

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**



**Comparación entre detección basada en reglas y *Machine Learning* para la
identificación de cadenas de procesos sospechosas en terminales (*endpoints*) Linux**

Daniel Esteban López Suárez

Universidad Ean
Facultad de Ingeniería
Programa Ingeniería de Sistemas
Bogotá, Colombia
07/04/2026

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

**Comparación entre detección basada en reglas y *Machine Learning* para la
identificación de cadenas de procesos sospechosas en *endpoints* Linux**

Daniel Esteban López Suárez

Trabajo de grado presentado como requisito para optar al título de:

Ingeniero de sistemas

Directora:

Marie José Chéry Leal

Universidad Ean

Facultad de ingeniería

Programa ingeniería de sistemas

Bogotá, Colombia

07/04/2026

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Resumen

El artículo compara dos enfoques de detección de actividad sospechosa en un dispositivo final Linux: un esquema basado en reglas asociado a una plataforma de gestión de eventos e información de seguridad y un detector de anomalías basado en aprendizaje automático. El problema se centra en relaciones padre-hijo entre procesos y en secuencias de descarga y ejecución inmediata, porque estos patrones pueden aparecer tanto en actividades legítimas como en fases tempranas de compromiso. La investigación se desarrolla mediante un laboratorio controlado con *auditd*, captura de registros de auditoría, construcción de variables de comportamiento y comparación de métricas como tasa de detección, falsos positivos, precisión, sensibilidad, puntaje F1 y tiempo de detección. El aporte esperado es una evaluación reproducible que permita identificar las condiciones en las que cada estrategia resulta más conveniente para apoyar decisiones de monitoreo en sistemas Linux.

Palabras clave: aprendizaje automático, detección de anomalías, dispositivo final, Linux, seguridad informática.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Abstract

This article compares two approaches for detecting suspicious activity on a Linux endpoint: a rule-based scheme associated with a security information and event management platform and a machine-learning-based anomaly detector. The problem focuses on parent-child process relationships and download-then-execute sequences, because these patterns may appear in both legitimate activity and early compromise stages. The study uses a controlled laboratory with *auditd*, audit log collection, behavioral feature construction, and performance comparison through detection rate, false positives, precision, recall, F1 score, and detection time. The expected contribution is a reproducible evaluation to identify the conditions under which each strategy is more suitable for Linux system monitoring.

Keywords: anomaly detection, endpoint, Linux, machine learning, cybersecurity.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Introducción

Un dispositivo final, o *endpoint*, es un equipo de usuario o servidor que participa en una red y ejecuta procesos, aplicaciones y servicios desde los cuales se consumen o exponen recursos institucionales. En los últimos años, la seguridad de estos dispositivos se ha consolidado como una preocupación central, debido a que muchas intrusiones no comienzan con programas maliciosos visibles, sino con el abuso de cuentas válidas, herramientas administrativas y procesos legítimos del sistema. El informe X-Force Threat Intelligence Index 2024 reporta que el uso de credenciales robadas para acceder a cuentas válidas aumentó 71 % y representó el 30 % de los incidentes atendidos por IBM X-Force en 2023 (IBM Security, 2024).

Este fenómeno se relaciona con los ataques de uso de herramientas legítimas del sistema, conocidos en la literatura como *living-off-the-land* (LotL). En este tipo de ataque, el adversario usa utilidades nativas o ya instaladas para ejecutar acciones maliciosas que se mezclan con la operación normal del sistema. La guía conjunta de la Agencia de Ciberseguridad e Infraestructura de Estados Unidos, CISA por sus siglas en inglés, y sus aliados advierte que estas técnicas dificultan la detección porque reducen la necesidad de introducir herramientas externas y permiten que la actividad maliciosa se confunda con tareas administrativas habituales (CISA et al., 2024).

Los enfoques tradicionales de detección basados en reglas, implementados en sistemas de detección de intrusiones en host, HIDS por las siglas de *Host-based Intrusion Detection Systems*, y en plataformas SIEM, por las siglas de *Security Information and Event Management*, continúan siendo útiles por su interpretabilidad y por su capacidad para identificar patrones de ataque previamente documentados. Sin embargo, la literatura señala que los enfoques basados en firmas

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

o reglas pueden perder efectividad cuando el atacante modifica ligeramente la técnica, cuando el comportamiento legítimo presenta variaciones o cuando el sistema genera demasiado ruido operativo (Scarfone & Mell, 2007; Garcia-Teodoro et al., 2009; Sommer & Paxson, 2010).

En sistemas Linux, una fuente relevante para observar ese comportamiento interno es *auditd*, el componente de espacio de usuario del sistema de auditoría de Linux responsable de escribir registros de auditoría en disco y de operar junto con utilidades como *auditctl*, *ausearch* y *aureport* (Kerrisk, 2026). Esta telemetría permite reconstruir eventos de ejecución, argumentos, marcas de tiempo y relaciones entre procesos padre e hijo, denominadas *parent-child* en inglés, elementos que son útiles para detectar secuencias anómalas sin depender únicamente de la presencia de un archivo malicioso conocido.

En este contexto, el problema de investigación se delimita al análisis de dos comportamientos críticos en la seguridad de dispositivos finales Linux: las relaciones padre-hijo inusuales entre procesos y las secuencias de descarga y ejecución inmediata. Estos patrones pueden aparecer en fases tempranas de compromiso y, al mismo tiempo, confundirse con actividades legítimas según el contexto operativo. En un laboratorio local con un dispositivo Linux instrumentado mediante *auditd*, no está claramente establecido qué enfoque ofrece un mejor equilibrio entre capacidad de detección, control de falsos positivos y rapidez de respuesta.

La pregunta que guía esta investigación es: ¿qué diferencias de desempeño existen, en términos de tasa de detección, tasa de falsos positivos y tiempo de detección, entre un enfoque basado en reglas tipo SIEM y un detector de anomalías basado en aprendizaje automático para identificar relaciones padre-hijo inusuales y secuencias de descarga y ejecución en un dispositivo

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

final Linux instrumentado con *auditd* en un laboratorio local? Como hipótesis de trabajo se plantea que el detector de anomalías basado en aprendizaje automático puede ofrecer un mejor balance global entre detección y control de falsos positivos, especialmente en escenarios donde el comportamiento legítimo del sistema presenta mayor variabilidad.

