Transformación Digital y Conocimiento /julio-diciembre 2026/ Vol. 4, - No. 2
Doi: 10.70577/innovascit.v4i2.348

Detección de intrusiones en redes IoT mediante modelos ligeros de aprendizaje automático: revisión técnico-aplicada y propuesta de validación experimental para Esmeraldas, Ecuador

Intrusion detection in IoT networks using lightweight machine-learning models: a technical-applied review and experimental validation proposal for Esmeraldas, Ecuador

RESUMEN

, El crecimiento del Internet de las Cosas (IoT) amplía la superficie de ataque de redes formadas por dispositivos con recursos limitados, heterogeneidad de protocolos y ciclos de actualización reducidos. El objetivo de este estudio fue analizar enfoques de detección de intrusiones basados en aprendizaje automático y formular una arquitectura ligera, reproducible y evaluable para entornos IoT con conectividad y capacidad computacional restringidas. Se realizó una revisión técnico-aplicada de sistemas de detección de intrusiones, selección de características, algoritmos clásicos y profundos, métricas de evaluación, conjuntos de datos IoT y criterios de despliegue en edge/fog. La evidencia muestra que Random Forest, árboles de decisión, SVM, k-NN y modelos de boosting mantienen alta capacidad de detección cuando se combinan con selección de atributos, mientras que arquitecturas profundas ofrecen ventajas en extracción automática de características a costa de mayor memoria, latencia y consumo. Se analizaron N-BaIoT, TON_IoT y CICIoT2023 como referencias experimentales: N-BaIoT incorpora tráfico de nueve dispositivos IoT infectados con Mirai y BASHLITE; TON_IoT integra tráfico de red, telemetría y sistemas operativos en una arquitectura edge-fog-cloud; y CICIoT2023 reúne 33 ataques organizados en siete categorías sobre una topología de 105 dispositivos. Se propone validar modelos ligeros mediante una metodología que separe entrenamiento, validación y prueba por sesiones o dispositivos, evitando fuga de información, y que mida no solo accuracy y F1-score, sino también tasa de falsos positivos, tiempo de inferencia, memoria, uso de CPU y estabilidad frente a cambios de dominio. Como línea aplicada para Esmeraldas, se plantea desplegar el detector en un gateway local y evaluar su funcionamiento aun cuando la conexión hacia servicios remotos sea intermitente. Se concluye que la utilidad de un IDS ligero depende de equilibrar capacidad de detección, costo computacional, explicabilidad y generalización a tráfico no observado.

PALABRAS CLAVE: Internet de las Cosas; IoT; detección de intrusiones; aprendizaje automático; ciberseguridad; edge computing; IDS ligero; Esmeraldas.

ABSTRACT

The growth of the Internet of Things (IoT) expands the attack surface of networks composed of resource-constrained devices, heterogeneous protocols, and limited update cycles. This study analyzes machine-learning-based intrusion detection approaches and formulates a lightweight, reproducible architecture for IoT environments with limited connectivity and computing capacity. A technical-applied review was conducted covering intrusion detection systems, feature selection, classical and deep-learning algorithms, evaluation metrics, IoT datasets, and edge/fog deployment criteria. Evidence indicates that Random Forest, decision trees, SVM, k-NN, and boosting models can provide strong detection performance when combined with feature selection, while deep architectures can automate representation learning at the cost of greater memory, latency, and processing demand. N-BaIoT, TON_IoT, and CICIoT2023 were analyzed as experimental benchmarks. A local validation methodology is proposed that separates training, validation, and testing by sessions or devices to prevent data leakage and evaluates accuracy, F1-score, false-positive rate, inference time, memory footprint, CPU usage, and cross-domain stability. For Esmeraldas, the proposed deployment places the detector at a local edge gateway so that security monitoring can continue during intermittent upstream connectivity. A useful lightweight IDS must balance detection capability, computational cost, explainability, and generalization to previously unseen traffic.

