta de valor regulatoria del framework.</li> </ul>                                                                                                                                                                                                                                              |

*Nota.* FODA = Fortalezas, Oportunidades, Debilidades, Amenazas. Elaboración propia a partir del análisis de la propuesta de intervención descrita en las secciones 4.1 a 4.8.

#### 4.9.2 Análisis de costos

El análisis de costos se realiza sobre dos dimensiones: el costo de la infraestructura cloud y el costo del componente de inteligencia artificial, dado que ambos responden a modelos de facturación estructuralmente distintos (capacidad reservada/por uso de AWS frente a consumo por token del modelo Claude Sonnet 4.5 invocado a través de Amazon Bedrock, dentro de la misma facturación de AWS).

**Infraestructura de AWS.** Los servicios involucrados en el pipeline de detección (CloudTrail, registro de eventos de administración; EventBridge, bus de eventos por defecto; Lambda, cómputo bajo demanda; y CloudWatch Logs, registro de ejecución) cuentan con niveles de uso gratuito que, dado el bajo volumen de eventos que por diseño genera un entorno de señuelos (ningún activo señuelo debería recibir tráfico legítimo), resultan suficientes tanto para un entorno de validación como para un despliegue productivo de tamaño moderado. El principal costo variable proviene del almacenamiento de los reportes generados (Amazon S3), que por su tamaño reducido (documentos de texto de pocos kilobytes) representa una fracción mínima de los niveles incluidos en el free tier. En conjunto, el costo de infraestructura del framework puede considerarse marginal frente al costo de las alternativas que reemplaza, en particular frente al costo de horas de analista dedicadas a la investigación manual de alertas o al licenciamiento de plataformas comerciales de detección y respuesta.

**Componente de inteligencia artificial.** El consumo del modelo Claude Sonnet 4.5, invocado a través de Amazon Bedrock, se factura por volumen de tokens procesados (de entrada y de salida) dentro de la facturación de AWS, sin una cuenta ni un proveedor de facturación separados, y depende directamente de tres factores: la longitud de las instrucciones de sistema, la cantidad de eventos efectivamente analizados y la cantidad de

iteraciones del ciclo de razonamiento con herramientas que cada análisis requiere. Dado que el framework está diseñado para que el agente se active únicamente ante interacciones con activos señuelo (eventos de baja frecuencia y alta relevancia, por oposición al volumen masivo de eventos que procesarán los sistemas de detección de anomalías sobre tráfico productivo), el costo total esperado se mantiene acotado y previsible, escalan linealmente con la cantidad de incidentes detectados y no con el volumen total de actividad del entorno.

**Comparación cualitativa.** Frente al costo de adoptar una plataforma comercial de Extended Detection and Response (XDR) o de Security Information and Event Management (SIEM) con capacidades de análisis asistido por IA (que suelen facturarse por volumen de eventos ingeridos o por usuario, independientemente de la tasa real de incidentes), el modelo de costos de HoneyAgent resulta particularmente favorable para organizaciones de tamaño pequeño o mediano: el framework solo "trabaja" (y por lo tanto solo genera costo variable relevante) cuando efectivamente detecta una interacción con un activo señuelo, lo cual, por construcción, equivale a decir que solo genera costo cuando hay una señal de compromiso que amerita atención. Esta característica responde directamente a la justificación de la propuesta planteada en la sección 1.2: ofrecer a organizaciones fintech con recursos acotados una vía de acceso a capacidades de detección y análisis asistido por IA que de otro modo estarían reservadas a organizaciones con presupuestos de seguridad significativamente mayores.

## 5 Factores externos condicionantes

El desarrollo y la eventual adopción de HoneyAgent están condicionados por un conjunto de factores externos al alcance de este trabajo, que no dependen de las decisiones de diseño del framework pero que inciden directamente sobre su viabilidad y su vigencia en el tiempo.

**Factor regulatorio.** El marco normativo bajo el cual opera HoneyAgent no es estático. Tanto la Comunicación “A” 8398 del BCRA (Banco Central de la República Argentina, 2026, p. 1) como el estándar PCI DSS v4.0.1 (PCI Security Standards Council, 2024) están sujetos a revisiones periódicas que podrían modificar los requisitos de documentación, trazabilidad y notificación de incidentes sobre los que se apoya la propuesta de valor regulatoria del framework (sección 3.5). Un cambio en los plazos de notificación o en los campos exigidos en un reporte de incidente, por ejemplo, requeriría ajustes en el formato de salida del agente que exceden el control del equipo de desarrollo.

**Factor tecnológico: dependencia de proveedores externos.** El funcionamiento de HoneyAgent depende de dos proveedores externos cuya evolución no está bajo control del framework: Amazon Web Services, que provee la infraestructura, los servicios de detección (CloudTrail, EventBridge, Lambda, GuardDuty) y, a través de Amazon Bedrock, el acceso al modelo de lenguaje; y Anthropic, que desarrolla y mantiene ese modelo (Claude Sonnet 4.5). Cambios en la disponibilidad, el precio o las condiciones de acceso al modelo a través de Bedrock, ya señalados como una debilidad en la sección 4.9.1, constituyen un factor externo que condiciona la sostenibilidad económica del framework a mediano plazo.

**Factor relativo a la evolución de las amenazas.** El panorama de amenazas sobre el cual se diseñó HoneyAgent no es estático. Verizon Business (2025, p. 10) e IBM Security (2025) documentan un crecimiento sostenido de los ataques basados en identidad, pero esta tendencia podría revertirse o desplazarse hacia otros vectores no cubiertos por los cuatro tipos

de señuelo implementados (sección 4.3), lo que reduciría la relevancia relativa del framework frente a nuevas superficies de ataque.

