Scientific Research Publishing

Int. J. Communications, Network and System Sciences, 2026, 19(1), 1-11
https://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

Mitigating Cybersecurity Risks in LAN Environments Using AI Techniques

Mohammed Gambo[1], Kenneth Riddle[1], Olatunde Abiona[1], Clement Onime[2]

[1]Department of Computer Information Systems, Indiana University Northwest, Gary, USA
[2]Information & Communication Technology Section, International Center for Theoretical Physics, ICTP, Trieste, Italy
Email: mgambo@iu.edu, koriddle@iu.edu, oabiona@iu.edu, onime@ictp.it

How to cite this paper: Gambo, M., Riddle, K., Abiona, O. and Onime, C. (2026) Mitigating Cybersecurity Risks in LAN Environments Using AI Techniques. *Int. J. Communications, Network and System Sciences*, 19, 1-11.
https://doi.org/10.4236/ijcns.2026.191001

Received: November 21, 2025
Accepted: January 17, 2026
Published: January 20, 2026

Abstract

Local Area Networks (LANs) are critical to organizational infrastructure, yet they remain highly vulnerable to sophisticated cyber threats such as insider misuse, ARP spoofing, privilege escalation, and zero-day exploits. Traditional defenses including firewalls, signature-based intrusion detection systems, and static authentication protocols are increasingly inadequate for dynamic and evolving attack vectors. Artificial Intelligence (AI) and Large Language Models (LLMs) offer new opportunities to strengthen LAN security through anomaly detection, contextual analysis, and adaptive response. This paper explores how machine learning (ML), deep learning (DL), and LLM-powered systems (e.g., ChatGPT, Gemini, Claude, Copilot, Falcon) can mitigate cybersecurity risks in LAN environments. It also examines the integration of AI with advanced authentication technologies such as Multi-Factor Authentication (MFA), biometric verification, behavioral biometrics, adaptive access control, and Zero Trust Architecture (ZTA). A hybrid framework is proposed that combines anomaly detection and mitigation of network security risks, 'LLM assisted decision making, and AI enhanced authentication to detect and mitigate against security risks.

Keywords

Cyber Threats, LAN, MFA, Zero Trust Architecture, Artificial Intelligence, LLM, ML, DL

1. Introduction

Local Area Networks (LANs) are the backbone of modern connectivity, enabling communication, collaboration, and resource sharing across organizational and home environments. From academic institutions and government agencies to

healthcare systems and financial organizations and private homes, LANs remain central to daily operations. However, this critical reliance also makes them prime targets for cyberattacks. The security of LAN infrastructures is increasingly threatened by insider misuse, malware injection, spoofing attacks, and zero-day vulnerabilities that bypass conventional defenses.

Traditional LAN security frameworks primarily rely on firewalls, signature-based intrusion detection systems (IDS), and access control mechanisms. While effective against known attack signatures, these methods are limited in detecting or adapting to threats. Moreover, as networks expand to support remote work, Internet of Things (IoT) devices, and cloud integration, LAN attack surfaces have grown significantly, further exposing weaknesses in static defense models.

Artificial Intelligence (AI), particularly machine learning (ML) and deep learning (DL), offers promising solutions for enhancing cybersecurity. Unlike rule-based systems, AI-driven tools can learn patterns of normal behavior, identify anomalies, and adapt to evolving attack strategies. More recently, the increasing use of Large Language Models (LLMs) such as ChatGPT, Gemini, Claude, Copilot, and Falcon necessitates the introduction of novel approaches to cybersecurity. These models extend AI capabilities beyond anomaly detection to include natural language log interpretation, contextual analysis, and direct interaction with human analysts.

This paper examines how AI and LLM techniques can help mitigate cybersecurity attacks in LAN environments by combining anomaly detection, contextual reasoning, and adaptive authentication. In addition, the paper proposes a hybrid framework that merges AI-driven detection with modern authentication technologies such as Multi-Factor Authentication (MFA), biometric verification, behavioral biometrics, adaptive access control, and Zero Trust Architecture.