KEYWORDS: Internet of Things; IoT; intrusion detection; machine learning; cybersecurity; edge computing; lightweight IDS; Esmeraldas.

TRANSFORMACIÓN DIGITAL Y CONOCIMIENTO

Recepción: 04/10 /2026

Aceptación: 29 /10/2026

Publicación: 31/12/2026

AUTOR/ES

Jair Oswaldo Bedoya Benavides

jair.bedoya.benavides@utelvt.edu.ec

Universidad Técnica "Luis Vargas Torres" de Esmeraldas, Ecuador

Esmeraldas - Ecuador

CITACIÓN:

Bedoya, J. (2026). Detección de intrusiones en redes IoT mediante modelos ligeros de aprendizaje automático: revisión técnico-aplicada y propuesta de validación experimental para Esmeraldas, Ecuador. Revista InnovaSciT. *4 (2,). p. 1896 – 1910.*

Doi: 10.70577/innovascit.v4i2.348

INTRODUCCIÓN

El Internet de las Cosas conecta sensores, actuadores, cámaras, medidores, sistemas domésticos e infraestructura industrial mediante redes heterogéneas. Esta expansión mejora la observabilidad y automatización de procesos, pero también incorpora dispositivos con firmware desactualizado, credenciales débiles, recursos limitados y exposición continua a redes públicas, lo que incrementa la superficie de ataque (Sicari et al., 2015; Al-Garadi et al., 2020).

Los sistemas de detección de intrusiones (IDS) constituyen una capa complementaria a controles preventivos como autenticación, segmentación, cifrado y gestión de actualizaciones. Un IDS basado en firmas es eficaz frente a patrones conocidos, mientras que un sistema de detección de anomalías puede reconocer desviaciones del comportamiento habitual y, potencialmente, ataques no incluidos en reglas previas. Sin embargo, la detección de anomalías también puede generar mayor tasa de falsos positivos y requiere modelos que se adapten al comportamiento específico de cada entorno (Moustafa et al., 2021; Rahman et al., 2025).

El aprendizaje automático ha adquirido relevancia en este ámbito porque permite clasificar grandes volúmenes de tráfico y encontrar relaciones entre múltiples atributos. No obstante, en IoT no basta con maximizar accuracy: un modelo puede obtener resultados elevados en laboratorio y resultar poco útil si consume demasiada memoria, introduce latencia, no generaliza a otros dispositivos o depende permanentemente de conectividad hacia la nube. Las revisiones recientes destacan por ello la necesidad de IDS ligeros y evaluaciones que incorporen recursos computacionales además de métricas de clasificación (Khan et al., 2024; IEEE ICCST, 2025).

Otro problema recurrente es la metodología de validación. Cuando segmentos del mismo flujo o de la misma sesión aparecen simultáneamente en entrenamiento y prueba, el clasificador puede memorizar patrones específicos y producir resultados artificialmente altos. La división aleatoria tradicional debe reemplazarse, cuando sea posible, por particiones por dispositivo, sesión, escenario o ventana temporal, especialmente cuando se pretende estimar generalización a tráfico futuro.

Los conjuntos de datos especializados han permitido comparar métodos. Meidan et al. (2018) construyeron N-BaIoT con nueve dispositivos comerciales infectados con Mirai y BASHLITE. Moustafa et al. (2021) desarrollaron TON_IoT en una arquitectura que integra edge, fog y cloud con tráfico de red, telemetría y datos de sistemas operativos. CICIoT2023 amplió la escala a 105 dispositivos y 33 ataques agrupados en siete categorías, proporcionando un escenario útil para clasificación binaria y multiclase (Neto et al., 2023).