El objetivo general del estudio es comparar el desempeño de un enfoque basado en reglas tipo SIEM y de un detector de anomalías basado en aprendizaje automático para identificar relaciones padre-hijo inusuales y secuencias de descarga y ejecución en un dispositivo final Linux instrumentado con *auditd*. Para alcanzarlo, la investigación desarrolla un laboratorio local de experimentación, construye escenarios reproducibles de actividad normal y sospechosa, implementa reglas de detección, entrena un modelo de anomalías con variables extraídas de los registros de auditoría y evalúa ambos enfoques con métricas homogéneas de desempeño.

El prototipo entregable se compone del laboratorio Linux instrumentado, las reglas de auditoría y detección, el módulo de preprocesamiento de registros, el detector basado en aprendizaje automático y el reporte comparativo de resultados. Las fases de desarrollo se articulan de manera secuencial: preparación del entorno, generación de escenarios, recolección y normalización de registros, implementación de los detectores, ejecución de pruebas repetidas y análisis de resultados. Esta organización permite vincular el alcance técnico con los objetivos revisados en la tutoría.

El aporte esperado consiste en proporcionar evidencia experimental reproducible sobre el desempeño relativo de ambos enfoques. Más que declarar un método superior en todos los casos, el trabajo analiza las condiciones en las que cada estrategia resulta más efectiva y los compromisos

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

existentes entre detección, ruido operativo e interpretabilidad. Con ello se busca apoyar el diseño de estrategias de monitoreo más realistas para sistemas Linux.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Marco teórico

Con base en la literatura revisada, el marco teórico se organiza en torno a la seguridad de dispositivos finales, los ataques de uso de herramientas legítimas del sistema, la detección basada en reglas, la detección de anomalías con aprendizaje automático, el uso de *auditd* como fuente de telemetría y las métricas de evaluación apoyadas en MITRE ATT&CK. Esta organización relaciona los conceptos del problema con el diseño experimental propuesto.

Seguridad en *endpoints* y ataques *living-off-the-land*

Los dispositivos finales cumplen un rol operativo y de seguridad dentro de una red porque concentran sesiones de usuario, ejecución de procesos, archivos de trabajo, credenciales, conexiones salientes y servicios locales. Por esa razón, un compromiso en este nivel puede permitir persistencia, movimiento lateral o ejecución de comandos sin que necesariamente exista un programa malicioso nuevo. Desde la perspectiva defensiva, observar el comportamiento del sistema operativo resulta tan importante como inspeccionar archivos o tráfico de red.

Los ataques LotL aprovechan esta característica porque usan herramientas legítimas, intérpretes de comandos y utilidades administrativas para realizar acciones maliciosas. La Agencia de Ciberseguridad e Infraestructura de Estados Unidos, CISA por sus siglas en inglés, y sus aliados recomiendan que la detección de estas técnicas se apoye en telemetría detallada y en análisis de comportamiento, ya que los indicadores aislados pueden ser insuficientes cuando la actividad se camufla dentro del funcionamiento normal del sistema.

Detección basada en reglas en HIDS y SIEM

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Los HIDS monitorean eventos del propio host, mientras que las plataformas SIEM centralizan, correlacionan y presentan eventos de seguridad provenientes de distintas fuentes. En ambos casos, las reglas permiten identificar patrones conocidos mediante condiciones explícitas, por ejemplo una combinación de proceso padre, comando ejecutado, ruta del archivo y conexión de red. Esta aproximación facilita la explicación de las alertas y reduce la opacidad del sistema de detección (Scarfone & Mell, 2007).

La principal limitación del enfoque basado en reglas es su dependencia de patrones definidos previamente. Cuando la técnica cambia, cuando los argumentos de ejecución varían o cuando una acción legítima se parece a un indicador malicioso, la regla puede fallar o producir falsos positivos. Esta tensión ha sido documentada en revisiones sobre detección de intrusiones basada en anomalías y en análisis críticos sobre el uso operativo del aprendizaje automático en seguridad (Garcia-Teodoro et al., 2009; Sommer & Paxson, 2010).

Detección de anomalías con Machine Learning

La detección de anomalías identifica observaciones que se desvían del comportamiento considerado normal. En seguridad informática, esta estrategia permite buscar actividades no vistas durante la definición de reglas, aunque exige una representación adecuada del comportamiento y un control cuidadoso de falsos positivos (Chandola et al., 2009). El aprendizaje automático, o *Machine Learning* (ML), aporta algoritmos capaces de aprender patrones a partir de datos, pero no elimina la necesidad de validar la calidad de las variables, el equilibrio de clases y la pertinencia de las métricas.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Entre los algoritmos usados en este campo se encuentran Isolation Forest, los autocodificadores, conocidos como *autoencoders*, y los métodos de agrupamiento, conocidos como *clustering*. Isolation Forest identifica valores atípicos mediante particiones aleatorias del espacio de características (Liu et al., 2008). Los autocodificadores aprenden una representación comprimida del comportamiento normal y señalan como anómalas las instancias con alto error de reconstrucción. Los métodos de agrupamiento separan observaciones por similitud y permiten detectar registros que quedan lejos de los grupos dominantes.

Auditd como fuente de telemetría en Linux

auditd es el demonio de auditoría de Linux encargado de escribir registros de auditoría y de operar junto con herramientas de consulta y configuración como *auditctl*, *ausearch* y *aureport* (Kerrisk, 2026). A través de reglas de auditoría, el sistema puede capturar eventos de ejecución de procesos, llamadas al sistema, accesos a archivos y marcas de tiempo. Por sí mismo, *auditd* no es un detector, sino una fuente de datos que requiere procesamiento posterior.

Para el presente estudio, esta fuente resulta adecuada porque permite observar relaciones entre procesos, comandos ejecutados, rutas, usuarios y eventos temporales. La utilidad analítica depende de convertir los registros, no *logs* sin procesar, en variables comparables, como frecuencia de ejecución, profundidad de cadena padre-hijo, intervalos entre eventos, creación de archivos ejecutables y presencia de conexiones asociadas.

Patrones de procesos y modelado del comportamiento

Las relaciones padre-hijo inusuales entre procesos y las secuencias de descarga y ejecución inmediata son patrones relevantes porque pueden reflejar acciones tempranas de intrusión. MITRE

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

ATT&CK documenta el abuso de intérpretes de comandos Unix en la técnica T1059.004 y la transferencia de herramientas hacia el entorno comprometido en la técnica T1105, incluyendo el uso de utilidades como *curl* y *wget* en Linux (MITRE ATT&CK, 2025a, 2025b).

