Plataforma Open Source para la Gestión Centralizada de Eventos de Seguridad y Monitoreo en Entornos Institucionales

Juan Diego Lugo Sánchez, Lindsay Vanessa Pinto Morato, Diego Alejandro González Vargas
Departamento de Ingeniería de Sistemas y Computación
Universidad de los Andes
Bogotá, Colombia
{jd.lugo, l.pintom, da.gonzalezv1}@uniandes.edu.co

***Resumen*—Este artículo muestra el proceso de desarrollo de una plataforma para la gestión centralizada de eventos de monitoreo y seguridad en un entorno institucional. La solución integra varias fuentes con herramientas open source. La plataforma unifica eventos operativos y de seguridad en un flujo de análisis, permitiendo correlación basada en reglas, visualización unificada y notificación automática.**

Así mismo, se incorpora un módulo experimental de machine learning para la predicción de severidad el cuál fue entrenado a partir de datos históricos de logs de Zabbix. Los resultados obtenidos en un entorno de pruebas real demuestran la viabilidad técnica de integrar múltiples fuentes de monitoreo bajo una arquitectura modular open source, así como el potencial del modelo de predicción para automatizar la priorización de incidentes en fases posteriores de despliegue.

***Index Terms*—monitoreo, ciberseguridad, SIEM, IDS, open source, machine learning**

I. Introducción

La organización encargada de la administración y monitoreo de los recursos tecnológicos del departamento de sistemas requiere contar con un control y seguimiento integral de los activos bajo su gestión. De esta manera puede acomodarse a las necesidades de los diferentes grupos y personas del departamento, así como atender incidentes de seguridad que se generen tanto por el mal uso de los recursos de manera intencionada como por fallas no esperadas en los sistemas.

Para poder realizar esta labor, se debe apoyar de herramientas como software y hardware que permitan esa gestión. Sin embargo, los sistemas existentes presentaban deficiencias en términos de cobertura, descentralización y análisis. Por ello, este proyecto busca establecer un sistema de gestión centralizada con mayor alcance y con funcionalidades de análisis de eventos que respondan de manera más efectiva a las necesidades misionales de la organización.

II. Diseño de la Solución

II-A. Identificación de Requerimientos

La organización identificó las siguientes necesidades críticas para mejorar su postura de ciberseguridad y monitoreo:

- **Centralización de eventos:** Integrar múltiples fuentes de información (infraestructura, seguridad endpoint, red) en una única plataforma de análisis.
- **Detección temprana de amenazas:** Implementar capacidades de detección de intrusiones y análisis de vulnerabilidades en tiempo real.
- **Correlación de eventos:** Relacionar eventos de diferentes fuentes para identificar incidentes complejos y patrones de ataque.
- **Automatización de respuesta:** Generar alertas automáticas y facilitar la priorización de incidentes mediante análisis inteligente.
- **Visibilidad unificada:** Proveer dashboards centralizados que permitan el monitoreo integral del estado de la infraestructura y la seguridad.

Estos requerimientos guiaron el diseño de una arquitectura modular que pudiera escalar progresivamente y adaptarse a las necesidades cambiantes de la organización.

II-B. Arquitectura del Sistema

La plataforma fue diseñada siguiendo una arquitectura modular de tres capas: recolección, procesamiento y presentación. Esta arquitectura permite la integración flexible de múltiples fuentes de datos y facilita la correlación de eventos de seguridad e infraestructura.

Los componentes principales de la arquitectura son:

- **Capa de Recolección:** Incluye agentes instalados en servidores y estaciones de trabajo para monitoreo de infraestructura, agentes de seguridad endpoint para recolección de logs del sistema operativo y verificación de integridad, y sensores de red para inspección de tráfico mediante Sistema de Detección de Intrusiones (IDS, por sus siglas en inglés).
- **Capa de Procesamiento:** Centraliza el servidor Extended Detection and Response (XDR) que normaliza eventos mediante decodificadores, aplica reglas de correlación para identificar patrones de ataque, y almacena eventos en bases de datos indexadas para consulta histórica.
- **Capa de Presentación:** Proporciona dashboards de visualización unificada, sistema de alertas y notificaciones automáticas, y interfaces de consulta y análisis de eventos históricos.

- **Módulo Experimental:** Incorpora un pipeline de machine learning implementado en Jupyter Notebook para preprocesamiento de datos, entrenamiento del modelo predictivo de severidad, y evaluación de desempeño del modelo.

La arquitectura sigue un flujo de datos donde los eventos son capturados por los agentes en los endpoints, transmitidos de forma segura al servidor XDR central, normalizados y correlacionados según reglas predefinidas, y finalmente presentados en dashboards unificados. El módulo de machine learning opera de forma independiente sobre datos históricos exportados del sistema de monitoreo.

Esta separación en capas permite escalar cada componente de manera independiente, facilita el mantenimiento y actualización de módulos individuales, y proporciona flexibilidad para integrar nuevas fuentes de datos o algoritmos de análisis.