2. Related Work

Cybersecurity related problems are common on LANs due to the the nature of the network and use. In Mirsk et all, Kitsume was developed. Kitsume is a plug and play intrusion detection system designed for live LAN traffic, instead of requiring labeled attack data, it learns normal traffic by itself (unsupervised) and raises an alert whenever traffic looks unusual. It is engineered to run in real time on low-power devices (e.g., Raspberry Pi), so it can sit close to traffic flows without requirement for heavy servers [1]. Kitsune proves that online, label-free detection on a LAN is realistic. In testing, it detects a variety of attacks (e.g., ARP spoofing, bot traffic) with some accuracy like more expensive offline system(s). It can run on low-cost hardware without huge training jobs, which matters for schools, small businesses, and any environment that can't deploy massive security appliances. The system adapts to the local network's "personality" instead of relying on generic signatures. Finally, because it looks at traffic as it happens, detection can occur before damage spreads, which is vital for limiting lateral movement inside a LAN [1]. Despite the strengths, Kitsune shows the common challenges of unsu-

pervised detection. First, concept drift: as legitimate network behavior naturally changes (new applications, new devices, seasonal patterns), a model trained on "yesterday's normal" can raise false positives today. Managing thresholds and update schedules is non-trivial, and without that tuning analysts may be flooded. Second, it does not generate user friendly messages: the system can say "this looks weird", but it doesn't give the analyst why in plain language which limits its use by non-technical operators. That slows triage and can reduce trust. Third, Kitsune stops at detection. It doesn't connect its alerts to triggers or identity-aware actions like just-in-time MFA, device quarantine, or dynamic network segmentation. In other words, it sounds the alarm but does not close the loop to contain the threat on the LAN [1].

In Shone et all, the authors build a deep autoencoder and a Random Forest pipeline to detect intrusions. The main promise is to let the model learn features automatically instead of handcrafting them, then use a strong classifier to make final decisions, [2] exemplifying why deep feature learning matters. On common IDS datasets, the learned features allow the model to spot attacks more accurately than traditional methods that rely on fixed, manually engineered features. The study also helped push the field away from one-size-fits-all features toward data-driven representation learning. Another useful outcome that can be adapted is the hybrid design combining deep representation and classical classifier into a flexible design capable of delivering practical performance gains without requiring the heaviest deep networks [2]. The main issue is generalization to modern LANs. Many results depend on older laboratory style datasets (e.g., KDD/NSL-KDD) that don't reflect today's encrypted traffic, cloud services, or IoT noise. Also, the evaluations are mostly offline: train on a dataset, test on a dataset not continuously on a live LAN where traffic changes and latency matters. Finally, like many pure detections works, this paper does not discuss what to do after an alert: there is no built-in pathway to trigger identity checks (MFA), least-privilege adjustments, or quarantine. So, operators still have to address the "now what?" problem once something is flagged [2].

The paper by Du *et al.* (2017) presented DeepLog, which treats logs the way we read sentences: first they turn raw log lines into stable templates, then they train Long Short-Term Memory (LSTM) model to learn the usual order in which those templates appear during healthy operation. When the system later sees a sequence that no longer "fits the grammar" of normal activity, it flags the moment where the story goes off track and points the analyst to that spot. It turns a flood of log messages into a clear timeline you can read. For example, you might see failed admin login → unusual service starts → sudden spike of new connections, a sequence that normally never happens. That makes it easier to spot what's wrong fast and decide what to check first. Some of its limitations are that most results come from curated datasets cleaner than a real LAN, the non-plain English LSTM output still requires interpretation by an adept or technical reader or operator, and the work stops at detection it doesn't by itself trigger MFA, segment a device,

or open a ticket with policy context [3].