El modelado del comportamiento requiere transformar eventos en una representación útil para la comparación. Esa representación puede basarse en secuencias temporales, grafos de relación entre procesos o vectores de características. Los modelos recientes de análisis de registros muestran que la forma de representar los eventos incide en la capacidad de separar actividad normal y anómala, incluso cuando el algoritmo de clasificación es potente (Du et al., 2017; Khan et al., 2024).

MITRE ATT&CK y métricas de evaluación

MITRE ATT&CK ofrece una taxonomía de técnicas adversarias que permite vincular los escenarios del laboratorio con comportamientos observados en incidentes reales. En este estudio, la taxonomía se usa como referencia para justificar los escenarios y no como sustituto de la evaluación empírica. Las métricas principales son tasa de detección, tasa de falsos positivos, precisión, sensibilidad o *recall*, puntaje F1 y tiempo de detección. Estas medidas permiten comparar efectividad técnica y costo operativo.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Estado del arte

La revisión prioriza estudios publicados entre 2024 y 2025 sobre detección de anomalías en registros, auditoría de eventos y análisis del comportamiento de procesos, por ser los más cercanos al problema de esta investigación. La búsqueda se orienta a trabajos con validación experimental y relación directa con representación de secuencias, análisis sintáctico, grafos de eventos o evaluación comparativa de modelos.

La búsqueda bibliográfica se realizó en Scopus, Web of Science, IEEE Xplore, ACM Digital Library y bases editoriales especializadas. Se emplearon combinaciones de términos en español e inglés, entre ellos detección de anomalías en registros, *log anomaly detection*, registros de auditoría, *audit logs*, eventos de proceso y redes neuronales de grafos, o *graph neural networks*. El resumen de los trabajos analizados se presenta en la Tabla 1.

Una tendencia visible es el paso de flujos de procesamiento, llamados *pipelines*, donde el análisis sintáctico, o *parsing*, y la clasificación se tratan por separado, hacia modelos integrados que aprenden directamente desde el texto del registro. Hashemi y Mäntylä (2024) proponen OneLog, un modelo de extremo a extremo, o *end-to-end*, que procesa caracteres del registro y reduce la dependencia de componentes manuales.

Khan et al. (2024) cuestionan la idea de que una mayor exactitud del análisis sintáctico siempre conduce a una mejor detección de anomalías. Su estudio compara múltiples analizadores y técnicas de detección, y concluye que la distinguibilidad entre eventos es más decisiva para separar comportamiento normal y anómalo. Este hallazgo es importante para el presente proyecto

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

porque respalda la necesidad de diseñar variables que representen adecuadamente relaciones padre-hijo y secuencias de descarga y ejecución.

Niu et al. (2024) presentan WDLog, un marco robusto basado en el enfoque Wide & Deep para detección de anomalías en registros. Su aporte se centra en sostener el desempeño cuando los registros evolucionan o cuando aparecen anomalías de frecuencia. Zhen et al. (2025) proponen un método de detección de malware basado en registros de auditoría y redes neuronales de grafos, lo que resulta cercano al interés de esta investigación por reconstruir relaciones entre procesos. Ali et al. (2025) amplían la comparación entre técnicas tradicionales y profundas de aprendizaje automático para anomalías en registros, y muestran que la elección del modelo debe valorar precisión, tiempo y sensibilidad al ajuste de hiperparámetros.

En conjunto, la literatura reciente muestra tres tendencias principales: el interés por modelos integrados que reducen la dependencia del análisis manual de registros, el uso de representaciones más expresivas como secuencias y grafos, y la preocupación por robustez, interpretabilidad y costo de despliegue. La brecha específica se mantiene en la comparación experimental directa entre reglas tipo SIEM y un detector de anomalías con aprendizaje automático sobre un dispositivo final Linux instrumentado con *auditd*.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 1*Síntesis de estudios recientes sobre detección de anomalías en registros y auditoría de procesos*

Autor(es) / año	Estudio	Objetivo	Metodología	Variables	Hallazgos	Limitaciones	Relación con el proyecto
Hashemi y Mäntylä (2024)	OneLog: towards end-to-end software log anomaly detection	Proponer un detector integrado de anomalías en registros.	Red convolucional a nivel de carácter evaluada en seis conjuntos de datos.	Texto del registro, caracteres, números, puntuación y secuencias.	Logra buen desempeño y aprende reglas de análisis de forma implícita.	No se centra en Linux ni en auditoría de procesos.	Sustenta el valor de representar adecuadamente los registros antes de clasificar.
Khan et al. (2024)	Impact of log parsing on deep learning-based anomaly detection	Evaluar cómo influye el análisis sintáctico en la detección.	Comparación de 13 analizadores y 7 detectores en tres conjuntos de datos.	Exactitud del analizador, distinguibilidad y exactitud de detección.	La distinguibilidad pesa más que la exactitud del análisis sintáctico.	Usa conjuntos generales, no auditoría Linux local.	Justifica diseñar variables de procesos padre-hijo y descarga-ejecución.
Niu et al. (2024)	A robust Wide & Deep learning framework for log-based anomaly detection	Diseñar un marco robusto ante cambios en registros.	Modelo Wide & Deep con características temporales, invariantes y estadísticas.	Plantillas de registro, tiempo, invariantes y frecuencia.	Mejora robustez ante evolución del registro y anomalías de frecuencia.	Evalúa registros de sistemas generales, no el laboratorio propuesto.	Aporta criterios de robustez para reducir falsos positivos.
Zhen et al. (2025)	A novel malware detection method based on audit logs and graph neural network	Detectar malware a partir de registros de auditoría.	Extracción de eventos, construcción de grafos y red neuronal de grafos.	Secuencias de proceso, invocaciones y estructura del evento.	El uso de estructura y semántica mejora la detección y la explicación.	No compara reglas SIEM contra aprendizaje automático en Linux local.	Es el antecedente más cercano por su uso de auditoría y grafos.
Ali et al. (2025)	A comprehensive study of machine learning techniques for log-based anomaly detection	Comparar técnicas tradicionales y profundas para anomalías en registros.	Estudio empírico con criterios de exactitud, tiempo y sensibilidad.	Desempeño, tiempo de entrenamiento, predicción e hiperparámetros.	Modelos tradicionales pueden competir con modelos profundos según el contexto.	No aborda específicamente procesos Linux con auditd.	Refuerza la necesidad de comparar desempeño y costo operativo.

Nota. Elaboración propia a partir de la literatura revisada.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Metodología

Enfoque y alcance de la investigación

La investigación se desarrolla bajo un enfoque cuantitativo de tipo aplicado, con diseño experimental de laboratorio. El alcance es aplicado porque construye y evalúa un prototipo de detección sobre telemetría real de un dispositivo final Linux, y es experimental porque controla el entorno, los escenarios y las variables de comparación para evaluar dos estrategias sobre el mismo corpus de auditoría.