En la provincia de Esmeraldas, los escenarios rurales y periurbanos pueden presentar heterogeneidad de dispositivos y conectividad irregular, por lo que una arquitectura

Doi: 10.70577/innovascit.v4i2.348

dependiente exclusivamente de servicios remotos puede perder capacidad de monitoreo durante interrupciones. Desde Tecnologías de la Información, esto justifica estudiar modelos que puedan ejecutarse localmente en un gateway, enviar únicamente alertas o resúmenes y conservar la operación de seguridad aun con conectividad intermitente.

El aporte del presente artículo no consiste en proponer otro clasificador basado únicamente en accuracy. La propuesta articula selección de características, costo computacional, evaluación de generalización, explicabilidad y despliegue en edge como criterios simultáneos de decisión.

En este contexto, el objetivo de este estudio fue analizar críticamente modelos de aprendizaje automático aplicados a detección de intrusiones IoT y formular un protocolo experimental reproducible para seleccionar un IDS ligero que pueda validarse en un entorno de laboratorio y posteriormente en escenarios de conectividad limitada en Esmeraldas.

METODOLOGÍA Y MATERIALES

El estudio se concibió como una revisión técnico-aplicada orientada a la toma de decisiones de arquitectura. Más que comparar algoritmos por una cifra aislada de accuracy, se examinó qué combinación de datos, preprocesamiento, modelo y plataforma permite construir un detector de intrusiones IoT con bajo costo computacional y capacidad de operar en un gateway. La heterogeneidad de datasets y protocolos de evaluación impidió realizar una síntesis estadística conjunta de resultados.

Se incorporaron investigaciones sobre IDS para IoT, selección de características, clasificación tabular, aprendizaje profundo y despliegue en edge o fog. N-BaIoT, TON_IoT y CICIoT2023 se tomaron como referentes porque describen dispositivos, ataques y condiciones de captura con suficiente detalle para discutir reproducibilidad. También se consideraron trabajos metodológicos sobre generalización y fuga de información, dado que una partición inadecuada puede sobreestimar el rendimiento de un clasificador.

La literatura se examinó desde cuatro dimensiones: calidad del conjunto de datos, desempeño de detección, costo de inferencia y capacidad de transferencia. En la primera se revisaron clases, balance, preprocesamiento y estrategia de partición; en la segunda, precision, recall, F1-score y tasa de falsos positivos; en la tercera, latencia, memoria, CPU y tamaño del modelo; y en la cuarta, comportamiento frente a dispositivos, sesiones o escenarios no observados durante el entrenamiento.

Los hallazgos se integraron para identificar modelos que mantengan un equilibrio entre detección, ligereza e interpretabilidad. La propuesta resultante separa una fase de comparación reproducible con datasets públicos y una fase de validación en un laboratorio IoT local. El criterio de selección final no depende de una sola métrica, sino del comportamiento conjunto del modelo y de su viabilidad para continuar operando en un gateway aun cuando la conectividad externa sea intermitente.

Doi: 10.70577/innovascit.v4i2.348

ANÁLISIS DE RESULTADOS

Características de los IDS para IoT

Un IDS para IoT puede ubicarse en el dispositivo, gateway, edge/fog o nube. Los dispositivos finales ofrecen máxima proximidad pero recursos muy restringidos; la nube dispone de mayor capacidad pero depende de conectividad y agrega latencia. El gateway o nodo edge constituye un compromiso útil porque concentra tráfico de varios dispositivos, permite ejecutar modelos clásicos o compactos y puede mantener funciones básicas aun sin conexión permanente a Internet.

Tabla 3

Datasets relevantes para evaluación de IDS IoT

Dataset	Características	Fortalezas	Limitaciones
N-BaIoT	9 dispositivos IoT; Mirai y BASHLITE; tráfico benigno y malicioso	Tráfico de dispositivos reales y botnets conocidas	Escala y diversidad de ataques más limitada que datasets recientes.
TON_IoT	Red, telemetría, Windows/Linux; arquitectura edge-fog-cloud	Heterogeneidad de fuentes y escenarios	Mayor complejidad de preparación y distintas modalidades de datos.
CICIoT2023	105 dispositivos; 33 ataques; 7 categorías	Escala amplia y taxonomía de ataques moderna	Volumen elevado; riesgo de evaluar solo clasificación offline.