The paper written by Halvorsen *et al.* (2024), examined a simple question: "does generative models like Variational Autoencoders and Generative Adversarial Network really help intrusion detection?" using a survey. Their results show that when labeled attacks are rare, synthesizing realistic examples or learning compact representations can make detectors less brittle and extend what they can "see". The work highlights the need to keep the focus practical as good data is just as important as the algorithm and also tests environments should mimic really messy (often encrypted) enterprise networks, as well as the need for alerts to plug straight into day-to-day Security Operations Center workflow. Some of its limitations are shared across the literature and include benchmark-centric, say little about privacy in logs or how to plug into Network Access Control/Identity Provider /Endpoint Detection & Response, and rarely connect a risk score to a concrete, reversible action that can be audited. Importantly, work implicitly shows the need for systems that are capable of pairing detection with non-technical explanation(s) and safe enforcement [4].

In a foundational work written by Paxson (1999). A system initially called Bro now referred to as Zeek, showed that you can sit quietly on a fast link, decode protocols, and emit high-level security events in near real time. Its lasting contribution is the split between low-level packet mechanism and a flexible policy scripting layer, that allows the modification of detection logic without touching the wire-level plumbing. Zeek logs still anchor many Security Operations Centers due to its ability to turn raw traffic into readable artifacts for hunting and response. Some of its limitations follow from that design, being policy and protocol-driven, it excels at known behaviors and forensics but is not, by itself, the best tool for novel patterns without extra analytics; also, it focuses on detection/logging rather than identity-aware enforcement. And as encrypted traffic grows, its detection narrows unless it is paired with endpoint/identity signals or behavior models over metadata [5].

The 2021 paper written by Guo *et al.* (2021) presented LogBERT, a system that extends the "logs as language" idea by swapping Recurrent Neural Networks for a BERT-style Transformer trained in a self-supervised way. Instead of needing labeled attacks, the model learns normal context by predicting masked events and judging whether the next event is plausible given what typically comes before and after. In practice, that bidirectional context catches subtle sequence violations that long rule pipelines or unidirectional models miss, and the learned embeddings help cluster similar incidents and speed pivots during investigations. Some of its limitations are that its strongest results were obtained using curated corpora (which do not present the messiness and drift present in real LAN logs). Also, transformers add compute/latency that can strain strict real-time use, and similar to the other works, it stops at detection rather than producing plain-language explanations or triggering identity-aware actions on its own [6].

Positioning vs. Industry Tools

Commercial Security Orchestration, Automation, and Response (SOAR) platforms execute playbooks and ticketing, while Network Detection and Response (NDR) tools provide high-fidelity traffic analytics. The system we propose complements these by adding a tight detect → explain → act loop: streaming anomaly detection paired with LLM-generated, evidence-linked narratives and proportionate, reversible controls. In deployments, our system can feed alerts and narratives into SOAR for case management and ingest data from NDR/Zeek; its novelty is the explanation-first, identity-aware policy bridge that turns detection into safe action with built-in guardrails.

3. Synthesis

From the works reviewed in the previous section, Kitsune proves that lightweight, online detection can learn "normal" on the wire without labels. Shone *et al.* show that deep representations beat hand-built features. DeepLog and LogBERT teach us to read logs as language so that security signals arrive as a sequence with context, not isolated spikes. Halvorsen *et al.* explain where generative models can fill blind spots when labeled attacks are rare. Zeek gives us the durable event fabric that makes network activity legible to humans and machines. Put together, they present the need to observe the network richly, learn what is typical, and raise timely, context-rich alerts which are the main ingredients for a safer LAN. However, they were limited by drawbacks such as: most results were obtained on curated corpora, not on a real noisy campus or enterprise LAN. Detectors often produce scores without a clear, human-readable (non-technical) narrative, and even when a narrative exists, it rarely translates into safe, identity-aware action—no just-in-time MFA, no micro-segmentation, no reversible quarantine with audit. Also, encryption keeps growing, making more difficult to examine traffic payloads, and concept drift means the definition of "normal" keeps moving. Our contribution and work is designed to address these gaps by linking detection to explanation and to proportionate, reversible enforcement in one continuous loop.