**Factor económico y organizacional.** La adopción de HoneyAgent por parte de fintechs argentinas está condicionada por las restricciones presupuestarias y de personal técnico especializado que caracterizan a las organizaciones medianas y pequeñas del ecosistema (sección 1.3). Aun cuando el framework fue diseñado para minimizar la barrera operativa mediante configuración declarativa (sección 4.2), su adopción efectiva depende de decisiones organizacionales, como la asignación de presupuesto y la priorización frente a otras inversiones de seguridad, que exceden el alcance de este trabajo.

## Conclusiones

El presente trabajo abordó el problema de la detección de ataques sobre identidades y credenciales en infraestructuras cloud de organizaciones fintech, un vector de ataque cuya relevancia fue documentada extensamente en la literatura (Verizon Business, 2025; IBM Security, 2025) y cuya atención resulta además una exigencia regulatoria explícita bajo PCI DSS v4.0.1 y la Comunicación “A” 8398 del BCRA.

Para atender este problema se diseñó e implementó HoneyAgent, un framework de honeypots distribuidos en AWS que incorpora un agente de inteligencia artificial con capacidad de uso de herramientas, orientado a mejorar la detección y el análisis de este tipo de ataques. Los cinco objetivos específicos planteados en la sección 1.4.2 permiten evaluar el grado de cumplimiento del objetivo general: se diseñó la arquitectura de honeypots de identidad configurable mediante YAML (objetivo 1); se implementó el pipeline de detección basado en CloudTrail, EventBridge y Lambda (objetivo 2); se desarrolló el agente LLM con capacidad de uso de herramientas y razonamiento trazable (objetivo 3); se validó la capacidad de detección del framework mediante un prototipo mínimo viable que implementa de punta a punta el honeypot de identidad (objetivo 4, con alcance acotado a un tipo de señuelo, según se detalla en la sección 4.9.1); y se evaluó la alineación del framework con los requisitos de documentación y trazabilidad de PCI DSS y del BCRA (objetivo 5).

El análisis del estado del arte (Capítulo 2) permitió identificar una brecha específica: la literatura existente aborda de forma fragmentada los honeypots cloud-native, los honeytokens de identidad y los agentes LLM aplicados a ciberseguridad, sin integrarlos en un framework único que opere sobre infraestructura real y considere el contexto regulatorio de las fintechs latinoamericanas. HoneyAgent se posiciona en esa intersección, aportando cuatro contribuciones concretas (sección 1.5): un framework configurable orientado a identidades AWS, un pipeline de detección nativo sin agentes de terceros, un agente LLM con

razonamiento explícito y trazable, y una alineación demostrable con los marcos regulatorios PCI DSS y BCRA.

La evaluación de la propuesta (sección 4.9) muestra que las principales fortalezas de HoneyAgent, su arquitectura desacoplada, su dependencia exclusiva de servicios administrados de AWS y su capacidad de generar evidencia auditable, se ven contrapesadas por debilidades igualmente concretas: la dependencia de un proveedor externo de modelos de lenguaje y una cobertura de validación limitada al honeypot de identidad. Estas limitaciones no invalidan la propuesta, pero delimitan con precisión el alcance de lo demostrado en este trabajo y el camino que queda pendiente, que se detalla en la sección de Trabajos futuros.

En conjunto, los resultados obtenidos permiten sostener que un framework de honeypots de identidad combinado con un agente LLM de razonamiento explícito constituye una vía viable para que organizaciones fintech con recursos acotados mejoren su capacidad de detección de ataques sobre identidades cloud, al tiempo que generan la evidencia documental que sus obligaciones regulatorias exigen.

## Trabajos futuros

Este trabajo dejó abiertas líneas de continuación concretas, que se desprenden directamente de las limitaciones señaladas en la sección 4.9.1 y en las Conclusiones.

**Validación end-to-end de los tres tipos de señuelos restantes.** El prototipo desarrollado válida de punta a punta únicamente el honeypot de identidad (IAM); los honeypots de almacenamiento (S3), credenciales (Secrets Manager) y base de datos (RDS) quedaron documentados como entradas deshabilitadas en honeypots.yaml (sección 4.2), pero no ejecutados en un entorno real. Una línea de trabajo inmediata es desplegar y validar estos tres tipos restantes bajo el mismo esquema de escenarios de ataque simulados utilizado para el honeypot de identidad, confirmando que el pipeline de detección y el agente LLM operan sin ajustes específicos por tipo de activo.

**Ampliación del catálogo de herramientas del agente.** El prototipo implementa una única herramienta (lookup\_ip\_reputation). El diseño modular del agente (sección 4.5) permite incorporar nuevas fuentes de inteligencia sin modificar su lógica central, por lo que una extensión natural es sumar herramientas de consulta a listas de IPs maliciosas conocidas, bases de datos de vulnerabilidades (CVE) y sistemas de tickets, ampliando la capacidad de enriquecimiento del razonamiento del agente ante cada evento.

**Extensión a otros proveedores de nube.** La arquitectura de HoneyAgent replica un patrón (señuelo, registro de auditoría, enrutamiento de eventos, agente LLM) que no es exclusivo de AWS. Extender el framework a Azure o Google Cloud Platform, sustituyendo CloudTrail, EventBridge y Lambda por sus servicios equivalentes, permitiría validar la generalidad del patrón arquitectónico propuesto.

**Reducción de la dependencia de un único proveedor de modelos de lenguaje.** La sección 4.9.1 identificó como debilidad la dependencia del modelo Claude Sonnet 4.5, accedido a través de Amazon Bedrock, para el funcionamiento del agente. Una línea de

trabajo futuro es evaluar la portabilidad del agente hacia modelos alternativos, incluyendo modelos de código abierto auto alojados, para mitigar el riesgo de disponibilidad y costo asociado a un proveedor externo único.