Nota. Fuentes: Meidan et al. (2018), Moustafa et al. (2021) y Neto et al. (2023).

Algoritmos clásicos y costo computacional

Los algoritmos clásicos mantienen relevancia porque pueden ofrecer buen rendimiento con menor complejidad de inferencia. Los árboles de decisión son interpretables; Random Forest mejora estabilidad a costa de multiplicar árboles; SVM puede resultar efectivo en espacios de alta dimensión pero escalar peor con grandes volúmenes; k-NN desplaza costo hacia la inferencia; y boosting ofrece alta capacidad predictiva con complejidad dependiente del número y profundidad de los estimadores.

Tabla 4

Comparación conceptual de modelos para un IDS ligero

Modelo	Ventaja	Costo/limitación	Uso propuesto
Decision Tree	Alta	Puede sobreajustar	Línea base

Doi: 10.70577/innovascit.v4i2.348

	interpretabilidad y baja latencia		interpretable.
Random Forest	Robusto y buen desempeño tabular	Mayor memoria que un árbol	Candidato principal en gateway.
SVM	Buen margen de separación	Entrenamiento y escalamiento costosos en datasets grandes	Comparación con variables seleccionadas.
k-NN	Simple y sin entrenamiento complejo	Inferencia costosa y memoria alta	Benchmark, no primera opción edge.
Gradient Boosting/XGBoost	Alta precisión en datos tabulares	Más hiperparámetros y costo	Candidato cuando la latencia sea aceptable.
MLP/CNN	Representaciones no lineales complejas	Mayor memoria y cómputo	Referencia, no necesariamente modelo final.

Nota. La selección final debe basarse en métricas de detección y costo medido en el hardware objetivo.

3.4. Selección de características

La reducción de atributos puede disminuir tiempo de entrenamiento, memoria y latencia, además de mejorar interpretabilidad. Las revisiones sobre IDS ligeros señalan el uso frecuente de métodos filter como correlación, información mutua y pruebas estadísticas, así como técnicas wrapper o embedded. Para un prototipo reproducible se propone comparar un conjunto completo frente a subconjuntos obtenidos por importancia de Random Forest, información mutua y eliminación de atributos altamente correlacionados.

Métricas de evaluación

En datasets desbalanceados, accuracy puede ocultar clases minoritarias. Por ello se deben reportar precision, recall, F1-score y tasa de falsos positivos:

$$Precision = TP / (TP + FP)$$

$$Recall = TP / (TP + FN)$$

$$F1 = 2 \cdot (Precision \cdot Recall) / (Precision + Recall)$$

$$FPR = FP / (FP + TN)$$

Para clasificación multiclase se deben reportar promedios macro y weighted, además de matriz de confusión. En un IDS operativo, un FPR pequeño es especialmente relevante porque un volumen elevado de falsas alarmas reduce la utilidad del sistema.

Doi: 10.70577/innovascit.v4i2.348

Riesgo de fuga de información

Una debilidad frecuente consiste en dividir aleatoriamente filas cuando existen múltiples registros muy similares de una misma sesión. Esto puede colocar muestras casi duplicadas en entrenamiento y prueba. Para estimar mejor el desempeño real se recomienda separar por captura, sesión, dispositivo o bloque temporal; cuando el dataset lo permita, una prueba cross-device o cross-scenario aporta evidencia adicional de generalización.

Explicabilidad

En entornos institucionales o productivos, una alerta debe poder relacionarse con variables de tráfico comprensibles. La importancia de variables, SHAP u otras técnicas de explicación pueden ayudar a identificar qué atributos influyeron en la clasificación. Sin embargo, la explicación no debe sustituir la validación: una característica muy influyente puede reflejar un artefacto del dataset en lugar de un patrón general de ataque.