4. Proposed Design

The TRIDENT Architecture

TRIDENT employs a LAN-first security pipeline that (1) watches what's happening on the network, (2) learns what a drifting "normal" looks like, (3) spots suspicious changes quickly, (4) explains them in simple non-technical language tied to the people and devices involved, and (5) takes adequate action automatically (with safety checks) while opening a ticket for humans to review.

LAN Traffic (Packets, Flows, Logs)

This is the raw "lifeblood" of a network responsible for moving everything between devices and servers. TRIDENT quietly observes packets on the wire, summarize them into flows (who talked to whom, how much, how often), and collecting key logs (authentications, DNS lookups, file shares, Transportation Layer Security (TLS) handshakes). Even when payloads are encrypted, the surrounding de-

tails such as server names (SNI), certificate info, timing, size patterns, and typical peer sets can still tell a lot about behavior. The goal at this stage is simple: capture rich, reliable telemetry without changing how the network operates (Figure 1).

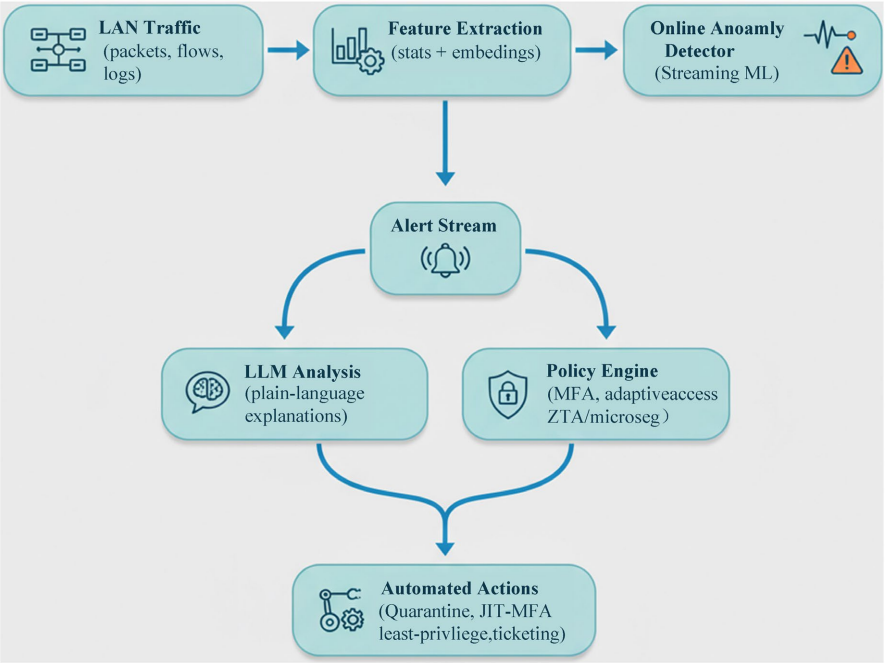

Figure 1. TRIDENT—Threat Recognition Intelligent Explanation & Enforced Deterrence.

Feature Extraction (Stats + Embeddings)

Raw telemetry is noisy, so TRIDENT reduces it into machine-readable signals to better capture rates (connections per minute), bursts and cooldowns, distinct peer counts, common vs. rare destinations, protocol traits (Server Message Block (SMB), Domain Name System (DNS), Transportation Layer Security (TLS)), and metadata fingerprints (JA3/JA3S). For logs, sequence tokens are extracted from template lines; for text-like fields, embeddings are generated that capture similarity and context. All of which makes good features like normal behavior easy to recognize and unusual behavior stand out quickly: this is the fuel the detector needs.