III. Metodología de Implementación

El desarrollo de este proyecto se realizó mediante un enfoque incremental compuesto por cinco etapas principales:

1. **Preparación del entorno:** Esta etapa consistió en la instalación de servidores dedicados en el centro de datos de la organización, la provisión de máquinas virtuales de prueba para validación de componentes, y la configuración inicial de la red y permisos de acceso necesarios para la operación del sistema.
2. **Producto mínimo viable:** Se implementó la captura básica de eventos desde diferentes agentes de monitoreo del sistema operativo y de red, estableciendo la comunicación segura entre agentes y el servidor XDR para garantizar la ingesta inicial de datos operativos.
3. **Integración y correlación:** Se activaron los módulos de correlación en el servidor XDR, se configuró la normalización automática de eventos mediante decodificadores específicos para cada fuente de datos, y se crearon reglas básicas de correlación para detectar patrones de ataque conocidos.
4. **Modelado y aprendizaje automático:** Se exportaron datos históricos de un año desde el sistema de monitoreo de infraestructura, se realizó limpieza y preprocesamiento de los datos, se ejecutó ingeniería de características para extracción de atributos relevantes, y se entrenó el modelo predictivo de severidad.
5. **Pruebas y validación:** Se realizó análisis de cobertura de agentes desplegados, evaluación del volumen de eventos procesados diariamente, y validación del desempeño del modelo predictivo en diferentes escenarios.

Este enfoque permitió validar cada componente antes de extender su funcionalidad y garantizó una integración progresiva de los módulos sin afectar las operaciones existentes de la organización.

IV. Implementación

IV-A. Monitoreo de infraestructura

Se desplegaron agentes de monitoreo en servidores críticos de la infraestructura del departamento utilizando el sistema Zabbix. Los agentes se configuraron en cada servidor Linux y Windows para registrar disponibilidad de servicios, uso de recursos del sistema (CPU, memoria, disco), y estado de servicios críticos.

El propósito principal fue generar un histórico extenso de eventos operacionales que posteriormente se utilizó como dataset de entrenamiento en el módulo de aprendizaje automático. La configuración incluyó la definición de triggers personalizados para cada tipo de servidor y la integración con el servidor XDR para envío de eventos de alta criticidad.

IV-B. Seguridad endpoint

Se configuró el sistema en el servidor XDR para recolectar logs del sistema operativo de cada endpoint, monitorear la integridad de archivos críticos mediante checksums, y detectar vulnerabilidades conocidas en el software instalado. Debido a las restricciones de red internas de la organización, específicamente las políticas de firewall que bloqueaban el puerto por defecto, se modificaron los puertos de autenticación de agentes al 8088 y de flujo de datos al 8090.

Esta modificación se realizó tanto en el archivo de configuración del servidor (`ossec.conf`) como en los agentes desplegados, con el propósito de permitir la comunicación bidireccional a través de las reglas de firewall existentes. Así mismo, se generaron guías de instalación documentadas para facilitar el despliegue de nuevos agentes por parte del equipo de operaciones.

IV-C. Sistema de Detección de Intrusiones de red

Se configuró un software open source como IDS en un servidor con acceso a un puerto SPAN del switch core de la red del departamento, con el propósito de inspeccionar el tráfico de red en busca de patrones de ataque conocidos. Se utilizaron las reglas de la comunidad Emerging Threats (ET/Open) para la detección de amenazas, las cuales se actualizan automáticamente de forma diaria.

Las alertas generadas por Suricata se configuraron para enviarse automáticamente al servidor XDR mediante un agente de Wazuh que lee el archivo de alertas de Suricata (`fast.log`). Esta configuración se eligió en lugar del archivo JSON por defecto (`eve.json`) debido a problemas detectados en la transmisión relacionados con el tamaño excesivo de los paquetes de datos.

El archivo `fast.log` contiene únicamente la información esencial de cada alerta (timestamp, IP origen/destino, mensaje), lo cual permite una transmisión estable sin saturar el canal de comunicación entre el IDS y el servidor XDR.

IV-D. Ingesta y visualización

Todos los eventos provenientes de las diferentes fuentes (Zabbix, Wazuh, Suricata) fueron centralizados en el servidor XDR. En este servidor se configuraron decodificadores específicos en Wazuh para normalizar el formato de cada tipo de evento, permitiendo la correlación efectiva entre diferentes fuentes.

Se implementaron dashboards personalizados en la interfaz web de Wazuh para visualizar métricas clave como: volumen

de eventos por fuente, distribución de alertas por severidad, y eventos de seguridad en tiempo real. El propósito de estos dashboards es proporcionar al equipo de seguridad una vista unificada del estado de la infraestructura y facilitar la detección temprana de incidentes.

IV-E. Módulo de Machine Learning

Se diseñó un pipeline completo de machine learning en Jupyter Notebook para la predicción automática de severidad de eventos. Los datos históricos de Zabbix de un año completo se exportaron a formato CSV, se limpiaron eliminando registros duplicados y valores nulos, y se balancearon las clases mediante técnicas de sobremuestreo para evitar sesgo hacia la clase mayoritaria.

El preprocesamiento incluyó la extracción de características textuales del mensaje de alerta mediante TF-IDF y la codificación de variables categóricas. Se entrenó un modelo Random Forest utilizando la biblioteca scikit-learn, el cual demostró un desempeño superior al 95 % de exactitud en el conjunto de validación. El modelo entrenado se guardó en formato pickle para su potencial integración futura con el sistema en producción.