Como se trata de un experimento técnico, no se define población ni muestra de participantes humanos. La unidad de análisis corresponde a los eventos de auditoría generados por ejecuciones controladas. Las variables independientes son el enfoque de detección, reglas tipo SIEM o detector de anomalías, y el tipo de escenario ejecutado, actividad normal, relación padre-hijo inusual o descarga seguida de ejecución. Las variables dependientes son tasa de detección, tasa de falsos positivos, precisión, sensibilidad, puntaje F1 y tiempo de detección. Las variables de control incluyen el mismo dispositivo Linux, las mismas reglas de auditoría, ventanas temporales equivalentes y repeticiones bajo condiciones comparables.

Instrumentos

Los instrumentos principales son el dispositivo final Linux, el servicio *auditd* como fuente de telemetría, las reglas de auditoría para registrar creación de procesos y eventos relacionados, la plataforma SIEM seleccionada para correlación de reglas, y el módulo de procesamiento en Python. Las herramientas de análisis de datos son Python, pandas y NumPy para depuración y

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

construcción de variables, scikit-learn para entrenamiento y evaluación de modelos, Jupyter Notebook para reproducibilidad del análisis, y matplotlib o seaborn para visualización de métricas.

La validación de instrumentos se realiza mediante una prueba piloto. En esta etapa se comprueba que *auditd* capture los eventos requeridos, que las marcas de tiempo sean consistentes, que las relaciones entre procesos puedan reconstruirse y que las etiquetas de cada escenario coincidan con la ejecución real. Una muestra de registros se revisa manualmente antes de iniciar la evaluación comparativa.

Procedimiento experimental

El procedimiento se organiza en cinco etapas. En la primera se prepara el laboratorio: instalación del dispositivo Linux, activación de *auditd*, definición de reglas de auditoría, sincronización del reloj y verificación de escritura de registros. En la segunda se generan ventanas de actividad legítima que conforman la línea base del comportamiento normal. En la tercera se ejecutan escenarios controlados de relaciones padre-hijo inusuales y descarga seguida de ejecución, alineados con técnicas MITRE ATT&CK como T1059.004 y T1105 (MITRE ATT&CK, 2025a, 2025b).

En la cuarta etapa, los mismos registros alimentan los dos detectores. El enfoque basado en reglas aplica condiciones explícitas sobre proceso padre, proceso hijo, argumentos, rutas y eventos de red. El detector basado en aprendizaje automático recibe variables numéricas y categóricas derivadas de los registros, aprende una línea base de comportamiento y asigna puntajes de anomalía a nuevas ventanas de eventos. En la quinta etapa, los resultados se consolidan en matrices de confusión y tablas comparativas por escenario.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Técnicas para el análisis de la información

El análisis combina depuración de registros, normalización, ingeniería de características y evaluación comparativa. Los registros se agrupan por ventanas temporales y por relaciones de proceso; luego se construyen variables como frecuencia de ejecución, profundidad de la cadena padre-hijo, intervalos entre eventos, aparición de intérpretes de comandos, creación de archivos ejecutables, rutas involucradas y presencia de conexiones asociadas.

Para cada enfoque se calculan tasa de detección, tasa de falsos positivos, precisión, sensibilidad, puntaje F1 y tiempo de detección. Cuando el detector de anomalías produzca puntajes continuos, también se reporta el área bajo la curva precisión-sensibilidad, útil en escenarios desbalanceados. La comparación entre detectores se presenta mediante estadísticas descriptivas por escenario y, cuando las repeticiones generen pares comparables, mediante la prueba de rangos con signo de Wilcoxon, adecuada para contrastar mediciones pareadas sin asumir normalidad.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Resultados

Los resultados de esta investigación se presentan como una secuencia lógica entre los objetivos específicos, la metodología aplicada y los hallazgos verificables del prototipo. Debido a que el estudio se desarrolla como experimento técnico de laboratorio, los resultados no corresponden a respuestas de participantes humanos, sino a evidencias generadas por la instrumentación de un dispositivo final Linux, la ejecución de escenarios controlados y el procesamiento de registros de auditoría. En consecuencia, la sección diferencia entre resultados de construcción del prototipo, resultados de procesamiento de datos y resultados preliminares y ampliados de comparación.

La guía metodológica de la asignatura recomienda no manipular ni omitir hallazgos, incluso cuando estos no confirmen la hipótesis inicial. Por esa razón, los valores preliminares reportados en la primera sustentación se interpretan como evidencia de funcionamiento del flujo técnico, no como conclusión final sobre la superioridad de uno de los enfoques. Posteriormente, el 2 de junio de 2026 se ejecutó una corrida controlada con una línea base normal ampliada, repeticiones diferenciadas por escenario y ventanas de etiquetado más precisas.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 2*Relación entre objetivos específicos, metodología y resultados del avance*

Objetivo específico	Metodología aplicada	Resultado y análisis
Desarrollar el laboratorio local.	Preparación de VM Linux, activación de auditd y reglas auditctl.	Se estableció una arquitectura reproducible para capturar ejecuciones, cambios de permisos y actividad en un directorio de laboratorio.
Construir escenarios reproducibles.	Ejecución de actividad normal, relaciones padre-hijo inusuales y descarga local seguida de ejecución.	Se definieron escenarios inocuos que generan telemetría comparable sin usar malware ni atacar terceros.
Implementar reglas tipo SIEM.	Condiciones explícitas sobre proceso padre, proceso hijo, herramientas de descarga y ejecución desde el laboratorio.	El detector de reglas entrega alertas interpretables y permite rastrear la razón de cada detección.
Entrenar detector de anomalías.	Construcción de variables numéricas y categóricas e implementación inicial de Isolation Forest.	El modelo se entrenó con eventos normales etiquetados y produjo puntajes comparables para la validación controlada.
Evaluar comparativamente.	Cálculo de métricas como detección, falsos positivos, precisión, F1 y tiempo.	La corrida ampliada permitió comparar métricas por evento y cobertura por ventana; aún se requiere validación externa.

Nota. Elaboración propia a partir del prototipo técnico y de la primera sustentación.

Resultados técnicos del laboratorio

El primer resultado es la construcción de una ruta técnica completa para la obtención de evidencia. El laboratorio se organiza sobre una máquina virtual Linux, con *auditd* como fuente de telemetría y reglas de auditoría orientadas a eventos de ejecución de procesos, cambios de permisos y actividad dentro de un directorio de prueba. Esta decisión permite observar operaciones que son relevantes para el problema, sin convertir el laboratorio en un entorno ofensivo ni depender de herramientas maliciosas.