**Evaluación empírica con analistas de seguridad reales.** La explainability del agente fue diseñada conceptualmente (sección 3.4.5) y validada mediante la revisión de los informes generados, pero no fue sometida a una evaluación con analistas de seguridad en organizaciones fintech reales. Un estudio de usabilidad que mida el tiempo de revisión, la confianza depositada en el veredicto del agente y la tasa de coincidencia con el criterio de un analista humano permitiría cuantificar el valor práctico de la explainability propuesta.

## Anexo I

El prototipo mínimo viable desarrollado para validar la propuesta de intervención descrita en el Capítulo 4 se encuentra publicado en el repositorio HoneyAgent (<https://github.com/w7romerog/honeyagent>). Este anexo documenta su estructura de carpetas y la configuración declarativa central del framework.

**Estructura del repositorio.** El código se organiza en cinco áreas principales: agent/ (el agente LLM, su ciclo de tool use y sus herramientas), infra/ (definiciones de Terraform), scripts/ (empaquetado de la función Lambda y simulación de ataques), tests/, y el archivo honeypots.yaml en la raíz. El directorio reports/, donde se escriben los reportes generados en cada ejecución, no se versiona en el repositorio: se crea localmente al correr el agente.

```

honeypots.yaml
requirements.txt
pytest.ini
.env.example
README.md
LICENSE
agent/
  __init__.py
  agent.py                                (loop de tool use + instrucciones de
sistema)
  config.py                              (validador de esquema de
honeypots.yaml)
  lambda_handler.py                      (entry point de la función Lambda)
  report_generator.py                    (genera el reporte .md y el .json)
  tools/
    __init__.py
    base.py
    lookup_ip_reputation.py
    registry.py
infra/                                    (Terraform)
  main.tf
  variables.tf
  iam_identity.tf
  cloudtrail.tf
  eventbridge.tf
  lambda.tf
  s3_bucket.tf.disabled_stub
  secrets_manager.tf.disabled_stub
  rds_endpoint.tf.disabled_stub
scripts/
  build_lambda_package.sh
  simulate_attack.py
tests/
  test_agent.py

```

**Configuración declarativa (honeypots.yaml).** El archivo completo, tal como quedó implementado, se reproduce a continuación. Únicamente el honeypot de identidad (iam\_identity) tiene enabled: true; los otros tres tipos quedan documentados con enabled: false, conforme al alcance del MVP descrito en la sección 4.2.

```
# honeypots.yaml
# Configuración declarativa de honeypots. Único tipo implementado en el MVP:
# iam_identity. Los otros tres quedan documentados con enabled: false.
#
# Campos comunes:
#   type          → iam_identity | s3_bucket | secrets_manager | rds_endpoint
#   name          → identificador del honeypot
#   enabled       → si se aprovisiona en infra/
#   description   → contexto para el reporte del agente
#   config        → parámetros específicos del tipo

honeypots:

  # Honeypot de identidad (IAM) – implementado
  - type: iam_identity
    name: hp-billing-readonly
    enabled: true
    description: >
      Usuario IAM señuelo que simula una cuenta de automatización de solo
      lectura
      para facturación. Cualquier actividad con sus credenciales indica que
      fueron
      comprometidas.
    username: billing-readonly
    config:
      attach_policy_arn: "arn:aws:iam::aws:policy/ReadOnlyAccess"
      create_access_key: true
      tags:
        Purpose: honeypot
        HoneyAgent: "true"
    alerts:
      on_events:
        - "*"

  # S3 Bucket – documentado, no implementado
  - type: s3_bucket
    name: hp-fake-credentials-bucket
    enabled: false
    description: >
      Bucket con objetos que simulan credenciales y dumps de base de datos.
      Cualquier GetObject/ListBucket sería señal de reconocimiento o
      exfiltración.
    config:
      objects:
        - key: "credentials/aws_keys_backup.csv"
          content: "FAKE_ACCESS_KEY,FAKE_SECRET_KEY"
        - key: "db/prod_dump.sql.gz"
          content: "-- FAKE DATABASE DUMP --"
      block_public_access: true
      versioning: false
    alerts:
      on_events:
        - GetObject
        - PutObject
        - DeleteObject
        - ListBucket
```

```

# Secrets Manager – documentado, no implementado
- type: secrets_manager
  name: hp-prod-db-password
  enabled: false
  description: >
    Secret falso que simula la contraseña maestra de la base de datos de
    producción.
  config:
    secret_value: "FAKE_SUPER_SECRET_PASSWORD_DO_NOT_USE"
    description: "prod/database/master - DO NOT DELETE"
  alerts:
    on_events:
      - GetSecretValue
      - DescribeSecret
      - DeleteSecret
      - PutSecretValue

# RDS Endpoint – documentado, no implementado
- type: rds_endpoint
  name: hp-prod-db-endpoint
  enabled: false
  description: >
    Endpoint RDS señuelo que simula la base de datos de producción.
  config:
    engine: postgres
    instance_class: db.t3.micro
    db_name: "prod_customers"
  alerts:
    on_events:
      - DescribeDBInstances
      - ModifyDBInstance
      - "AuthenticateDatabaseAccess"

# Pipeline de detección
detection:
  audit_source: cloudtrail
  routing_engine: eventbridge
  executor: lambda
  complementary_layer:
    name: guardduty
    enabled: false

# Configuración del agente LLM
agent:
  model_env_var: AGENT_MODEL
  tools:
    - lookup_ip_reputation
  output_format: markdown+json
  reports_bucket: "honeyagent-reports-${account_id}"

```



## Anexo II

Durante la etapa de validación end-to-end del cronograma (sección 4.8, etapa 5), se ejecutó `scripts/simulate_attack.py` para invocar la operación `sts:GetCallerIdentity` con las credenciales del honeypot de identidad `billing-readonly`. Este anexo reproduce el reporte generado por el agente para ese evento real, tal como quedó persistido en el bucket de reportes (objeto `reports/iam_identity/2026/07/05/20260705T233433Z_billing-readonly`, en formato `.md` y `.json`).