Online Anomaly Detector (Streaming ML)

Here the TRIDENT model learns what "normal" looks like for each user, device, and subnet as traffic streams by. It watches for shifts: a workstation suddenly talking to many new peers over SMB, an account pulling ten times its usual data, or a new domain that lights up after midnight. Because it runs online, it updates baselines as the environment changes (installation of new software or rollouts, semester start or resumptions from holidays, appearance of new devices) and scores events in near real time. The output isn't just a mystery number but it's a risk score along with associated short reasons and pointers to the evidence that triggered it.

Adversary note. Sophisticated actors can blur metadata (e.g., domain fronting,

Encrypted Client Hello, traffic padding/burst shaping), weakening signals like Server Name Indication and timing; to reduce this blind spot, our system correlates network metadata with endpoint and identity signals (process → connection, signed login context) and applies destination reputation and new-domain-age checks so evasion must succeed across multiple independent channels.

Alert Stream

All detector findings flow into a single, consistent feed. Each alert carries a risk level (High/Medium/Low), a compact explanation ("after-hours file transfers (SMB fan-out) to 20 new peers"), and links to the flows/logs involved. The feed is used to simultaneously drive an explanation layer that turns the signal into plain English for humans, and a policy engine that decides whether an automatic control should fire.

LLM Analysis (Plain-Language Explanations)

TRIDENT's LLM analysis engine acts as an AI assistant that reads the alert plus relevant context (identity, device role, asset sensitivity) and provides a short narrative a human can act on: who did what, when, where, how it differs from this entity's normal, and why it matters. It also proposes ranked next steps (e.g., "trigger MFA for user; move host to restricted Virtual LAN (VLAN) if MFA fails"). This step reduces guesswork by humans and ensures that any automatic action is accompanied by a clear and audit-able reason.

TRIDENT can use either (a) a general-purpose Large Language Model (LLM) via Application Programming Interface (API), which offers strong reasoning and broad language understanding, or (b) a smaller, domain-specific model fine-tuned on cybersecurity incident reports and LAN logs. The general model provides richer explanations out of the box but adds latency, cost, and data-handling controls; the fine-tuned model is faster, cheaper, and deployable on-premises, but requires curation of training data and periodic refresh to avoid topic drift. In practice, we combine them: the small model handles high-volume routine alerts; the general model is reserved for high-risk cases that justify deeper analysis.

Policy Engine (MFA, Adaptive Access, Zero Trust Architecture—ZTA/Micro-Segmentation)

TRIDENT's policy engine is responsible for transforming risk into action. Based on policy and confidence, the engine can challenge the user to perform additional MFA; temporarily tighten privileges (least-privilege); apply micro-segmentation or quarantine via Network Access Control (NAC); rate-limit a destination on the firewall or simply monitor and log if risk is low. Actions taken by the policy engine are proportionate, time-boxed, and reversible, with approvals where required. The engine integrates with Identity Providers (IdP) (e.g., Okta/Azure AD), firewalls, Endpoint Detection and Response (EDR) tools, and Trouble Ticketing Systems (TTS) so its actions and responses are consistent and traceable in line with Zero Trust Architecture (ZTA) principles.

Automated Actions (Quarantine, JIT-MFA, Least-Privilege, Ticketing)

When action is warranted, the system enforces it immediately and safely: isolate

a risky host, prompt a user for MFA, reduce a service account's access, or block a suspicious destination while opening/updates a ticket that includes the LLM's explanation and the evidence. Every action has rollback and an audit trail. Human (analyst) feedback (true/false positive, action outcome) loops back to improve thresholds, narratives, and policies, so over time the system gets faster, clearer, and more accurate.

Fail-Safes

To prevent disruptions from a faulty detector or mis-prompted LLM, TRIDENT enforces guardrails: (i) two-key consent for disruptive actions (e.g., quarantine of critical servers); (ii) time-boxed controls with auto-rollback if no confirmation arrives; (iii) a safe-mode that routes new detection patterns to monitor-only until a human approves enforcement; and (iv) blast-radius caps that limit how many assets a single rule can affect without explicit approval. Every action is logged with reason, evidence, and approver for audit.