V. Pipeline de Machine Learning

El pipeline de aprendizaje automático que se desarrolló en este proyecto aún no se encuentra integrado con el Sistema de Gestión de Información y Eventos de Seguridad (SIEM) operativamente. Esto se debe a que, antes de habilitar un flujo de inferencia en tiempo real, se hacía necesario validar la viabilidad del modelo, su estabilidad bajo diferentes escenarios y su capacidad de generalización.

La integración está considerada como trabajo futuro, el cual puede realizarse mediante un microservicio o Interfaz de Programación de Aplicaciones (API) que permita enviar predicciones directamente al sistema de correlación. A continuación se describen las etapas principales de este flujo:

V-A. Ingesta de datos

La fase de ingesta tuvo como propósito estandarizar los datos provenientes de los registros exportados desde la plataforma de monitoreo en el servidor Zabbix ubicado en el centro de datos de la organización. El dataset inicial incluyó 26,454 eventos. Los datos utilizados para entrenar y validar el modelo corresponden a registros reales obtenidos de la herramienta de monitoreo en la institución, recopilados durante un periodo de un año entre septiembre de 2024 y septiembre de 2025.

Estos eventos incluyen las etiquetas (tags) generadas automáticamente por los agentes de monitoreo, que asigna metadatos relacionados con el tipo de incidencia, el componente afectado y otros atributos del evento. Estas etiquetas fueron posteriormente procesadas y expandidas en columnas individuales en el notebook de Jupyter, facilitando su inclusión en el pipeline de características.

Durante la etapa de ingesta se llevaron a cabo las siguientes acciones en el código Python del notebook:

- **Estandarización de columnas:** Homogenización de nombres provenientes de Zabbix en el archivo CSV exportado, tales como `Hora` a `timestamp`, `Equipo` a `host`, `Problema` a `message`, con el propósito de normalizar el dataset para su procesamiento posterior.
- **Conversión de marcas de tiempo:** Transformación de campos temporales al tipo `datetime` de pandas para habilitar análisis temporal y extracción de características temporales.
- **Creación de metadatos:** Incorporación de un identificador único por evento (`event_id`) y un campo de procedencia (`source`) para trazabilidad durante el análisis.
- **Generación de artefactos:** Almacenamiento de snapshots del dataset crudo y normalizado en los directorios `data/raw/` y `data/processed/` para reproducibilidad del experimento.

La distribución de severidades resultante mostró mayor concentración en las categorías intermedias con 10,065 eventos de severidad 2, 9,065 de severidad 3, 6,483 de severidad 4 y 840 de severidad 1.

V-B. Preprocesamiento de datos

El preprocesamiento tuvo como objetivo transformar el dataset normalizado en una matriz de características limpia que pudiera ser utilizada por los algoritmos de machine learning. Este proceso incluyó las siguientes operaciones realizadas en el notebook de Jupyter con la biblioteca pandas:

- **Eliminación de duplicados** según `event_id` para evitar sesgo en el entrenamiento causado por eventos repetidos.
- **Gestión de valores nulos** en columnas clave como `timestamp`, `message` y `severity`, eliminando registros incompletos que no podían ser imputados de manera confiable.
- **Conversión y unificación de duraciones:** Normalización de duraciones textuales (expresadas en formatos como "2h 30m.º "150 minutos") a minutos para homogenizar la representación numérica.
- **Cálculo del tiempo de resolución:** Diferencia temporal entre `timestamp` y `timestamp_recovery` para obtener una métrica de tiempo de resolución que se utilizó como característica del modelo.

El tiempo medio de resolución (MTTR) de los datos analizados fue de 445.26 minutos, equivalente a 7.4 horas.

V-B1. Estandarización de la severidad: Con el fin de utilizar la severidad como variable objetivo para el modelo, se definió una escala numérica de la siguiente forma:

Severidad textual	Código
Informativa	1
Advertencia	2
Promedio	3
Alta	4

V-B2. Expansión de etiquetas: El campo `tags` contenía datos en formato llave:valor por lo cuál se realizó expansión de las etiquetas en columnas individuales mediante funciones de procesamiento de cadenas en Python, haciendo más manejable el conjunto de características con variables derivadas de atributos del evento. De igual forma, algunos eventos que contienen más de 2 valores por característica fueron expandidos a nuevas columnas para maximizar la información disponible.

El dataset final incluyó 20,058 registros tras los procesos de limpieza y normalización.

V-C. Ingeniería de características

Esta fase tuvo como propósito capturar patrones de relevancia del entorno operativo mediante la extracción y transformación de variables. Las variables consideradas dentro del modelo incluyen:

- **Variables numéricas:** `duration_min`, `resolution_time_min` que capturan aspectos temporales de los eventos.
- **Variables categóricas:** `host`, `component`, `scope`, `service`, `target` que identifican el origen y contexto del evento.
- **Variable textual:** `message`, procesada mediante vectorización Term Frequency–Inverse Document Frequency (TF-IDF) implementada con scikit-learn para convertir el texto en representaciones numéricas.
- **Atributos derivados:** Hora del día, día de la semana y recuento de incidentes recientes extraídos de los timestamps para capturar patrones temporales.