El segundo resultado es la definición de tres escenarios experimentales. El escenario de actividad normal incluye comandos administrativos benignos, lectura de archivos del sistema, creación de archivos temporales y ejecución de comandos simples. El escenario de relación padre-hijo inusual simula cadenas como Python, awk o find invocando una shell para crear archivos

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

inocuos. El escenario de descarga y ejecución utiliza un servidor local en 127.0.0.1, descarga un script de prueba, modifica permisos y lo ejecuta. Estos escenarios representan comportamientos observables en incidentes reales, pero se ejecutan con contenido seguro y controlado.

El tercer resultado es el diseño de un flujo de procesamiento. Los registros crudos de auditoría se agrupan por identificador de evento, se extraen campos relevantes y se convierten en un conjunto de datos tabular. El propósito no es conservar todos los registros sin procesar, sino obtener variables que puedan ser usadas por los dos enfoques de detección en igualdad de condiciones.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 3
Variables construidas para la comparación de detectores

Grupo	Variables	Uso analítico
Identificación temporal	timestamp_iso e interval_seconds	Ordenar eventos y medir cercanía entre acciones.
Relación de procesos	pid, ppid, parent_process y child_process	Reconstruir cadenas padre-hijo y detectar relaciones poco comunes.
Contexto de ejecución	user, command y path	Describir quién ejecuta, qué comando se invoca y desde dónde.
Señales de comportamiento	has_curl_wget, chmod_event y exec_from_tmp	Marcar descarga, cambio de permisos y ejecución desde el laboratorio.
Etiqueta experimental	scenariio y label	Comparar predicciones contra actividad normal o sospechosa.

Nota. Las variables resumen registros auditd para evitar trabajar con datos crudos no procesados.

Resultados preliminares de comparación

En la primera sustentación se reportó una corrida preliminar con 1.417 eventos procesados, dos detectores ejecutados y tres escenarios etiquetados. La lectura inicial mostró una tasa de detección o *recall* de 0,48 para el enfoque de reglas tipo SIEM y de 0,15 para Isolation Forest. Estos valores no deben interpretarse como resultado final, porque la línea base normal aún requiere mayor volumen y diversidad. Sin embargo, sí evidencian que ambos detectores pueden ejecutarse sobre el mismo conjunto de variables y producir salidas comparables.

La diferencia preliminar sugiere una explicación técnica razonable: las reglas explícitas pueden responder mejor cuando el escenario sospechoso coincide con condiciones definidas previamente, mientras que un modelo de anomalías necesita suficientes ejemplos normales para aprender qué desviaciones son realmente relevantes. Por tanto, el desempeño bajo de Isolation Forest en esta etapa no invalida el enfoque; indica que el modelo requiere una base de entrenamiento más representativa y un ajuste cuidadoso del umbral de anomalía.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Tabla 4*Lectura de métricas preliminares reportadas en la primera sustentación*

Indicador	Valor	Interpretación
Eventos procesados	1.417	Evidencia de ejecución inicial del pipeline; requiere repetición.
Detectores ejecutados	2	Reglas tipo SIEM e Isolation Forest sobre la misma telemetría.
Escenarios etiquetados	3	Actividad normal, relación padre-hijo inusual y descarga-ejecución.
Recall preliminar reglas	0,48	Mejor respuesta inicial ante patrones definidos explícitamente.
Recall preliminar Isolation Forest	0,15	Necesita línea base normal más amplia antes de concluir desempeño.

Nota. Los valores son preliminares y no sustituyen la evaluación final con repeticiones controladas.

Validación controlada con línea base ampliada

El 2 de junio de 2026 se realizó una nueva corrida controlada para superar la principal limitación identificada en la primera sustentación: la ausencia de una línea base normal suficiente para entrenar el modelo de anomalías. La ejecución completa procesó 35.787 eventos de audito. De ellos, 34.364 quedaron etiquetados como normales, 319 como sospechosos y 1.104 permanecieron sin etiqueta porque corresponden a actividad auxiliar fuera de las ventanas experimentales.

La nueva corrida incluyó cuatro ventanas prolongadas de actividad normal y ocho repeticiones de cada uno de los dos escenarios sospechosos: relación padre-hijo inusual y descarga local seguida de ejecución. Las ventanas se registraron con precisión de milisegundos y separación explícita entre escenarios. Además, se corrigió el parser para conservar identificadores numéricos como pid, ppid y usuario sin alterar la decodificación de argumentos textuales de EXECVE. Con esta base, Isolation Forest se entrenó únicamente con eventos etiquetados como normales.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

El detector de reglas alcanzó una precisión de 1,00, recall de 0,7273, tasa de falsos positivos de 0,00 y F1 de 0,8421 por evento. A nivel de ventanas detectó las 16 de 16 ventanas sospechosas y no generó alertas en las cuatro ventanas normales. Isolation Forest, con contaminación configurada en 0,05, alcanzó precisión de 0,0824, recall de 0,4671, tasa de falsos positivos de 0,0483 y F1 de 0,1400. El valor 0,05 obtuvo el mejor F1 entre los umbrales explorados: 0,02, 0,05, 0,10 y 0,15.

La lectura ampliada favorece al detector de reglas para los patrones conocidos y controlados del laboratorio. Isolation Forest detectó al menos una anomalía en las 16 ventanas sospechosas, pero también generó alertas en las cuatro ventanas normales. Por tanto, el modelo conserva valor como línea base exploratoria complementaria, no como sustituto directo de reglas explicables ni como evidencia de desempeño en producción.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 5
Métricas de la validación controlada con línea base normal ampliada

Indicador	Valor	Interpretación
Eventos auditd procesados	35.787	Conjunto total de eventos reconstruidos para la corrida ampliada.
Eventos normales etiquetados	34.364	Línea base utilizada para entrenar Isolation Forest.
Eventos sospechosos etiquetados	319	Eventos ubicados dentro de las ventanas controladas de prueba.
Precisión de reglas	1,00	No se observaron falsos positivos en las ventanas normales.
Recall de reglas	0,7273	Las reglas identificaron 232 de 319 eventos sospechosos.
F1 de reglas	0,8421	Mejor balance entre precisión y recall en la corrida controlada.
Recall de Isolation Forest	0,4671	El modelo identificó 149 de 319 eventos sospechosos.
FPR de Isolation Forest	0,0483	El modelo alertó sobre 1.660 de 34.364 eventos normales.
Cobertura de ventanas sospechosas	16/16	Ambos enfoques generaron al menos una alerta en cada ventana sospechosa.

