Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Adaptive Three-Layer Cloud Intrusion Detection Framework with Memory-Augmented LLM-Assisted Decision Making

Syed Waqas Ali
University of Engineering and Technology
Ibrar Ali Shah
University of Engineering and Technology
Farzana Zahid
farzana.zahid@waikato.ac.nz

University of Waikato
Daniyal Munir
Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
Schotten Hans D.
Intelligent Networks, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, 67663, Germany
Md Akbar Hossain
Eastern Institute of Technology

Research Article

Keywords: Cloud computing, Intrusion detection system, Reinforcement learning, Large language models

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10872518/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

An Adaptive Three-Layer Cloud Intrusion Detection Framework with Memory-Augmented LLM-Assisted Decision Making

Syed Waqas Ali[1], Ibrar Ali Shah[1*], Farzana Zahid[2*], Daniyal Munir[3], Hans D. Schotten[3,4], Md Akbar Hossain[5]

[1]Department of Computer Software Engineering, University of Engineering and Technology, Mardan, 23200, Pakistan.
[2]Department of Computer Science, University of Waikato, Hamilton, 3240, Waikato, New Zealand.
[3]WICON, RPTU University, Kaiserslautern, 67633, Germany.
[4]Intelligent Networks, German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, 67663, Germany.
[5]Eastern Institute of Technology, Auckland, 1010, New Zealand.

*Corresponding author(s). E-mail(s): ibrar@uetmardan.edu.pk; farzana.zahid@waikato.ac.nz;
Contributing authors: waqas.ali@uajk.edu.pk; daniyal.munir@rptu.de; schotten@eit.uni-kl.de; ahossain@eit.ac.nz;

Abstract

Cloud computing security is a major concern of organizations as their sensitive data is being managed on an off-premises platform. Several factors, such as the cloud's layered architecture, its dynamic environment, and dealing with unseen or zero-day attacks, lead to a complex and challenging security environment. Existing Intrusion detection system (IDS) solutions often focus on a single cloud layer and rely solely on machine learning models, which may perform well on benchmark datasets but remain limited when evaluated across heterogeneous cloud monitoring layers. This study proposes an adaptive confidence-aware multi-layer intrusion detection framework using reinforcement learning for network, host, and hypervisor environments. An XGBoost model is employed at each layer to detect known attacks, while prediction confidence thresholds calibrated via the Q-learning algorithm (layer-wise) are employed to separate reliable decisions from uncertain low-confidence events. Low-confidence cases are then passed to

the Chroma memory-matching gate. The unresolved events are further escalated to Large language model (LLM) for semantic analysis and explanation. Final attack promotion is performed using calibrated LLM confidence thresholds, while a weighted-fusion fallback mechanism is available for low-confidence ATTACK predictions. Events that remain unresolved after this process are retained in a review bucket to avoid forced classifications. Confirmed attack knowledge and generated explanations are stored in ChromaDB to support future analysis and incremental retraining. Experimental results show that the proposed framework autonomously learns layer-specific confidence thresholds of 0.79, 0.78, and 0.73 for the Network intrusion detection system (NIDS), Host intrusion detection system (HIDS), and Hypervisor layers, respectively. Across the complete 15,000-event evaluation set, the fixed 0.85 static baseline produces 3,111 would be LLM escalations. In the adaptive cold run, this workload is reduced to 2,714 actual LLM calls. After confirmed attack knowledge is stored in ChromaDB, the warm run resolves 2,068 uncertain events through memory matching and reduces actual LLM calls to 646. This corresponds to a 76.20% cold-to-warm LLM-call reduction and a 79.24% reduction compared with the static baseline workload. Under the complete 15,000-event warm-run evaluation, the framework achieves 94.63% accuracy, 94.10% precision, 93.79% recall, and 93.95% F1-score.