**1. Reporte en Markdown.** Se reproduce a continuación el contenido íntegro del archivo generado por el agente:

```
# HoneyAgent — Reporte de Incidente (Honeypot de Identidad)

_Generado: 2026-07-05 23:47:45 UTC_

## 1. Resumen del evento

- **Honeypot**: hp-billing-readonly
- **Identidad invocadora**: arn:aws:iam::762528397495:user/billing-readonly
- **Servicio**: sts.amazonaws.com
- **Operación**: GetCallerIdentity
- **IP de origen**: 201.212.166.180
- **Momento del evento**: 2026-07-05T23:34:33Z
- **Región**: us-east-1
- **Resultado**: success
- **Parámetros relevantes**: `{}`

## 2. Enriquecimiento mediante herramientas externas

| Herramienta | Argumentos |
|---|---|
| lookup_ip_reputation | `{"ip": "201.212.166.180"}` |

## 3. Razonamiento

> Texto tal cual generado por el agente, sin resumir:

---

## Análisis de Amenaza — Honeypot IAM

**Evento ID interno**: hp-billing-readonly
**Timestamp**: 2026-07-05T23:34:33Z UTC
**Identidad comprometida**:
`arn:aws:iam::762528397495:user/billing-readonly`

---

### Qué observé
```

Se registró una invocación exitosa de la operación `GetCallerIdentity` del servicio AWS Security Token Service (STS) desde la dirección IP **\*\*201.212.166.180\*\*** utilizando las credenciales del usuario IAM señuelo `billing-readonly`.

La operación `GetCallerIdentity` es un método de reconocimiento típico en la cadena de ataque: permite al actor verificar que las credenciales robadas o filtradas son válidas, confirmando la identidad de la cuenta AWS, el ID de usuario y el ARN asociado. No requiere permisos especiales (cualquier identidad autenticada puede invocarla) y no genera cambios en la infraestructura, lo que la convierte en una técnica segura para el atacante durante la fase de enumeración inicial.

El evento tuvo éxito, confirmando que las credenciales del honeypot están activas y fueron utilizadas por un actor externo. Dado que esta identidad es un señuelo deliberado y **\*\*nunca debe autenticarse legítimamente\*\***, cualquier uso constituye evidencia directa de compromiso de credenciales.

---

### Qué busqué y por qué

Invocé `lookup\_ip\_reputation` para enriquecer la dirección IP de origen **\*\*201.212.166.180\*\*** y obtener:

- País de origen y proveedor de servicios de Internet (ISP).
- Organización asociada al rango IP.
- Sistema autónomo (AS) responsable del tráfico.

**\*\*Objetivo\*\***: Determinar si la IP proviene de un entorno compatible con el uso legítimo esperado de un usuario de billing (ej. rangos corporativos conocidos, proveedores cloud de la organización) o si corresponde a un actor externo no autorizado. Dado que el honeypot nunca debe ser usado, esta información ayuda a caracterizar al atacante y evaluar si se trata de un escaneo automatizado, un actor de amenaza avanzado o un acceso desde infraestructura residencial/comercial comprometida.

---

### Cómo cambió mi evaluación

El enriquecimiento arrojó los siguientes datos:

- **\*\*País\*\***: Argentina (AR)
- **\*\*ISP/Organización\*\***: Telecom Argentina S.A.
- **\*\*AS\*\***: AS7303 (Telecom Argentina S.A.)

Telecom Argentina es un proveedor de servicios de Internet comercial y residencial, no un proveedor de infraestructura cloud ni un rango corporativo asociado a nuestra organización. La IP corresponde probablemente a un cliente residencial o una conexión comercial de banda ancha.

**\*\*Implicancias\*\***:

1. **\*\*No hay compatibilidad con uso legítimo\*\***: Ningún proceso operativo autorizado debería invocar credenciales de AWS desde una conexión residencial en Argentina, especialmente tratándose de un usuario señuelo.
2. **\*\*Perfil del atacante\*\***: El origen sugiere un actor individual o grupo que opera desde infraestructura de ISP comercial, compatible con:
  - Credenciales filtradas en repositorios públicos (GitHub, Pastebin, foros).
  - Ataque de phishing exitoso seguido de reconocimiento.
  - Compra de credenciales en mercados clandestinos.
3. **\*\*Fase del ataque\*\***: La invocación de `GetCallerIdentity` indica que el

atacante está en fase de **\*\*reconocimiento inicial\*\***, validando credenciales antes de escalar a operaciones más intrusivas (ej. enumeración de recursos, exfiltración de datos, movimiento lateral).

**\*\*Conclusión\*\***: El evento confirma compromiso de credenciales. El origen argentino desde ISP comercial descarta hipótesis de escaneo automatizado desde botnets conocidas o infraestructura cloud, apuntando a un actor con intencionalidad específica que validó las credenciales de forma exitosa.