Nota. Corrida ejecutada el 2 de junio de 2026 en laboratorio Linux controlado. Los resultados no equivalen a una validación en producción.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Análisis de resultados

El análisis de resultados no se limita a describir tablas o métricas, sino que relaciona los hallazgos del prototipo con el marco teórico, los objetivos y las limitaciones del diseño experimental. Desde esta perspectiva, el principal aporte del avance es demostrar que la comparación entre reglas tipo SIEM y un detector de anomalías puede realizarse bajo la misma fuente de telemetría, lo cual reduce sesgos asociados a comparar herramientas con datos distintos.

El comportamiento del detector de reglas se alinea con la literatura sobre sistemas basados en firmas: cuando el patrón de interés está definido con claridad, una regla puede entregar una alerta explicable y rastreable. Esto tiene valor operativo en un SOC porque el analista no solo recibe una clasificación, sino una razón: proceso padre, proceso hijo, ruta, comando o secuencia temporal. No obstante, esa misma fortaleza limita la capacidad de adaptación del enfoque ante variantes que no hayan sido anticipadas. En la corrida ampliada, las reglas alcanzaron un F1 de 0,8421 y cubrieron las 16 ventanas sospechosas sin alertar en las cuatro ventanas normales.

El resultado de Isolation Forest debe interpretarse desde la necesidad de línea base. Los modelos de anomalías no aprenden la intención maliciosa por sí mismos; aprenden la distribución de los datos que se les entregan. Si la muestra normal es pequeña, poco diversa o no representa adecuadamente la operación real del sistema, el modelo puede fallar al distinguir entre rareza estadística y riesgo de seguridad. Este punto coincide con los estudios revisados sobre detección de anomalías en registros, en los que la representación de eventos y la distinguibilidad de los datos resultan tan importantes como el algoritmo utilizado.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

La corrida ampliada permitió entrenar el modelo exclusivamente con 34.364 eventos normales etiquetados y comparar cuatro valores de contaminación. El mejor F1 explorado correspondió al umbral de 0,05, con valor de 0,1400. Aunque el modelo generó alertas dentro de todas las ventanas sospechosas, su precisión de 0,0824 y su tasa de falsos positivos de 0,0483 muestran que aún necesita mayor diversidad de actividad legítima, calibración y validación fuera del laboratorio.

Los resultados también muestran una limitación metodológica relevante: un laboratorio local permite controlar variables y repetir escenarios, pero no reproduce toda la complejidad de una organización real. En una empresa, los procesos legítimos pueden incluir tareas automatizadas, scripts de administración, herramientas de despliegue, agentes de monitoreo y actividades de usuarios con patrones muy diversos. Por ello, las conclusiones del estudio deben limitarse al entorno experimental y entenderse como base para pilotos controlados, no como garantía de desempeño universal.

Aun con esa restricción, el avance aporta una ruta reproducible: definir escenarios, capturar eventos, convertirlos en variables, aplicar dos enfoques de detección y evaluar con métricas homogéneas. Esa ruta es valiosa porque evita una comparación meramente conceptual entre reglas y aprendizaje automático. En lugar de afirmar que un método es mejor por principio, el proyecto propone medir en qué condiciones cada uno aporta mayor valor y cuál es el costo operativo de sus falsos positivos.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Propuesta de solución a la problemática

La solución propuesta es un flujo de monitoreo experimental para dispositivos finales Linux que combina telemetría auditd, reglas interpretables y detección de anomalías. La propuesta no reemplaza una plataforma SIEM o EDR empresarial; funciona como un prototipo replicable para evaluar qué tipo de detector conviene usar frente a cadenas de procesos sospechosas y secuencias de descarga seguida de ejecución.

En una implementación piloto, la organización iniciaría con endpoints Linux no productivos o de bajo riesgo, por ejemplo servidores de laboratorio, estaciones de desarrollo o máquinas de prueba. Allí se activarían reglas auditd enfocadas en ejecución de procesos, cambios de permisos y directorios críticos. Después se definirían escenarios normales propios de la organización y escenarios sospechosos seguros, sin malware real, para construir una base de comparación.

El flujo operativo se compone de siete pasos. Primero, instrumentar el endpoint y comprobar que auditd registre los eventos esperados. Segundo, recolectar actividad normal durante un periodo suficiente para representar el comportamiento cotidiano. Tercero, ejecutar escenarios controlados asociados a técnicas MITRE ATT&CK. Cuarto, transformar los registros en variables de comportamiento. Quinto, aplicar reglas explicables que funcionen como línea base. Sexto, entrenar un detector de anomalías con la actividad normal y evaluar su desempeño sobre escenarios nuevos. Séptimo, comparar tasas de detección, falsos positivos, precisión, F1 y tiempo de detección antes de escalar el piloto.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

La ventaja práctica de esta solución es que permite discutir seguridad con evidencia. Si las reglas generan demasiados falsos positivos, se ajustan condiciones. Si el modelo de anomalías produce alertas poco confiables, se amplía la línea base o se revisan variables. Si ambos enfoques detectan el mismo evento, el analista obtiene una señal reforzada. De esta forma, la propuesta contribuye a reducir ruido operativo y a mejorar la trazabilidad de las decisiones de monitoreo.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Análisis de costos

El análisis de costos se incluye para valorar la viabilidad de una implementación piloto y su posible sostenibilidad. En un proyecto de ingeniería no basta con que la solución funcione técnicamente; también debe estimarse qué recursos exige, qué costos directos e indirectos genera y bajo qué condiciones podría aportar beneficios frente al esfuerzo de implementación.

En este proyecto, el costo total puede organizarse en tres grupos: costos directos, costos indirectos y costos generales u overhead. Los costos directos corresponden a recursos necesarios para desarrollar y ejecutar el piloto, como horas de programación, configuración de la máquina virtual, almacenamiento de logs y eventual uso de servidores. Los costos indirectos incluyen actividades de soporte, validación, documentación, control de calidad y acompañamiento de seguridad. Los costos generales corresponden a recursos organizacionales compartidos, como administración, gestión del proyecto, infraestructura base y mantenimiento institucional.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 5
Categorías de costos para la implementación piloto

Categoría	Elementos incluidos	Lectura para viabilidad
Costos directos	Horas de desarrollo, configuración de auditd, VM, procesamiento en Python y almacenamiento de logs.	Aumentan si crecen los endpoints, escenarios o frecuencia de recolección.
Costos indirectos	Validación de reglas, revisión de evidencias, documentación, pruebas de calidad y soporte técnico.	Dependen del nivel de control requerido antes de llevar el piloto a producción.
Costos generales	Administración, coordinación, infraestructura compartida, gobierno de seguridad y mantenimiento.	Suelen distribuirse proporcionalmente entre proyectos de la organización.
Beneficios esperados	Reducción de ruido operativo, mejor priorización de alertas y evidencia reproducible para decisiones.	Deben compararse contra el costo de horas de analista y recursos de infraestructura.