V-D. Análisis exploratorio

Se realizó un análisis descriptivo en el notebook de Jupyter para identificar patrones asociados a la severidad, observando que los eventos con severidad *Alta* y *Promedio* tenían tiempos de resolución mayores que las categorías inferiores. La Figura 1 muestra esta relación.

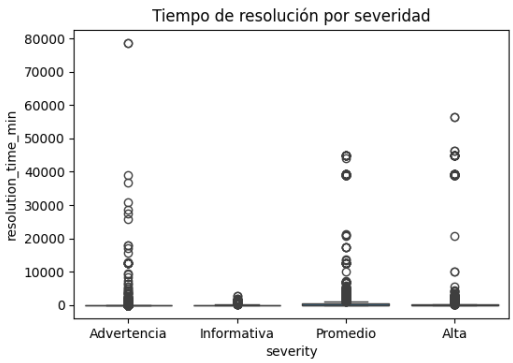

Figura 1. Distribución del tiempo de resolución por categoría de severidad.

Además, se generaron histogramas, conteos estratificados por severidad y correlaciones entre variables temporales mediante las bibliotecas matplotlib y seaborn, lo cual permitió seleccionar las características más relevantes para el modelo. De igual manera, el análisis exploratorio permitió identificar tendencias asociadas a la severidad:

- La severidad presenta desproporción entre clases siendo las clases dominantes las correspondientes a severidad 2 y 3.
- Las clases *Promedio* y *Alta* tuvieron los mayores tiempos de resolución.
- El contenido de texto del mensaje del evento mostró patrones de diferenciación entre severidades.

Estas observaciones hicieron que se optara por el uso de modelos que pudieran manejar relaciones no lineales y complejas.

V-E. Balanceo de clases

Debido al desbalance presente en la distribución de severidades, se aplicó la técnica SMOTE (Synthetic Minority Oversampling Technique) de la biblioteca imbalanced-learn en el ambiente de Python para generar ejemplos sintéticos de las clases con menos datos. Este enfoque mejoró la estabilidad del modelo y evitó que la predicción se concentrara en las clases que tenían más datos, con el propósito de mejorar las métricas de recall en clases minoritarias.

V-F. Preparación del dataset para modelado

El pipeline de preprocesamiento se construyó utilizando un *ColumnTransformer* de scikit-learn con las siguientes transformaciones específicas para cada tipo de variable:

- **Categóricas:** Imputación constante para valores faltantes y codificación One-Hot para convertir categorías en variables binarias.
- **Texto:** Vectorización TF-IDF con un máximo de 100 características para limitar la dimensionalidad y reducir el riesgo de overfitting.
- **Numéricas:** Imputación por media para valores faltantes y normalización mediante *StandardScaler* para estandarizar la escala de las variables.

El conjunto se dividió en 80 % entrenamiento y 20 % prueba, con estratificación por clase de severidad para mantener proporciones balanceadas en ambos conjuntos.

V-G. Evaluación comparativa de modelos

Se evaluaron múltiples modelos supervisados mediante validación cruzada de cinco particiones (5-fold cross-validation) en el conjunto de entrenamiento:

- Regresión Logística,
- Árbol de Decisión,
- Lineal Support Vector Machine (SVM),
- Gradient Boosting,
- Random Forest.

El modelo Random Forest fue el de mayor desempeño promedio comparado con los demás modelos analizados. Además, este modelo ofrece mayor robustez ante overfitting e interpretabilidad mediante análisis de importancia de características por su arquitectura de ensamble de múltiples árboles de decisión.

V-H. *Modelo final y métricas de desempeño*

El modelo final consistió en un Random Forest con 100 árboles y balanceo de clases mediante el parámetro `class_weight='balanced'` en scikit-learn. Los resultados obtenidos en el conjunto de prueba mostraron:

- **Exactitud**: 100 %.
- **Precision, recall y f1-score**: $\approx 1{,}00$ en todas las clases.

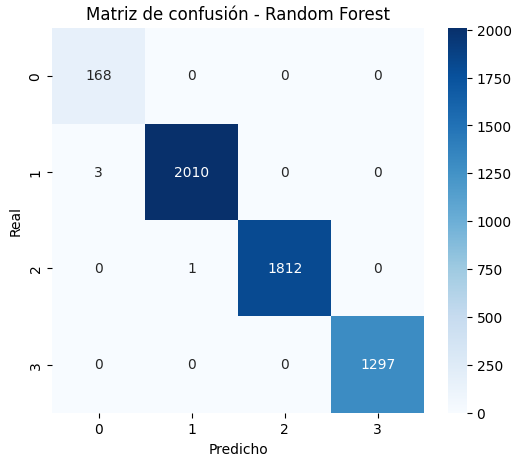

Figura 2. Matriz de confusión para el modelo Random Forest.

La matriz de confusión mostró una separación perfecta entre clases en el conjunto de prueba con solo 2 variaciones mínimas para la clase 1 y 2 (Figura 2).