Despliegue en edge y conectividad limitada

Un esquema edge reduce la necesidad de transmitir continuamente todo el tráfico a la nube. El gateway puede extraer características, ejecutar la inferencia y conservar temporalmente alertas. Cuando existe conectividad, se sincronizan eventos, métricas y actualizaciones de modelo. Esta arquitectura favorece continuidad operativa y disminuye ancho de banda, aunque exige controlar almacenamiento local, seguridad del gateway y actualización del modelo.

DISCUSIÓN

Precisión versus utilidad operativa

Un modelo con 99 % de accuracy no es automáticamente superior a otro con 97 % si requiere diez veces más memoria, duplica la latencia o genera más falsas alarmas. Las comparaciones publicadas muestran resultados muy elevados en datasets conocidos, por lo que el desafío actual se desplaza hacia generalización, costo y reproducibilidad. Las revisiones recientes insisten en que los IDS ligeros deben evaluar CPU, memoria y energía además del desempeño de clasificación (Khan et al., 2024; Rahman et al., 2025).

Qué modelos priorizar

Para un primer prototipo en la Carrera de Tecnologías de la Información se recomienda comenzar con Decision Tree, Random Forest, SVM y un método de boosting. Estos algoritmos permiten comparar capacidad predictiva y costo sin introducir de entrada la complejidad de arquitecturas profundas. Si un modelo clásico satisface los requisitos de detección y latencia, una red profunda más costosa necesitaría demostrar una ventaja operativa clara para justificar su adopción.

Pertinencia territorial

El valor territorial de la propuesta se encuentra en la posibilidad de ejecutar la detección cerca de la fuente de datos y mantenerla aun cuando la conexión externa sea

Doi: 10.70577/innovascit.v4i2.348

irregular. No se afirma que los ataques presentes en N-BaIoT, TON_IoT o CICIoT2023 representen exactamente el perfil de amenaza de Esmeraldas; estos datasets se utilizarían como referencia para desarrollar y comparar modelos. La fase local debe capturar tráfico legítimo de un laboratorio IoT y generar únicamente escenarios de ataque autorizados y aislados.

Implicaciones para Tecnologías de la Información y propuesta de validación experimental

La propuesta experimental se organiza en dos fases. La primera utiliza datasets públicos para seleccionar un modelo y un subconjunto de características. La segunda valida el modelo en una red IoT de laboratorio con gateway local, sin exponer sistemas de producción ni ejecutar ataques fuera de un entorno controlado.

Fase 1: comparación reproducible con datasets públicos

Tabla 5

Diseño experimental de la fase 1

Elemento	Propuesta
Datasets	N-BaIoT, TON_IoT y/o CICIoT2023.
Tareas	Clasificación binaria benigno/ataque y, posteriormente, multiclase.
Modelos	Decision Tree, Random Forest, SVM y Gradient Boosting/XGBoost.
Selección	Sin selección vs. información mutua/importancia/correlación.
Partición	Por sesión, dispositivo o bloque cuando la estructura del dataset lo permita.
Métricas	Precision, recall, macro-F1, FPR, matriz de confusión, tiempo y memoria.
Repetición	Semillas documentadas y ≥5 ejecuciones cuando exista aleatoriedad.

Nota. El objetivo es medir simultáneamente eficacia y costo computacional.

5.2. Fase 2: banco IoT local

Tabla 6

Arquitectura propuesta para validación local

Componente	Función
Dispositivos IoT	Sensores y actuadores de laboratorio generando tráfico legítimo.
Punto de acceso/switch	Segmentar la red de prueba.

Doi: 10.70577/innovascit.v4i2.348

Gateway edge	Capturar metadatos/flows y ejecutar el IDS.
Servidor de gestión	Entrenar modelos, almacenar registros y analizar resultados.
Sincronización	Enviar alertas/resúmenes cuando exista conectividad.
Registro local	Mantener eventos durante periodos sin conexión externa.