Nota. El costo total del proyecto se estima como costos directos más costos indirectos más costos generales.

Para una organización, la evaluación económica debería complementar el prototipo con indicadores como retorno de la inversión, valor presente neto o tasa interna de retorno, siempre que existan datos financieros suficientes. En esta etapa académica no se calculan esos indicadores con valores monetarios cerrados, porque el costo depende del número de endpoints, la infraestructura disponible, las licencias requeridas y el tiempo real de los analistas. Sin embargo, el proyecto deja definida la estructura para realizar ese cálculo en una fase de transferencia.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Plan de divulgación y transferencia

La divulgación del proyecto busca acercar un conocimiento técnico a públicos con distintos niveles de especialización. El tema de detección en endpoints Linux puede resultar complejo si se presenta solo con siglas, comandos o métricas; por ello, la comunicación debe explicar de manera sencilla qué problema se resuelve, por qué importan los falsos positivos y cómo el prototipo ayuda a comparar reglas y aprendizaje automático con evidencia.

El primer público objetivo es académico: docentes, jurados y estudiantes de ingeniería interesados en ciberseguridad, análisis de logs y aprendizaje automático aplicado. Para este grupo se propone un artículo de investigación, una presentación de sustentación y un repositorio o paquete técnico reproducible. El segundo público es técnico-operativo: administradores Linux, analistas SOC y equipos de seguridad que podrían adaptar la metodología a sus propios entornos. Para ellos se recomienda una guía práctica con arquitectura, comandos, variables, métricas y criterios de ajuste.

El tercer público es organizacional: líderes de tecnología o seguridad que necesitan decidir si una estrategia de monitoreo justifica inversión y operación. Para este grupo, la transferencia debe centrarse en beneficios, costos, riesgos, límites y pasos de adopción. La explicación debe evitar prometer que el aprendizaje automático reemplaza las reglas; el mensaje central es que ambos enfoques pueden complementarse si se miden con datos comparables.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Tabla 6
Plan de divulgación y transferencia del proyecto

Público	Producto de divulgación	Propósito
Académico	Artículo, sustentación y anexo metodológico.	Mostrar problema, método, resultados, límites y aporte investigativo.
Técnico	Guía de laboratorio, scripts, diccionario de variables y métricas.	Facilitar reproducción del piloto y adaptación a otros endpoints Linux.
Organizacional	Resumen ejecutivo, matriz de costos y ruta de implementación.	Apoyar decisiones de adopción, escalamiento y control de falsos positivos.
Socialización	Charla breve, póster o demostración guiada.	Traducir conceptos como SIEM, auditd, anomalía y falso positivo a lenguaje accesible.

Nota. La divulgación prioriza lenguaje claro, precisión técnica y apropiación social del conocimiento.

La transferencia a empresas u organizaciones puede realizarse mediante un piloto de alcance reducido. El piloto debe iniciar con activos Linux no críticos, definir escenarios propios de la organización, medir la tasa de falsos positivos y documentar cada ajuste. Solo después de validar estabilidad, trazabilidad y beneficio operativo tendría sentido escalar a un conjunto mayor de endpoints. Esta ruta responde a la recomendación de presentar el proyecto no solo como desarrollo técnico, sino como propuesta replicable y aplicable en contextos reales.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Discusión

Los resultados del avance permiten discutir tres aspectos centrales: la utilidad de las reglas, la condición necesaria para que el aprendizaje automático sea útil y la importancia de una fuente de telemetría bien procesada. En primer lugar, las reglas tipo SIEM conservan valor porque son claras, auditables y fáciles de explicar. En un entorno donde el analista debe justificar por qué una alerta es relevante, esta interpretabilidad es una ventaja. Sin embargo, el enfoque depende de que el patrón haya sido definido con anterioridad.

En segundo lugar, Isolation Forest representa una opción prometedora para detectar desviaciones, pero su desempeño depende de la línea base. El modelo no debe evaluarse únicamente por el resultado de una primera corrida, sino por su comportamiento después de entrenarse con suficientes ejemplos normales, validarse con escenarios variados y calibrarse para controlar falsos positivos. Esta discusión es coherente con la literatura revisada, que advierte que los modelos de anomalías pueden ser sensibles a la representación de los registros y a la calidad de los datos.

En tercer lugar, auditd ofrece una fuente poderosa de evidencia, pero también genera volumen y complejidad. La telemetría cruda no se convierte automáticamente en conocimiento; requiere depuración, extracción de campos, reconstrucción de relaciones y definición de variables. Por ello, el proyecto muestra que la ingeniería de características es una parte sustancial de la solución y no un paso secundario.

La principal limitación del estudio es el alcance de laboratorio. Los escenarios controlados permiten seguridad, repetibilidad y análisis, pero no cubren la diversidad completa de una red

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

empresarial. Además, aunque la corrida ampliada ya permite calcular precisión, recall, falsos positivos y F1 dentro del entorno controlado, todavía no permite generalizar el desempeño a producción ni comparar tiempos de detección con suficiente estabilidad. Esta limitación delimita el tipo de afirmaciones que pueden hacerse y orienta los próximos pasos: diversificar la línea base, registrar tiempos y validar el comportamiento en pilotos.

**para la identificación de cadenas de procesos sospechosas en
*endpoints Linux***

Conclusiones

La investigación permitió estructurar una comparación experimental entre reglas tipo SIEM y un detector de anomalías basado en aprendizaje automático para identificar cadenas de procesos sospechosas en endpoints Linux. El objetivo general se cumple en la medida en que se definió el laboratorio, se construyó la ruta de captura y procesamiento, y se evaluaron dos enfoques de detección con las mismas métricas. La corrida controlada ampliada procesó 35.787 eventos y permitió entrenar Isolation Forest con 34.364 eventos normales etiquetados. Estos resultados fortalecen la comparación, aunque todavía requieren validación externa.