Drift Handling

TRIDENT separates legitimate change from low-and-slow attacks by (i) using seasonal baselines (day-of-week/hour-of-day patters) and change point tests to recognize expected shifts; (ii) ingesting change signals (Configuration Management Database (CMDB) updates, software-release calendars, maintenance windows) to pre-label benign periods; (iii) gating new patterns through a canary phase in monitor-only mode; and (iv) watching for velocity/rarity cues (e.g., a gradual peer expansion confined to sensitive subnets) that distinguish stealthy campaigns from routine growth. Analyst feedback closes the loop to promote benign patterns into the baseline.

Key Building Blocks

1) Streaming anomaly detector: A live model that continuously learns the LAN's normal behavior and raises risk when patterns shift., e.g., sudden lateral scans, odd data egress, unusual admin actions, or after-hours access.

2) Alert stream (risk outputs): Every detection carries a risk score, short reason code, and the evidence the model used.

3) LLM analysis: Reads the alert and the relevant snippets (network, identity and device context) and produces a plain English narrative: who did what, when, on which asset, and why it matters plus a ranked list of next steps.

4) Policy engine: Applies proportionate, reversible controls based on risk and policy e.g., just-in-time MFA for the user, temporary least privilege for a service account, micro segmentation/quarantine for a device, or simple monitoring if confidence is low.

5) Automated actions and ticketing: Executes the action, opens/updates a ticket with the LLM's narrative and evidence, and notifies the right team.

Examples of How TRIDENT Works

As an illustration, consider a scenario where TRIDENT notices Host-23 suddenly opening twenty new Server Message Block (SMB) sessions in two minutes; the Large Language Model (LLM) provides human readable explanation of "After

hours, User A on Host-23 initiated new file transfers (fanned out) to over SMB to 20 other computers on the network (peers), this is an early lateral-movement pattern", and the policy engine immediately challenges User A to perform additional Multi-Factor Authentication (MFA) and places Host-23 on a restricted Virtual LAN (VLAN) to avoid further lateral-movement/attacks while a ticket is created with the full evidence. As a second example, consider a scenario where a service account pulls ten times its usual data from a rarely used server, the LLM flags "rare data access by svc-backup resulted in data download volume ten times above normal", and the policy engine triggers a time-box least-privilege action for two hours, audits recent tokens, and pings the owner to confirm.

When encrypted outbound traffic spikes to a brand-new domain, payloads remain unreadable, but metadata and timing look wrong, so the TRIDENT policy engine rate-limits the destination and prompts for MFA blocking and escalating if risk stays high. In order to address networks change due to new applications, new semesters or resumptions from holidays, appearance of new devices, the TRIDENT detector maintains rolling baselines with grace periods and drift checks so normal shifts don't create a flood of false positives; similarly unfamiliar behavior can result in running in monitor-only mode before any enforcement, and human/analyst feedback (marking true or false positives) trains the system to tighten thresholds over time.

Every automatic step is proportionate and reversible: we log the evidence and the LLM's explanation, support one-click rollback from the TRIDENT interface, and require human approval for disruptive actions like long quarantines. The prompts and alerts are carefully designed to minimize leaking personal data, redact secrets, and store only what's needed for security and audit; all actions, reasons, and approvals are traceable in the ticket, and policies are plain rules (for example, "High risk + sensitive server ⇒ JIT-MFA and micro-segmentation; Medium risk on student VLAN ⇒ monitor and notify"). Overall, success is measured by fewer false positives, faster time-to-contain (Mean Time to Recovery/Repair—MTTR), fewer repeat incidents, and clearer narratives that cut human analyst time per alert. In summary, the system learns what's normal, spots trouble early, explains it clearly, and acts safely with automatic actions that fit the risk and a real Local Area Network (LAN) where traffic is often encrypted, and behavior keeps changing.