V-I. *Importancia de variables*

El análisis de importancia de características del modelo Random Forest mostró que las características derivadas del campo `message` mediante TF-IDF tuvieron más influencia en las predicciones, especialmente aquellos términos asociados a problemas de conectividad o disponibilidad. Entre las características numéricas, `resolution_time_min` mostró una contribución significativa al modelo (Figura 3).

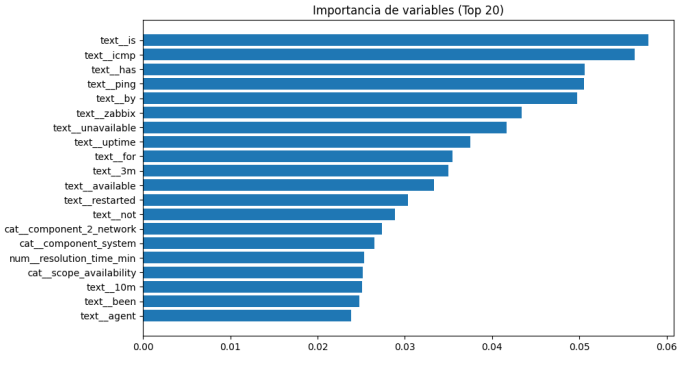

Figura 3. Importancia de variables para el Modelo.

V-J. *Limitaciones del modelo*

Las principales limitaciones identificadas son:

- El modelo opera en modo *offline* y no ha sido integrado en un flujo en tiempo real con el servidor XDR.
- El dataset proviene de una única fuente (Zabbix), lo que puede limitar la generalización a eventos de otras plataformas.
- La disponibilidad de etiquetas depende del sistema original de monitoreo y su configuración de triggers.
- No se incorpora retroalimentación continua ni reentrenamiento automático basado en nuevos eventos.

V-K. *Trabajo futuro*

Existen varias oportunidades para continuar y ampliar este trabajo en desarrollos posteriores:

- Desplegar el modelo como un microservicio RESTful mediante Flask o FastAPI, de modo que pueda consultarse mediante una API y generar predicciones en tiempo real para eventos recién ingresados al XDR.
- Exportación automática de predicciones a sistemas de visualización como dashboards de Wazuh o Grafana.
- Reentrenamiento programado basado en ventanas temporales deslizantes para adaptar el modelo a cambios en la infraestructura.
- Incorporación de datos provenientes de múltiples fuentes (SIEM, IDS, logs de red, firewall) para mejorar la capacidad de generalización.

La integración podría permitir la automatización de la priorización de alertas dentro de una arquitectura con capacidades de detección extendida.

VI. Resultados y Validación

VI-A. *Resultados operativos*

La plataforma habilitó las siguientes capacidades operativas en el entorno de producción:

- Ingesta simultánea de eventos de infraestructura desde Zabbix, seguridad endpoint desde agentes de Wazuh, y tráfico de red desde Suricata IDS.
- Normalización automática mediante decodificadores personalizados de Wazuh configurados en el servidor XDR para cada tipo de fuente.
- Dashboards con análisis temporal unificado que permiten visualizar eventos de múltiples fuentes en una única interfaz web.

VI-B. *Resultados del modelo predictivo*

Los resultados obtenidos durante la validación del modelo consideran tres etapas complementarias entre sí: (i) evaluación con datos reales del sistema de monitoreo, (ii) simulación de eventos provenientes de múltiples fuentes heterogéneas y (iii) pruebas de robustez orientadas a evaluar el comportamiento del modelo bajo ruido, ambigüedad, volúmenes elevados y variaciones de los eventos. Todos los experimentos fueron ejecutados sobre el pipeline final entrenado, que integra preprocesamiento y clasificación Random Forest.

VI-B1. Evaluación con datos reales: Para esta prueba se seleccionaron 100 eventos procesados por el pipeline de ingesta desde el servidor Zabbix con el fin de medir la capacidad del modelo para reproducir las severidades registradas originalmente en el sistema de monitoreo.

Los resultados de esta etapa se muestran en la Tabla I.

Métrica	Valor
Eventos evaluados	100
Coincidencias	97
Diferencias	3
Precisión global	97.0 %

Cuadro I
RESULTADOS DE EVALUACIÓN CON DATOS REALES.

Asimismo, la distribución de severidades predichas se mantuvo balanceada y cercana a las proporciones observadas en los datos de entrenamiento. La Tabla II resume estos valores.

Severidad	Porcentaje
Advertencia (2)	36.0 %
Promedio (3)	41.0 %
Alta (4)	23.0 %

Cuadro II
DISTRIBUCIÓN DE SEVERIDADES PREDICHAS CON DATOS REALES.

Estos resultados reflejan que bajo condiciones reales, el modelo reproduce de manera correcta las etiquetas originales, con una precisión superior al 95 %.

VI-B2. Simulación multifuente: El segundo experimento consistió en generar 50 eventos sintéticos provenientes de orígenes diversos como sistemas operativos, aplicaciones web, IDS, firewall, y monitoreo de infraestructura, entre otros. El propósito fue evaluar la capacidad del modelo para generalizar a contenidos textuales y numéricos no presentes en los datos de entrenamiento que provenían exclusivamente de Zabbix.