Nota. El entorno debe permanecer aislado de redes institucionales durante cualquier prueba de ataque.

Escenarios de prueba

Tabla 7

Escenarios experimentales propuestos

Código	Escenario	Propósito
E0	Tráfico normal estable	Establecer línea base y FPR.
E1	Cambio de carga legítima	Evaluar si el modelo confunde picos normales con ataques.
E2	Escenarios maliciosos autorizados del laboratorio	Evaluar detección en un entorno aislado.
E3	Interrupción de Internet con red local activa	Verificar continuidad del IDS edge.
E4	Dispositivo no visto en entrenamiento	Evaluar transferencia entre dispositivos.

Nota. Las pruebas de seguridad deben limitarse a un laboratorio controlado y autorizado.

Protocolo de preprocesamiento

Se eliminarán identificadores que funcionen como etiquetas indirectas del escenario cuando no sean generalizables, se documentará el tratamiento de valores faltantes y se aplicará codificación consistente a variables categóricas. El escalado se ajustará exclusivamente con datos de entrenamiento. Cualquier técnica de balanceo debe aplicarse después de separar entrenamiento y prueba para evitar contaminación.

Doi: 10.70577/innovascit.v4i2.348

Optimización multirrespuesta

Tabla 8

Criterios para seleccionar el modelo final

Criterio	Dirección deseada
Macro-F1	Maximizar
Recall de ataques	Maximizar
FPR	Minimizar
Latencia por flujo	Minimizar
Memoria del modelo	Minimizar
CPU	Minimizar
Número de características	Minimizar sin perder desempeño
Generalización cross-device	Maximizar
Interpretabilidad	Maximizar cuando el desempeño sea comparable.

Nota. La selección no debe basarse en una única métrica.

Medición de ligereza

El adjetivo 'ligero' debe demostrarse. Para cada modelo se registrarán tamaño serializado, memoria máxima durante inferencia, tiempo medio y percentil 95 de inferencia, uso de CPU y, si el hardware lo permite, consumo energético. Estas mediciones se realizarán en el mismo gateway y con el mismo lote de entradas.

Plan estadístico

Las métricas de clasificación se reportarán con intervalos obtenidos a través de repeticiones o particiones independientes. Para comparar tiempos de inferencia entre modelos se verificará la distribución y se aplicará ANOVA/Tukey o Kruskal-Wallis según corresponda. En clasificación se priorizarán diferencias en macro-F1 y FPR; no se interpretarán como superiores diferencias mínimas sin considerar su variabilidad.

Arquitectura de despliegue propuesta

El gateway recibirá metadatos de tráfico o flujos, aplicará el mismo pipeline de preprocesamiento utilizado durante el entrenamiento y ejecutará el clasificador. Las alertas se almacenarán localmente con marca temporal, clase estimada, nivel de confianza y variables explicativas. La comunicación con un servidor central será asíncrona: la pérdida temporal de Internet no detendrá la inferencia local. El servidor podrá utilizarse para reentrenamiento, análisis histórico y distribución controlada de nuevas versiones.

Criterios de seguridad y reproducibilidad

• Separar físicamente o mediante VLAN el banco de prueba de redes institucionales y de producción.

• Registrar versión de dataset, código, semillas aleatorias, hiperparámetros y lista exacta de

Doi: 10.70577/innovascit.v4i2.348

características.

• No utilizar datos de prueba para seleccionar hiperparámetros ni umbrales.

• Conservar matrices de confusión y resultados por clase, no únicamente métricas agregadas.

• Publicar, cuando sea posible, scripts de preprocesamiento y entrenamiento sin credenciales ni información sensible.

• Repetir la evaluación sobre al menos un dataset adicional o sobre tráfico local para estimar transferencia.