---

### ### Veredicto

**\*\*Severidad\*\***: **\*\*HIGH\*\***

**\*\*Justificación\*\***:

- **\*\*Confirmación de compromiso\*\***: El uso de credenciales del honeypot es evidencia inequívoca de filtración o robo.
- **\*\*Operación exitosa\*\***: El atacante confirmó validez de las credenciales y obtuvo información de la cuenta.
- **\*\*Fase de reconocimiento activa\*\***: `GetCallerIdentity` es el primer paso en una cadena de ataque; existe alta probabilidad de que el actor proceda con enumeración adicional (ListBuckets, DescribeInstances, GetAccountAuthorizationDetails) o acciones más dañinas.
- **\*\*Origen externo no autorizado\*\***: IP residencial/comercial argentina sin relación con infraestructura legítima de la organización.
- **\*\*Impacto potencial\*\***: Aunque el usuario señuelo `billing-readonly` tiene permisos limitados (presumiblemente de solo lectura en recursos de facturación), el atacante puede:
  - Obtener información sensible de costos, proyectos y arquitectura.
  - Usar esta información para ataques de ingeniería social o venta en mercados clandestinos.
  - Pivotar si existen misconfiguraciones de permisos o políticas inline no documentadas.

No se clasifica como **\*\*CRITICAL\*\*** porque:

- No hay evidencia (aún) de acceso a datos sensibles, modificación de recursos o movimiento lateral.
- El usuario honeypot está diseñado con permisos restringidos.
- El evento es de reconocimiento, no de exfiltración o sabotaje.

---

### ### Recomendación

**\*\*Acciones inmediatas (dentro de las próximas 2 horas)\*\***:

1. **\*\*Revocar credenciales comprometidas\*\***:

- Deshabilitar el usuario IAM `billing-readonly` (Account ID: 762528397495).
- Eliminar o rotar access keys asociadas.
- Invalidar sesiones activas con `aws iam delete-access-key` y revisión de tokens STS emitidos.

2. **\*\*Bloqueo de origen\*\***:

- Agregar la IP **\*\*201.212.166.180\*\*** y el rango /24 asociado (201.212.166.0/24) a la lista de denegación en WAF, Security Groups y NACLs perimetrales.
- Considerar bloqueo geográfico temporal de Argentina (AR) en recursos críticos si no hay operaciones legítimas en esa jurisdicción.

3. **\*\*Análisis de sesión completa\*\***:

- Revisar todos los eventos de CloudTrail asociados a `billing-readonly` en las últimas 24 horas (antes y después de 2026-07-05T23:34:33Z) para

identificar:

- Otras operaciones ejecutadas (enumeración, acceso a datos, modificaciones).
- IPs adicionales utilizadas por el atacante.
- Patrones de movimiento lateral o escalación de privilegios.

4. **\*\*Búsqueda de filtración de credenciales\*\*:**

- Ejecutar escaneo en GitHub, GitLab, Bitbucket y Pastebin con términos: `billing-readonly`, `AKIA\*` (prefijo de access keys AWS), dominios de correo corporativos.
- Revisar logs de autenticación de desarrolladores en las últimas 72 horas para identificar posibles vectores de compromiso (phishing, malware).

**\*\*Acciones de mediano plazo (próximas 48 horas)\*\*:**

5. **\*\*Auditoría de honeypots\*\*:**

- Verificar que todos los usuarios señuelo tienen alertas activas y políticas de denegación explícita (SCP o boundary policies).
- Confirmar que no existen permisos inline o attachments no documentados en `billing-readonly`.

6. **\*\*Documentación para cumplimiento\*\*:**

- Registrar este incidente en el sistema de gestión de incidentes de seguridad con clasificación según taxonomía interna.
- Preparar reporte para auditoría PCI DSS (Requisito 10: Seguimiento y monitoreo, Requisito 12.10.1: Plan de respuesta a incidentes).
- Notificar al oficial de seguridad de la información (si aplica bajo Comunicación A 8398 BCRA, considerando que el evento involucra credenciales de acceso a datos financieros/billing).

7. **\*\*Revisión de controles preventivos\*\*:**

- Evaluar implementación de AWS IAM Access Analyzer para detectar credenciales expuestas.
- Considerar rotación automática de credenciales y uso obligatorio de MFA para todos los usuarios IAM (incluyendo cuentas de servicio).

---

**\*\*Observaciones adicionales para trazabilidad regulatoria\*\*:**

- **\*\*PCI DSS v4.0.1\*\*:** Este evento es relevante para los requisitos 10.2.1 (registro de accesos a datos de tarjetahabientes) y 10.2.2 (acciones realizadas por individuos con privilegios). Aunque `billing-readonly` no accede directamente a CHD (Cardholder Data), el billing puede contener metadata sobre transacciones. La respuesta documentada cumple con 12.10.1 (respuesta a incidentes).
- **\*\*Comunicación A 8398 BCRA\*\*:** Artículo 2.3.1 (Gestión de incidentes) requiere clasificación, registro y tratamiento de eventos de ciberseguridad. Este incidente califica como "acceso no autorizado a credenciales de sistemas críticos" y debe reportarse internamente según matriz de escalamiento de la entidad. Si el usuario tenía acceso real a datos de clientes o sistemas core bancarios (más allá de billing), considerar notificación a BCRA según criticidad y tiempo de exposición.

---

**\*\*Fin del análisis.\*\***

**## 4. Veredicto**

**\*\*Severidad\*\*:** **\*\*HIGH\*\***

**\*\*Justificación\*\*:**

- **\*\*Confirmación de compromiso\*\*:** El uso de credenciales del honeypot es evidencia inequívoca de filtración o robo.
- **\*\*Operación exitosa\*\*:** El atacante confirmó validez de las credenciales y obtuvo información de la cuenta.
- **\*\*Fase de reconocimiento activa\*\*:** `GetCallerIdentity` es el primer paso en una cadena de ataque; existe alta probabilidad de que el actor proceda con enumeración adicional (ListBuckets, DescribeInstances, GetAccountAuthorizationDetails) o acciones más dañinas.
- **\*\*Origen externo no autorizado\*\*:** IP residencial/comercial argentina sin relación con infraestructura legítima de la organización.
- **\*\*Impacto potencial\*\*:** Aunque el usuario señuelo `billing-readonly` tiene permisos limitados (presumiblemente de solo lectura en recursos de facturación), el atacante puede:
  - Obtener información sensible de costos, proyectos y arquitectura.
  - Usar esta información para ataques de ingeniería social o venta en mercados clandestinos.
  - Pivotar si existen misconfiguraciones de permisos o políticas inline no documentadas.