Respecto al primer objetivo específico, se concluye que *auditd* es una fuente adecuada para observar eventos de ejecución, relaciones entre procesos y cambios de permisos en un dispositivo final Linux. Su uso permite construir evidencia reproducible, aunque exige configuración cuidadosa para evitar registros excesivos o poco pertinentes.

Frente al segundo objetivo específico, los escenarios de actividad normal, relación padre-hijo inusual y descarga seguida de ejecución ofrecen una base segura para generar telemetría comparable. Al no utilizar malware real ni conexiones externas peligrosas, el laboratorio conserva control ético y técnico, pero mantiene relación con comportamientos observados en incidentes documentados por MITRE ATT&CK.

En relación con el tercer objetivo específico, las reglas tipo SIEM mostraron una ventaja inicial: permiten explicar con claridad la razón de una alerta. Esta característica es relevante para reducir incertidumbre operativa en un SOC, especialmente cuando el analista necesita distinguir entre actividad legítima y secuencias sospechosas. En la corrida ampliada alcanzaron precisión de

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

1,00, recall de 0,7273 y F1 de 0,8421 por evento, además de cobertura de 16 de 16 ventanas sospechosas.

Sobre el cuarto objetivo específico, el detector basado en Isolation Forest quedó implementado como línea base de aprendizaje automático y se entrenó exclusivamente con eventos normales etiquetados. Su F1 de 0,1400 y su tasa de falsos positivos de 0,0483 evidencian que necesita mayor diversidad de datos normales y calibración. La conclusión no es que el aprendizaje automático sea inadecuado, sino que su utilidad depende de una representación sólida del comportamiento y de una validación rigurosa.

Finalmente, respecto al quinto objetivo específico, las métricas definidas permiten evaluar la comparación de manera objetiva. La tasa de detección por sí sola no es suficiente; también deben analizarse falsos positivos, precisión, F1 y tiempo de detección. Esta mirada evita declarar un enfoque superior de forma prematura y orienta la discusión hacia el equilibrio entre seguridad, ruido operativo e interpretabilidad.

**para la identificación de cadenas de procesos sospechosas en
endpoints Linux**

Recomendaciones

Se recomienda continuar ampliando la línea base normal antes de trasladar la evaluación del detector de anomalías a un entorno productivo. Esta línea debe incluir más sesiones, variaciones de comandos legítimos, usuarios diferentes y tareas administrativas habituales, de manera que el modelo no confunda rareza estadística con comportamiento sospechoso.

También se recomienda ejecutar múltiples repeticiones por escenario y registrar el tiempo entre el inicio del evento y la alerta generada por cada enfoque. Esta medición permitirá comparar no solo cuántos eventos se detectan, sino qué tan rápido se detectan y con cuánto ruido operativo.

Para una fase de transferencia, se sugiere aplicar el prototipo en un piloto organizacional limitado, con endpoints Linux no críticos y reglas auditd ajustadas al contexto real. La adopción debe acompañarse de un análisis de costos, beneficios y riesgos antes de escalar a ambientes productivos.

Como trabajo futuro, sería conveniente comparar Isolation Forest con otros modelos de anomalías, incluir representaciones de grafos de procesos y evaluar si la combinación de reglas más puntajes de anomalía reduce falsos positivos sin sacrificar capacidad de detección.

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Referencias

- Ali, S., Boufaied, C., Bianculli, D., Branco, P., & Briand, L. (2025). A comprehensive study of machine learning techniques for log-based anomaly detection. *Empirical Software Engineering*, 30, Article 129. <https://doi.org/10.1007/s10664-025-10669-3>
- Bace, R., & Mell, P. (2001). *Intrusion detection systems (NIST SP 800-31)*. National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-31>
- Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15. <https://doi.org/10.1145/1541880.1541882>
- Cybersecurity and Infrastructure Security Agency, National Security Agency, Federal Bureau of Investigation, Department of Energy, Environmental Protection Agency, Transportation Security Administration, Australian Signals Directorate's Australian Cyber Security Centre, Canadian Centre for Cyber Security, National Cyber Security Centre United Kingdom, & New Zealand National Cyber Security Centre. (2024). Identifying and mitigating living off the land techniques. <https://www.cisa.gov/resources-tools/resources/identifying-and-mitigating-living-land-techniques>
- Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1285-1298. <https://doi.org/10.1145/3133956.3134015>

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

- Garcia-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18-28. <https://doi.org/10.1016/j.cose.2008.08.003>
- Hashemi, S., & Mäntylä, M. (2024). OneLog: Towards end-to-end software log anomaly detection. *Automated Software Engineering*, 31, Article 37. <https://doi.org/10.1007/s10515-024-00428-x>
- IBM Security. (2024). X-Force Threat Intelligence Index 2024. <https://www.ibm.com/think/x-force/2024-x-force-threat-intelligence-index>
- Kerrisk, M. (2026). auditd(8): Linux manual page. man7.org. <https://man7.org/linux/man-pages/man8/auditd.8.html>
- Khan, Z. A., Shin, D., Bianculli, D., & Briand, L. C. (2024). Impact of log parsing on deep learning-based anomaly detection. *Empirical Software Engineering*, 29, Article 139. <https://doi.org/10.1007/s10664-024-10533-w>
- Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). Isolation Forest. 2008 Eighth IEEE International Conference on Data Mining, 413-422. <https://doi.org/10.1109/ICDM.2008.17>
- MITRE ATT&CK. (2025a). Command and Scripting Interpreter: Unix Shell, sub-technique T1059.004. <https://attack.mitre.org/techniques/T1059/004/>
- MITRE ATT&CK. (2025b). Ingress Tool Transfer, technique T1105. <https://attack.mitre.org/techniques/T1105/>

para la identificación de cadenas de procesos sospechosas en
endpoints Linux

Niu, X., Li, Z., Wang, J., & Zhang, L. (2024). A robust Wide & Deep learning framework for log-based anomaly detection. *Applied Soft Computing*, 153, Article 111314.

<https://doi.org/10.1016/j.asoc.2024.111314>

Scarfone, K., & Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS) (NIST SP 800-94). National Institute of Standards and Technology.

<https://doi.org/10.6028/NIST.SP.800-94>

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE Symposium on Security and Privacy*, 305-316.

<https://doi.org/10.1109/SP.2010.25>

Zhen, Y., Tian, D., Fu, X., & Hu, C. (2025). A novel malware detection method based on audit logs and graph neural network. *Engineering Applications of Artificial Intelligence*, 152, Article 110524. <https://doi.org/10.1016/j.engappai.2025.110524>