Perfilado reproducible y evaluación de ataques no observados

La prueba en gateway deberá registrar arquitectura de hardware, sistema operativo, versión de Python, conjunto de características, tamaño serializado del modelo y condiciones de carga. Para medir memoria residente y utilización de CPU pueden emplearse herramientas de sistema y bibliotecas como psutil; perf puede aportar contadores de rendimiento en sistemas compatibles y tracemalloc facilita estudiar asignaciones de memoria dentro de Python, pero no equivale por sí solo a medir toda la RAM consumida por el proceso. La latencia se evaluará mediante reloj monótono de alta resolución, después de calentamiento, informando mediana, percentiles y número de repeticiones.

La evaluación de amenazas no observadas requiere separar familias de ataques o dispositivos completos entre entrenamiento y prueba, además de contrastar detección binaria de anomalías con clasificación multiclase. Un ataque omitido del entrenamiento puede servir para estudiar generalización a una familia no vista, pero no demuestra detección universal de vulnerabilidades zero-day. Se reportarán recall por familia, falsos positivos sobre tráfico legítimo nuevo, degradación frente a cambio de dominio y costo de actualización del modelo.

En condiciones de conectividad intermitente, el gateway almacenará eventos de forma local con límites de memoria y políticas de retención definidos. El protocolo verificará continuidad de inferencia, recuperación tras desconexión y sincronización de alertas sin exponer datos sensibles innecesarios.

Doi: 10.70577/innovascit.v4i2.348

CONCLUSIONES

Los IDS basados en aprendizaje automático son una alternativa pertinente para redes IoT, pero su utilidad depende de la calidad metodológica de la validación y no únicamente de la accuracy reportada.

N-BaIoT, TON_IoT y CICIoT2023 ofrecen escenarios complementarios para desarrollar y comparar detectores, aunque ninguno representa por sí solo todos los entornos IoT.

Random Forest, Decision Tree, SVM y métodos de boosting constituyen candidatos adecuados para una primera evaluación de modelos ligeros por su desempeño en datos tabulares y menor complejidad respecto a muchas arquitecturas profundas.

La selección de características puede reducir latencia, memoria y costo sin comprometer necesariamente la capacidad de detección, por lo que debe formar parte del experimento.

La división de datos por sesión, dispositivo o escenario ofrece una estimación más rigurosa de generalización que una partición aleatoria de registros altamente relacionados.

Un IDS ligero debe reportar macro-F1, recall, FPR, latencia, memoria y CPU; utilizar únicamente accuracy no es suficiente.

El despliegue en un gateway edge es especialmente apropiado cuando se busca continuidad de monitoreo bajo conectividad externa irregular.

La propuesta permite evolucionar posteriormente desde una revisión técnico-aplicada hacia un artículo experimental original con resultados reproducibles obtenidos en la UTLVTE.

REFERENCIAS BIBLIOGRÁFICAS

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for Internet of Things (IoT) security. IEEE Communications Surveys & Tutorials, 22(3), 1646–1685. https://doi.org/10.1109/COMST.2020.2988293

Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., & Razaque, A. (2020). Deep recurrent neural network for IoT intrusion detection system. Simulation Modelling Practice and Theory, 101, 102031. https://doi.org/10.1016/j.simpat.2019.102031

Alrawashdeh, K., & Purdy, C. (2016). Toward an online anomaly intrusion detection system based on deep learning. 2016 15th IEEE International Conference on Machine Learning and Applications, 195–200. https://doi.org/10.1109/ICMLA.2016.0040

Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32. https://doi.org/10.1023/A:1010933404324

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20, 273–297. https://doi.org/10.1007/BF00994018

Doshi, R., Apthorpe, N., & Feamster, N. (2018). Machine learning DDoS detection for consumer Internet of Things devices. 2018 IEEE Security and Privacy Workshops, 29–35. https://doi.org/10.1109/SPW.2018.00013

Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2020). DDoSNet: A deep-learning model for detecting network attacks. 2020 IEEE 21st International Symposium on 'A World of Wireless, Mobile and Multimedia Networks'.