No se clasifica como **\*\*CRITICAL\*\*** porque:

- No hay evidencia (aún) de acceso a datos sensibles, modificación de recursos o movimiento lateral.
- El usuario honeypot está diseñado con permisos restringidos.
- El evento es de reconocimiento, no de exfiltración o sabotaje.

---

**## 5. Recomendación****\*\*Acciones inmediatas (dentro de las próximas 2 horas)\*\*:****1. \*\*Revocar credenciales comprometidas\*\*:**

- Deshabilitar el usuario IAM `billing-readonly` (Account ID: 762528397495).
- Eliminar o rotar access keys asociadas.
- Invalidar sesiones activas con `aws iam delete-access-key` y revisión de tokens STS emitidos.

**2. \*\*Bloqueo de origen\*\*:**

- Agregar la IP **\*\*201.212.166.180\*\*** y el rango /24 asociado (201.212.166.0/24) a la lista de denegación en WAF, Security Groups y NACLs perimetrales.
- Considerar bloqueo geográfico temporal de Argentina (AR) en recursos críticos si no hay operaciones legítimas en esa jurisdicción.

**3. \*\*Análisis de sesión completa\*\*:**

- Revisar todos los eventos de CloudTrail asociados a `billing-readonly` en las últimas 24 horas (antes y después de 2026-07-05T23:34:33Z) para identificar:
  - Otras operaciones ejecutadas (enumeración, acceso a datos, modificaciones).
  - IPs adicionales utilizadas por el atacante.
  - Patrones de movimiento lateral o escalación de privilegios.

**4. \*\*Búsqueda de filtración de credenciales\*\*:**

- Ejecutar escaneo en GitHub, GitLab, Bitbucket y Pastebin con términos: `billing-readonly`, `AKIA\*` (prefijo de access keys AWS), dominios de correo corporativos.
- Revisar logs de autenticación de desarrolladores en las últimas 72 horas para identificar posibles vectores de compromiso (phishing, malware).

**\*\*Acciones de mediano plazo (próximas 48 horas)\*\*:**

```

5. **Auditoría de honeypots**:
    - Verificar que todos los usuarios señuelo tienen alertas activas y políticas de denegación explícita (SCP o boundary policies).
    - Confirmar que no existen permisos inline o attachments no documentados en `billing-readonly`.

6. **Documentación para cumplimiento**:
    - Registrar este incidente en el sistema de gestión de incidentes de seguridad con clasificación según taxonomía interna.
    - Preparar reporte para auditoría PCI DSS (Requisito 10: Seguimiento y monitoreo, Requisito 12.10.1: Plan de respuesta a incidentes).
    - Notificar al oficial de seguridad de la información (si aplica bajo Comunicación A 8398 BCRA, considerando que el evento involucra credenciales de acceso a datos financieros/billing).

7. **Revisión de controles preventivos**:
    - Evaluar implementación de AWS IAM Access Analyzer para detectar credenciales expuestas.
    - Considerar rotación automática de credenciales y uso obligatorio de MFA para todos los usuarios IAM (incluyendo cuentas de servicio).

---

**Observaciones adicionales para trazabilidad regulatoria**:

- **PCI DSS v4.0.1**: Este evento es relevante para los requisitos 10.2.1 (registro de accesos a datos de tarjetahabientes) y 10.2.2 (acciones realizadas por individuos con privilegios). Aunque `billing-readonly` no accede directamente a CHD (Cardholder Data), el billing puede contener metadata sobre transacciones. La respuesta documentada cumple con 12.10.1 (respuesta a incidentes).

- **Comunicación A 8398 BCRA**: Artículo 2.3.1 (Gestión de incidentes) requiere clasificación, registro y tratamiento de eventos de ciberseguridad. Este incidente califica como "acceso no autorizado a credenciales de sistemas críticos" y debe reportarse internamente según matriz de escalamiento de la entidad. Si el usuario tenía acceso real a datos de clientes o sistemas core bancarios (más allá de billing), considerar notificación a BCRA según criticidad y tiempo de exposición.

---

**Fin del análisis.**

```

**2. Registro JSON.** El registro estructurado correspondiente, con el resumen del evento y el evento crudo de CloudTrail, se reproduce a continuación. El campo `agent_report.final_analysis` se abrevia por brevedad, al ser idéntico al texto ya reproducido en el punto 1 de este anexo; ningún otro campo fue modificado respecto del archivo real.

```

{
  "report_metadata": {
    "generated_at": "2026-07-05T23:47:45.133704+00:00",
    "framework": "HoneyAgent - MVP Cap. 4 (honeypot de identidad)",
    "report_format": "json"
  },
  "event_summary": {
    "honeypot_name": "hp-billing-readonly",
    "event_source": "sts.amazonaws.com",
    "event_name": "GetCallerIdentity",
    "source_ip": "201.212.166.180",