La Tabla III resume el promedio de severidad asignada por el modelo para cada tipo de fuente simulada.

Fuente	Severidad Promedio
Aplicación	2.00
Endpoint	2.00
Firewall	2.00
IDS	2.00
SIEM	2.00
Sistema	1.75
Zabbix	2.66

Cuadro III
PROMEDIO DE SEVERIDAD PREDICHA POR FUENTE EN SIMULACIONES MULTIFUENTE.

En cuanto a la coincidencia entre severidades reales y predichas, se obtuvo una precisión del 30.4 %. Este valor, aunque es menor que el obtenido con datos reales, se puede explicar porque las simulaciones se realizaron con expresiones y estructuras lingüísticas que no estaban presentes en el conjunto de entrenamiento, lo que aporta evidencia sobre las limitaciones actuales del modelo ante variaciones significativas en el vocabulario.

La distribución general de severidades predichas se presenta en la Tabla IV.

Severidad	Porcentaje
Informativa (1)	4.0 %
Advertencia (2)	90.0 %
Alta (4)	6.0 %

Cuadro IV
DISTRIBUCIÓN DE SEVERIDADES PREDICHAS EN SIMULACIÓN MULTIFUENTE.

VI-B3. Pruebas de robustez: Finalmente, se ejecutaron 511 escenarios de estrés divididos en cinco categorías: ruido textual (caracteres especiales, errores ortográficos), volúmenes elevados (múltiples eventos simultáneos), variación cruzada por fuente (mezcla de formatos), secuencias temporales crecientes (patrones temporales), y eventos ambiguos (mensajes poco informativos).

Estas pruebas permitieron evaluar la estabilidad y consistencia del modelo frente a modificaciones drásticas en los atributos de entrada. Los resultados agregados se presentan en la Tabla V.

Escenario	Severidad Promedio	Exactitud
Ambigüedad	2.00	0.0
Cruzado	2.00	-
Ruido textual	2.00	0.0
Secuencial	2.00	33.3 %
Volumen	2.00	23.8 %

Cuadro V
RESUMEN DE PRUEBAS DE ROBUSTEZ DEL MODELO.

A nivel global el análisis produjo las siguientes estadísticas:

- Eventos evaluados: 511
- Eventos con severidad original: 506
- Coincidencias: 120
- Precisión global: 23.7 %

Finalmente, la distribución general de severidades predichas mostró un patrón consistente en el que el modelo clasificó la totalidad de los 511 eventos con severidad *Advertencia (2)*. Si bien este comportamiento garantiza estabilidad ante ruido, evidencia una tendencia hacia la predicción conservadora, lo cuál es una oportunidad de mejora en la diferenciación de niveles extremos.

VI-C. Discusión

Los resultados permiten identificar tres conclusiones principales:

1. En datos reales el modelo obtiene una excelente precisión, preservando adecuadamente los patrones aprendidos.
2. En escenarios simulados heterogéneos, la generalización disminuye, particularmente en los momentos en que los mensajes se alejan de la estructura textual del entrenamiento.
3. En entornos de estrés, el modelo se comporta de manera estable pero conservadora, asignando niveles moderados de severidad. Esto sugiere que la diversidad del texto en el entrenamiento debe ampliarse para mejorar esta discriminación.

Este conjunto de experimentos permite evaluar el desempeño del sistema bajo diferentes condiciones operativas y establece lineamientos concretos para mejorar futuras versiones del pipeline.

VII. Modelo de Madurez de Ciberseguridad

Como parte del fortalecimiento del sistema XDR, se diseñó un Modelo de Madurez de Ciberseguridad para evaluar y mejorar continuamente la postura de seguridad institucional. Este modelo proporciona una metodología estructurada para medir, priorizar y mejorar las capacidades técnicas del SIEM (Security Information and Event Management) y del sistema de correlación, en alineación con los objetivos institucionales de gestión de riesgos y protección de la infraestructura.

El modelo se basa en el NIST (National Institute of Standards and Technology) Cybersecurity Framework (CSF) versión 2.0, el cual establece las actividades de ciberseguridad en seis funciones principales: Gobernar (GV), Identificar (ID), Proteger (PR), Detectar (DE), Responder (RS) y Recuperar (RC). El modelo está en consonancia con las directrices NIST SP (Special Publication) 800-92, NIST SP 800-53 Revisión 5 e ISO/IEC (International Organization for Standardization / International Electrotechnical Commission) 27001, garantizando cumplimiento con estándares internacionales y mejores prácticas de gestión de riesgos.

VII-A. Estructura del modelo

El modelo analiza las capacidades de la plataforma SIEM más XDR desde una perspectiva tanto técnica como organizacional, abarcando desde las capacidades de monitoreo y correlación hasta la gobernanza y respuesta ante incidentes. Se utiliza una escala de cuatro niveles de madurez, alineada con el CSF 2.0:

1. **Nivel 1 - Parcial:** Se caracteriza por procesos ad-hoc dependientes de las personas, con un enfoque reactivo ante incidentes y sin documentación formal de procedimientos.
2. **Nivel 2 - Conocimiento del riesgo:** Las prácticas están parcialmente definidas, existe conciencia de riesgos en la organización, y el enfoque organizacional está centrado en la identificación y evaluación de riesgos.
3. **Nivel 3 - Repetible:** Los procesos están documentados, son medibles y consistentes. La gestión de procesos permite replicar las mejores prácticas y mantener niveles de servicio predecibles.
4. **Nivel 4 - Adaptativo:** Se logra mejora continua mediante automatización e integración avanzada. Los procesos se ajustan dinámicamente basándose en métricas y lecciones aprendidas.