5. Limitations and Future Work

In this paper, we present TRIDENT, a LAN-first security pipeline that (i) learns normal behavior in real time and detects anomalies, (ii) uses an LLM to convert raw alerts into plain-language, identity-aware explanations with recommended actions, and (iii) executes proportionate, reversible controls (e.g., JIT-MFA, micro-segmentation, quarantine) with audit and rollback closing the loop from detect → explain → act.

However, TRIDENT still has some real-world limits. As people and apps

change, what's "normal" on the network also changes so the TRIDENT detector may raise false alarms without adequate tuning and retraining that includes human/analyst feedback. Because much of today's traffic is encrypted, TRIDENT, mostly rely on clues like server names, information from certificates, and timing, which means attacks hidden within the encrypted payload may slip by unless endpoint sensors are used. The TRIDENT system also depends on other external tools such as identity management, network access control, endpoint security, ticketing. If those external tools are slow or poorly configured, TRIDENT's automated actions may not work well. Also, TRIDENT's LLM based AI assistant that generates human readable explanations can sometimes be wrong or be tricked by bad input, this may be addressed by using redaction, guardrails, and additional checks. During busy periods, the AI assistant and automation steps can introduce additional latency and delay, this could be mitigated using batching and lighter models to remain fast. Even with guardrails, automated enforcement can misfire; our caps, approvals, and rollback are designed to keep any mistake narrow and reversible. Attackers could also deliberately obfuscate metadata (e.g., domain fronting, ECH, padding), which reduces what TRIDENT can infer from the wire alone. And finally, everything has to be compliant with standards, company policy and privacy rules, with full audit logs and easy rollback to avoid overreach.

Future work on TRIDENT would focus on improvements by introducing additional components such as a drift guardian that automatically notices when normal behavior changes and safely retunes the detector without flooding the human operators with new alerts. Another idea is to combine TRIDENT's network view with light data from computers and logins (endpoints and identity agents) to aid detections even when traffic is encrypted. The AI's explanation's reliability could be improved by linking every claim to real evidence (the exact logs/flows) and by adding a quick self-check before sends alerts. It is possible to include a safe "test mode" to try actions like MFA, segmentation, or quarantine before turning them on for production use and making every action easy to roll back. It is also possible to develop and ship ready-made response playbooks for common LAN threats so teams can act fast and consistently. We will also look to quantify drift detection with retrospective tests (pre/post major rollouts) and measure false-positive/negative rates for slow-ramp attacks, refining thresholds and canary durations. Lastly, additional improvements to performance tuning and cost using queues, priorities, and smaller/faster models so the system stays quick, affordable, and compliant as the network grows.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Mirsky, Y., Doitshman, T., Elovici, Y. and Shabtai, A. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *Proceedings* 2018 *Network and Distributed System Security Symposium*, San Diego, 18-21 February 2018, 1-15.

https://doi.org/10.14722/ndss.2018.23204

[2] Shone, N., Ngoc, T.N., Phai, V.D. and Shi, Q. (2018) A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, **2**, 41-50. https://doi.org/10.1109/tetci.2017.2772792

[3] Du, M., Li, F., Zheng, G. and Srikumar, V. (2017). DeepLog. *Proceedings of the* 2017 *ACM SIGSAC Conference on Computer and Communications Security*, 1285-1298. https://doi.org/10.1145/3133956.3134015

[4] Halvorsen, J., Izurieta, C., Cai, H. and Gebremedhin, A. (2024) Applying Generative Machine Learning to Intrusion Detection: A Systematic Mapping Study and Review. *ACM Computing Surveys*, **56**, 1-33. https://doi.org/10.1145/3659575

[5] Paxson, V. (1999) Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, **31**, 2435-2463. https://doi.org/10.1016/s1389-1286(99)00112-7

[6] Guo, H., Yuan, S. and Wu, X. (2021). Logbert: Log Anomaly Detection via Bert. 2021 *International Joint Conference on Neural Networks* (*IJCNN*), Shenzhen, 18-22 July 2021, 1-8. https://doi.org/10.1109/ijcnn52387.2021.9534113