```

```

    "aws_identity": "arn:aws:iam::762528397495:user/billing-readonly",
    "event_time": "2026-07-05T23:34:33Z",
    "aws_region": "us-east-1",
    "parameters": {},
    "result": "success"
  },
  "raw_cloudtrail_event": {
    "eventVersion": "1.11",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "AIDA3DCR37C34IN5R2FHC",
      "arn": "arn:aws:iam::762528397495:user/billing-readonly",
      "accountId": "762528397495",
      "accessKeyId": "AKIA3DCR37C3ZXLF24L",
      "userName": "billing-readonly"
    },
    "eventTime": "2026-07-05T23:34:33Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "GetCallerIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "201.212.166.180",
    "userAgent": "Boto3/1.43.40 md/Botocore#1.43.40 ua/2.1 os/windows#11
md/arch#amd64 lang/python#3.12.5 md/pyimpl#CPython m/b,Z,D,e
cfg/retry-mode#legacy Botocore/1.43.40",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData": {
      "ExtendedRequestId":
"MTp1cyl1YXN0LTE6UzoxNzgzMjk0NDczNDMwOlI6U01nU3FmWHO=",
      "RequestDetails": {
        "endpointType": "regional",
        "awsServingRegion": "us-east-1"
      }
    },
    "requestID": "528c6f6b-cc36-446a-be99-c110cbc50b3b",
    "eventID": "5dd09a4c-26f6-4d64-9fad-3b93c4ce20eb",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "762528397495",
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_128_GCM_SHA256",
      "clientProvidedHostHeader": "sts.us-east-1.amazonaws.com"
    }
  },
  "agent_report": {
    "success": true,
    "iterations": 2,
    "tools_called": [
      {
        "tool": "lookup_ip_reputation",
        "input": {
          "ip": "201.212.166.180"
        }
      }
    ],
    "final_analysis": "(idéntico al texto reproducido en las secciones 3 a 5
del reporte en Markdown del punto 1 de este anexo; se omite aquí por
brevedad, sin alterar ningún otro campo del registro real)"
  }
}

```

**3. Matriz de trazabilidad regulatoria.** La siguiente tabla vincula los requisitos normativos ya analizados en la sección 3.5 con los componentes específicos de HoneyAgent

que les dan cumplimiento, usando como evidencia el caso reproducido en los puntos 1 y 2 de este anexo.

**Tabla 4**

*Matriz de trazabilidad entre requisitos regulatorios y componentes de HoneyAgent*

| Marco / Requisito                        | Componente de HoneyAgent                                   | Evidencia en el caso reproducido                                                                                |
|------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| PCI DSS v4.0.1, Requerimiento 10         | AWS CloudTrail (sección 4.4)                               | Evento crudo de CloudTrail capturado automáticamente (Anexo II, punto 2).                                       |
| PCI DSS v4.0.1, Sub-requerimiento 10.2.1 | Regla de Amazon EventBridge + función Lambda (sección 4.4) | El evento se enrutó a la Lambda sin intervención manual.                                                        |
| PCI DSS v4.0.1, Sub-requerimiento 10.4   | Agente LLM (sección 4.5)                                   | El reporte se generó de forma automática 13 minutos después del evento (entre las 23:34:33 y las 23:47:45 UTC). |
| PCI DSS v4.0.1, Requerimiento 11.5       | Honeypot de identidad (sección 4.3)                        | Activación de las credenciales de billing-readonly documentada como evidencia de intrusión.                     |
| PCI DSS v4.0.1, Apéndice A3.5            | Reporte con razonamiento explícito (secciones 4.5 y 4.6)   | Las secciones “Qué observé” y “Cómo cambió mi evaluación” del reporte documentan el patrón identificado.        |
| BCRA Comunicación “A” 8398, Sección 10.2 | Pipeline completo (detección, análisis y reporte)          | El reporte documenta las etapas del ciclo de gestión del incidente de forma auditable.                          |
| BCRA Comunicación “A” 8398, Sección 10.4 | Reportes persistidos en Amazon S3 (sección 4.6)            | Objeto reports/iam_identity/2026/07/05/20260705T233433Z_billing-readonly en S3, disponible para auditoría.      |

*Nota.* PCI DSS = Payment Card Industry Data Security Standard, v4.0.1 (PCI Security Standards Council, 2024). BCRA = Banco Central de la República Argentina, Comunicación “A” 8398 (Banco Central de la República Argentina, 2026). Elaboración propia a partir del caso reproducido en este anexo y de los requisitos analizados en la sección 3.5.

## Acrónimos

**API.** Application Programming Interface (interfaz de programación de aplicaciones).

**ATT&CK.** Adversarial Tactics, Techniques, and Common Knowledge (marco de MITRE).

**AWS.** Amazon Web Services.

**BCRA.** Banco Central de la República Argentina.

**CDE.** Cardholder Data Environment (entorno de datos de titulares de tarjeta).

**CDK.** Cloud Development Kit.

**CHD.** Cardholder Data (datos de titulares de tarjeta).

**CVE.** Common Vulnerabilities and Exposures.

**DDoS.** Distributed Denial of Service (denegación de servicio distribuido).

**DNS.** Domain Name System.

**FODA.** Fortalezas, Oportunidades, Debilidades, Amenazas.

**HIH.** High-Interaction Honeypots (honeypots de alta interacción).

**HTTP.** Hypertext Transfer Protocol.

**IA.** Inteligencia Artificial.

**IAM.** Identity and Access Management.

**IDS.** Intrusion Detection Systems (sistemas de detección de intrusiones).

**IR.** Incident Response (respuesta ante incidentes).

**JSON.** JavaScript Object Notation.

**LIH.** Low-Interaction Honeypots (honeypots de baja interacción).

**LLM.** Large Language Model (modelo de lenguaje de gran escala).

**MCP.** Model Context Protocol.

**MIH.** Medium-Interaction Honeypots (honeypots de interacción media).

**MTTB.** Mean Time to Block (tiempo medio de bloqueo).

**OWASP.** Open Web Application Security Project.

**PCI DSS.** Payment Card Industry Data Security Standard.

**PCI SSC.** PCI Security Standards Council.

**PSP.** Proveedores de Servicios de Pago.

**RDS.** Relational Database Service.

**S3.** Simple Storage Service.

**SDK.** Software Development Kit.

**SIEM.** Security Information and Event Management.

**SM.** Secrets Manager.

**SOC.** Security Operations Center (centro de operaciones de seguridad).

**SSH.** Secure Shell.

**VPC.** Virtual Private Cloud.

**XDR.** Extended Detection and Response.

**YAML.** YAML Ain't Markup Language.

## Referencias

Amazon Web Services. (2026a). *Amazon EventBridge user guide*.