En la primera iteración del modelo se priorizaron las funciones Detectar (DE) y Responder (RS) para alcanzar el nivel 3 Repetible en un plazo de 12 meses. Las funciones Gobernar (GV), Identificar (ID), Proteger (PR) y Recuperar (RC) permanecen en nivel 2 Conocimiento del riesgo, estableciendo políticas, métricas y roles que permitan la sostenibilidad a largo plazo del modelo.

VII-B. Sistema de evidencias

Cada una de las funciones del CSF se evalúa mediante categorías y subcategorías con evidencias verificables que se clasifican en tres tipos principales:

- **Evidencia documental:** Incluye políticas de seguridad formalizadas, procedimientos operativos estándar, actas de revisión del comité de seguridad, registros de riesgos identificados, y reportes de auditoría.
- **Evidencias técnicas:** Comprenden configuraciones activas de Wazuh documentadas, reglas de Suricata implementadas, configuración de triggers en Zabbix, reglas de correlación desplegadas en el XDR, pipelines de datos verificados, y dashboards funcionales en producción.
- **Métricas operativas:** Incluyen Tiempo Medio de Detección (MTTD, por sus siglas en inglés Mean Time To Detect), Tiempo Medio de Recuperación (MTTR, por sus siglas en inglés Mean Time To Recover), cumplimiento de Acuerdo de Nivel de Servicio (SLA, por sus siglas en inglés Service Level Agreement), y porcentaje de cobertura de agentes desplegados.

El proceso de evaluación se lleva a cabo semestralmente en sesiones estructuradas con el equipo de seguridad. Estas evaluaciones permiten comparar el perfil actual o línea base con el perfil meta definido, identificando brechas específicas y priorizando acciones de mejora según su impacto en la reducción de riesgo.

VII-C. Objetivos y métricas por función

El modelo de madurez define objetivos SMART (específicos, medibles, alcanzables, relevantes y temporales) en línea con las funciones del CSF. Estos objetivos se incorporan en los cuadros de mando operativos y se revisan periódicamente para controlar el avance hacia los niveles de madurez definidos:

- **Identificar (ID):** La meta es cubrir al menos el 95 % de los hosts en alcance con agentes de monitoreo desplegados y reportando activamente. Se mide como porcentaje de hosts con agente en estado activo sobre el total de hosts críticos inventariados.
- **Proteger (PR):** El objetivo es implementar Control de Acceso Basado en Roles (RBAC, por sus siglas en inglés Role-Based Access Control) en todas las plataformas de seguridad, integración mediante el protocolo LDAP (Lightweight Directory Access Protocol), y garantizar el cifrado de la información tanto en tránsito mediante TLS (Transport Layer Security) como en reposo. Los indicadores incluyen auditoría de cambios de configuración y verificación del uso de protocolos seguros.
- **Detectar (DE):** Se establece como objetivo un MTTD menor o igual a 3 minutos para incidentes críticos, y mantener activas al menos 10 reglas de correlación priorizadas por riesgo en el servidor XDR. Se mide mediante el MTTD promedio mensual y el número de reglas activas en producción.
- **Responder (RS):** Se busca alcanzar un SLA de notificación de alerta inferior a 1 minuto desde la detección, y

tener playbooks documentados por nivel de severidad en uso activo. Se mide por porcentaje de alertas dentro del SLA y número de playbooks efectivamente utilizados en respuesta a incidentes.

- **Recuperar (RC):** Se realizará una prueba de recuperación ante desastres documentada cada semestre, con evidencia de simulacro ejecutado y registro de tiempos de restauración de servicios críticos.
- **Gobernar (GV):** El propósito es mantener una política de seguridad y monitoreo vigente y aprobada por la dirección, con una revisión formal semestral del modelo de madurez. Se evidencia mediante actas de revisión firmadas y control de versiones del modelo.

VII-D. Plan de acción y roadmap

El plan de acción del modelo de madurez se construye basándose en el análisis del perfil actual versus el perfil meta y las brechas identificadas. Estas brechas se agrupan y priorizan según tres criterios: impacto en la reducción de riesgo, esfuerzo de implementación, y criticidad para la continuidad del negocio.