Ferrag, M. A., Maglaras, L., Ahmim, A., Derdour, M., & Janicke, H. (2020). RDTIDS: Rules and decision tree-based intrusion detection system for Internet-of-Things networks. Future Internet, 12(3), 44.

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50, 102419. https://doi.org/10.1016/j.jisa.2019.102419

Guo, C., Zhou, Y., Ping, Y., Zhang, Z., Liu, G., & Yang, Y. (2014). A distance sum-based hybrid method for intrusion detection. Applied Intelligence, 40, 178–188.

Hall, M. A. (1999). Correlation-based feature selection for machine learning [Doctoral dissertation, University of Waikato].

Ho, T. K. (1995). Random decision forests. Proceedings of the Third International Conference

on Document Analysis and Recognition, 278–282.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30.

Khan, M. A., et al. (2024). A systematic literature review of recent lightweight detection approaches leveraging machine and deep learning mechanisms in Internet of Things networks. Journal of King Saud University - Computer and Information Sciences, 36(1), 101866. https://doi.org/10.1016/j.jksuci.2023.101866

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100, 779–796. https://doi.org/10.1016/j.future.2019.05.041

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. Informatica, 31, 249–268.

Liu, H., & Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence, 388–391.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.

Mahdavifar, S., & Ghorbani, A. A. (2019). Application of deep learning to cybersecurity: A survey. Neurocomputing, 347, 149–176. https://doi.org/10.1016/j.neucom.2019.02.056

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2018). N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12–22. https://doi.org/10.1109/MPRV.2018.03367731

Mirsky, Y., Doitshman, T., Elovici, Y., & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. Network and Distributed System Security Symposium. https://doi.org/10.14722/ndss.2018.23204

Moustafa, N., et al. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. Sustainable Cities and Society, 72, 102994. https://doi.org/10.1016/j.scs.2021.102994

Moustafa, N., Keshk, M., Debie, E., & Janicke, H. (2020). Federated TON_IoT Windows datasets for evaluating AI-based security applications. 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications. https://doi.org/10.1109/TrustCom50675.2020.00114

Moustafa, N., Turnbull, B., & Choo, K.-K. R. (2019). An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things. IEEE Internet of Things Journal, 6(3), 4815–4830. https://doi.org/10.1109/JIOT.2018.2871719

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors, 23(13), 5941. https://doi.org/10.3390/s23135941

Pajouh, H. H., Javidan, R., Khayami, R., Dehghantanha, A., & Choo, K.-K. R. (2016). A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Transactions on Emerging Topics in Computing, 7(2), 314–323. https://doi.org/10.1109/TETC.2016.2633228

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1, 81–106.

Rahman, M. M., Al Shakil, S., & Mustakim, M. R. (2025). A survey on intrusion detection system in IoT networks. Cyber Security and Applications, 3, 100082. https://doi.org/10.1016/j.csa.2024.100082

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should I trust you? Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144. https://doi.org/10.1145/2939672.2939778

Shafiq, M., Tian, Z., Bashir, A. K., Du, X., & Guizani, M. (2020). IoT malicious traffic identification using wrapper-based feature selection mechanisms. Computers & Security, 94, 101863. https://doi.org/10.1016/j.cose.2020.101863

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proceedings of ICISSP 2018, 108–116. https://doi.org/10.5220/0006639801080116

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50. https://doi.org/10.1109/TETCI.2017.2772792

Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. Computer Networks, 76, 146–164. https://doi.org/10.1016/j.comnet.2014.11.008

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

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., & de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. Journal of Network and Computer Applications, 84, 25–37. https://doi.org/10.1016/j.jnca.2017.02.009

CONFLICTO DE INTERÉS:
Los autores declaran que no existen conflicto de interés posibles.

FINANCIAMIENTO

No existió asistencia de financiamiento de parte de pares externos al presente artículo.

NOTA:

El articulo no es producto de una publicación anterior