<https://docs.aws.amazon.com/eventbridge/latest/userguide/>

Amazon Web Services. (2026b). *Amazon GuardDuty user guide*.

<https://docs.aws.amazon.com/guarddduty/latest/ug/>

Amazon Web Services. (2026c). *AWS CloudTrail user guide*.

<https://docs.aws.amazon.com/awsccloudtrail/latest/userguide/>

Amazon Web Services. (2026d). *AWS Lambda developer guide*.

<https://docs.aws.amazon.com/lambda/latest/dg/>

Banco Central de la República Argentina. (2026). *Comunicación A 8398: Modificaciones a Requisitos Mínimos para la Gestión y Control de Riesgos de Tecnología e Información de Seguridad*.

<https://www.bcra.gob.ar/SistemasFinancierosYdePagos/CiberSeguridad-y-Ciberresiliencia.asp>

Bridges, R. A., Mitchell, T. R., Muñoz, M., & Henriksson, T. (2025). SoK: Honeypots & LLMs, more than the sum of their parts? *arXiv:2510.25939*.

<https://arxiv.org/abs/2510.25939>

Chauhan, M., & Shiaeles, S. (2023). An analysis of cloud security frameworks, problems and proposed solutions. *Network*, 3(3), 422–450. <https://doi.org/10.3390/network3030018>

Chin, D. M., Isfaq, B., & Enoch, S. Y. (2025). A practical honeypot-based threat intelligence framework for cyber defence in the cloud. *arXiv:2512.05321*.

<https://arxiv.org/abs/2512.05321>

Devi Priya, V. S., & Chakkaravarthy, S. S. (2023). Containerized cloud-based honeypot deception for tracking attackers. *Scientific Reports*, 13, 1437.

<https://doi.org/10.1038/s41598-023-28613-0>

IBM Security. (2025). *X-Force Threat Intelligence Index 2025*.

<https://www.ibm.com/think/x-force/x-force-threat-intelligence-index-2025-attackers-s-teal-sell-user-identities>

Izhikevich, L., Tran, M., Kallitsis, M., Fass, A., & Durumeric, Z. (2023). Cloud watching:

Understanding attacks against cloud-hosted services. *Proceedings of the 2023 ACM on Internet Measurement Conference*, 313–327.

<https://doi.org/10.1145/3618257.3624818>

Lazer, S. J., Aryal, K., Gupta, M., & Bertino, E. (2026). A survey of agentic AI and

cybersecurity: Challenges, opportunities and use-case prototypes. arXiv:2601.05293.

<https://arxiv.org/abs/2601.05293>

Machmeier, S. (2024). Honeypot implementation in a cloud environment. *arXiv:2301.00710*.

<https://arxiv.org/abs/2301.00710>

Morić, Z., Dakić, V., & Regvart, D. (2025). Advancing cybersecurity with honeypots and deception strategies. *Informatics*, 12(1), 14.

<https://doi.org/10.3390/informatics12010014>

OWASP Foundation. (2025). *OWASP Top 10 2025*. <https://owasp.org/Top10/2025/en/>

PCI Security Standards Council. (2024). *PCI DSS v4.0.1*.

[https://www.pcisecuritystandards.org/document\\_library/](https://www.pcisecuritystandards.org/document_library/)

Rabzelj, M., & Sedlar, U. (2025). Beyond the leak: Analyzing the real-world exploitation of stolen credentials using honeypots. *Sensors*, 25(12), Article 3676.

<https://doi.org/10.3390/s25123676>

Reti, D., Becker, N., Angeli, T., Chattopadhyay, A., Schneider, D., Vollmer, S., & Schotten,

H. D. (2024). Act as a honeytokens generator! An investigation into honeytokens

generation with large language models. In *Proceedings of the 11th ACM Workshop on*

- Adaptive and Autonomous Cyber Defense (AACD '24)* (pp. 1–11). Association for Computing Machinery. <https://doi.org/10.1145/3689935.3690394>
- Srinivas, S., Kirk, B., Zendejas, J., Espino, M., Boskovich, M., Bari, A., Dajani, K., & Alzahrani, N. (2025). AI-augmented SOC: A survey of LLMs and agents for security automation. *Journal of Cybersecurity and Privacy*, 5(4), Article 95. <https://doi.org/10.3390/jcp5040095>
- Verizon Business. (2025). *2025 Data Breach Investigations Report*. <https://www.verizon.com/business/resources/reports/dbir/>
- Vinay, V. (2025). The evolution of agentic AI in cybersecurity: From single LLM reasoners to multi-agent systems and autonomous pipelines. arXiv:2512.06659. <https://arxiv.org/abs/2512.06659>
- Yu, T., Xin, Y., & Zhang, C. (2024). HoneyFactory: Container-based comprehensive cyber deception honeynet architecture. *Electronics*, 13(2), 361. <https://doi.org/10.3390/electronics13020361>