El roadmap consta de cuatro etapas secuenciales que guían la transformación hacia un entorno gestionado y adaptable:

1. **Fase 1 - Consolidación SIEM (0-2 meses):** Busca integrar todas las fuentes de eventos identificadas y estabilizar la plataforma XDR. Los entregables verificables incluyen agentes desplegados en al menos el 95 % de los hosts críticos, dashboards iniciales funcionales para visualización de eventos, y documentación básica de la arquitectura implementada.
2. **Fase 2 - Correlación y Detección (2-4 meses):** Se enfoca en desarrollar reglas de correlación y alertas procesables. Los entregables incluyen al menos 10 reglas de correlación priorizadas por riesgo y validadas en producción, sistema de alertamiento automatizado configurado con umbrales apropiados, y primeras métricas de MTTD recolectadas.
3. **Fase 3 - Respuesta y Gobierno (4-6 meses):** Formaliza procesos de respuesta y establece gobernanza clara. Los entregables incluyen playbooks de respuesta documentados por nivel de severidad, implementación de RBAC en las plataformas de seguridad, y generación de informes ejecutivos mensuales sobre el estado de seguridad.
4. **Fase 4 - Mejora Adaptativa (6-9 meses):** Implementa optimización continua y capacidades avanzadas. Los entregables incluyen revisión trimestral sistemática de efectividad de reglas de correlación, pruebas de modelos de Machine Learning en escenarios piloto controlados, y automatización de respuestas a incidentes de baja complejidad.

VII-E. Gobernanza y mejora continua

En términos de gobernanza, la Coordinación de Seguridad e Infraestructura de la organización es la responsable del modelo de madurez. Sus funciones incluyen mantener el modelo actualizado conforme evoluciona la infraestructura, liderar las revisiones semestrales de métricas y niveles de madurez, y presentar resultados a la dirección.

La madurez se gestiona como un proceso cíclico siguiendo la metodología PHVA (Planear-Hacer-Verificar-Actuar). En la fase Planear se definen objetivos y se identifican brechas. En Hacer se ejecutan las iniciativas del roadmap. En Verificar se miden las métricas y se valida el cumplimiento de objetivos. En Actuar se integran lecciones aprendidas y se ajustan las metas según nuevas tecnologías o amenazas emergentes.

VII-F. Beneficios esperados

Entre los beneficios principales del modelo se encuentran:

- Disminución del MTTD y del MTTR mediante mejor correlación de eventos y automatización de respuestas.
- Aumento de la cobertura de monitoreo y mejora en la trazabilidad completa de incidentes desde detección hasta resolución.
- Priorización informada de inversiones y esfuerzos técnicos basada en análisis de riesgo cuantificado.
- Institucionalización de la mejora continua mediante ciclos estructurados de evaluación y actualización.
- Mayor rendición de cuentas ante la dirección con métricas objetivas de desempeño en ciberseguridad.

VIII. Discusión

La integración de herramientas open source demuestra que es posible construir una plataforma de monitoreo y seguridad robusta sin depender de soluciones propietarias costosas. La combinación de Zabbix para monitoreo de infraestructura, Wazuh como plataforma XDR, y Suricata como IDS, proporciona capacidades comparables a soluciones comerciales.

La normalización automática de eventos y la correlación basada en reglas permiten obtener contexto adicional para incidentes complejos que involucran múltiples sistemas. Esta capacidad de relacionar eventos de diferentes fuentes resulta fundamental para detectar ataques sofisticados que operan en múltiples etapas.

El modelo predictivo de machine learning evidencia la viabilidad técnica de automatizar la priorización de eventos en entornos de alta carga. Sin embargo, su despliegue en producción requiere consideraciones adicionales como ingestión de datos en tiempo real, validación continua del desempeño del modelo, y controles para evitar sobrecarga del personal con falsos positivos.

El modelo de madurez proporciona un marco estructurado para la evolución continua de las capacidades de ciberseguridad. La alineación con estándares internacionales facilita futuras auditorías y certificaciones, mientras que las métricas cuantitativas permiten justificar inversiones ante la dirección.

IX. Conclusiones

El pipeline desarrollado permitió automatizar de forma efectiva la ingesta, el preprocesamiento y la clasificación de severidad de eventos de monitoreo. El modelo de machine learning demostró un rendimiento excelente en datos reales

de Zabbix con precisión superior al 95 %, y se comportó de manera estable ante simulaciones controladas.

Sin embargo, las pruebas de robustez evidenciaron limitaciones en la capacidad del modelo para generalizar en casos de variaciones textuales significativas. En escenarios con ruido artificial, la precisión disminuyó al 23.7 %, aunque el modelo mantuvo un comportamiento conservador apropiado para evitar subestimar incidentes críticos.

La integración de múltiples herramientas open source (Zabbix, Wazuh, Suricata) resultó técnicamente viable y proporcionó capacidades de detección y respuesta comparables a soluciones comerciales. El modelo de madurez de ciberseguridad ofrece un marco estructurado para la evolución continua del sistema.

En conjunto, los resultados confirman la utilidad del enfoque propuesto pero muestran una necesidad clara de ampliar el conjunto de entrenamiento del modelo de ML y fortalecer la representación de la muestra para mejorar la precisión en escenarios más complejos y diversos.

REFERENCIAS

[1] Zabbix Documentation. Available: https://www.zabbix.com/documentation

[2] Wazuh Documentation. Available: https://documentation.wazuh.com/

[3] Suricata IDS Documentation. Available: https://suricata.io/documentation/

[4] Scikit-learn: Machine Learning in Python. Pedregosa et al., JMLR (2011).

[5] Project repository (clean version): https://github.com/LindsayPinto/Severity_prediction_centralized.git