Keywords: Cloud computing . Intrusion detection system . Reinforcement learning . Large language models

1 Introduction

Cloud computing has progressively replaced traditional on-premises infrastructures by providing scalable, on-demand computing resources, rapid deployment capabilities, and cost-effective infrastructure management over the internet [1, 2]. This widespread adoption enables organisations to depend on cloud environments for core business operations and the storage of sensitive data, thereby making these platforms essential for ensuring operational continuity and maintaining service availability [3–5]. Reports from Gartner indicate that cloud computing is expected to become a business necessity by 2028, and market forecasts further suggest that the cloud market will double in size within the next eight years [6].

Cloud computing offers significant benefits to organisations and customers; however, its shared, multi-tenant, and multi-layered architecture expands the attack surfaces, making it vulnerable to various cyber threats [7, 8]. Each architectural layer presents distinct security concerns, such as network-level attacks involving abnormal traffic patterns, host-level attacks arising from system misuse and suspicious activities, and sophisticated attacks, like Virtual machine (VM) escape, that target the virtualisation layer by exploiting hypervisor vulnerabilities, which can affect multiple VMs simultaneously [9–11]. To address these security concerns, cloud-based Intrusion detection system (IDS) are widely used to monitor network traffic (NIDS), system activities (HIDS), and virtualisation environments for each architectural layer [9].

Generally, IDS operating at a single layer often fails to detect attacks that span across other layers, highlighting the need for proactive multi-layer intrusion detection mechanisms to improve cloud security [11]. To improve the performance of cloud-based NIDS and HIDS, Machine learning (ML) techniques have been widely reported [12–15]. However, these IDS still face challenges as evolving zero-day attacks and unrealistic evaluation strategies lead to overly optimistic results and poor generalisation [16, 17].

Recently, Large language model (LLM)s have emerged as powerful tools for attack analysis and detection in cloud environments, enabling intelligent automation across threat detection, vulnerability assessment, and incident response [18, 19]. The ability of LLMs to understand the context and interpret complex patterns makes them particularly well-suited for cloud-based IDS, where cloud-specific threats such as cross-tenant lateral movement, hypervisor intrusions, and API misuse demand more adaptive detection than rigid rule-based systems can offer; LLMs address this gap through semantic log analysis across dynamic, auto-scaling cloud infrastructures [20, 21]. In practice, IDS are now being integrated into security operations centre (SOC) workflows through platforms such as Microsoft Security Copilot, where they assist with phishing analysis, threat intelligence summarization, and real-time incident response support [22]. More recently, LLM-based agentic systems have further extended these capabilities by autonomously coordinating multi-step security operations across the entire incident lifecycle, from risk assessment and detection to response and post-incident recovery [23].

Our survey (Section 2) shows that despite growing interest in ML and LLM-assisted intrusion detection for cloud environments, most existing studies focus on a single detection layer and do not consider how network, host, and hypervisor layers collectively secure and operate within a unified detection system [19, 24]. In addition, current IDS pipelines rely on static confidence thresholds that do not adapt to the different behavior of these layers. The use of static confidence threshold can result in two key issues. First, it may cause unnecessary escalation, where events that are uncertain at a specific layer-level classifier (i.e., with confidence below the learned threshold) are forwarded to the next stage instead of being resolved at the current layer. This often results in additional analysis by LLMs, increasing computational cost. Second, it can lead to insufficient escalation, where events that should be further analyzed are not forwarded, potentially resulting in missed attacks [18].

Although LLMs provide strong contextual reasoning for analysing uncertain and zero-day attacks, their uncontrolled use can reduce detection efficiency and increase operational overhead [20]. Moreover, direct reliance on LLM outputs without confidence control may introduce unreliable automatic decisions, particularly for ambiguous alerts such as low-confidence attack predictions, inconclusive hypervisor events, or host-level logs with overlapping benign and malicious behaviour. Therefore, LLM-assisted intrusion detection requires selective escalation, calibrated attack confirmation, fallback handling for borderline ATTACK predictions, and review-based abstention for unresolved events. In this study, calibrated attack confirmation means accepting an LLM-predicted ATTACK only when its confidence satisfies the learned layer-specific threshold, fallback handling refers to applying weighted fusion when the LLM predicts ATTACK but remains below this threshold, and review-based abstention

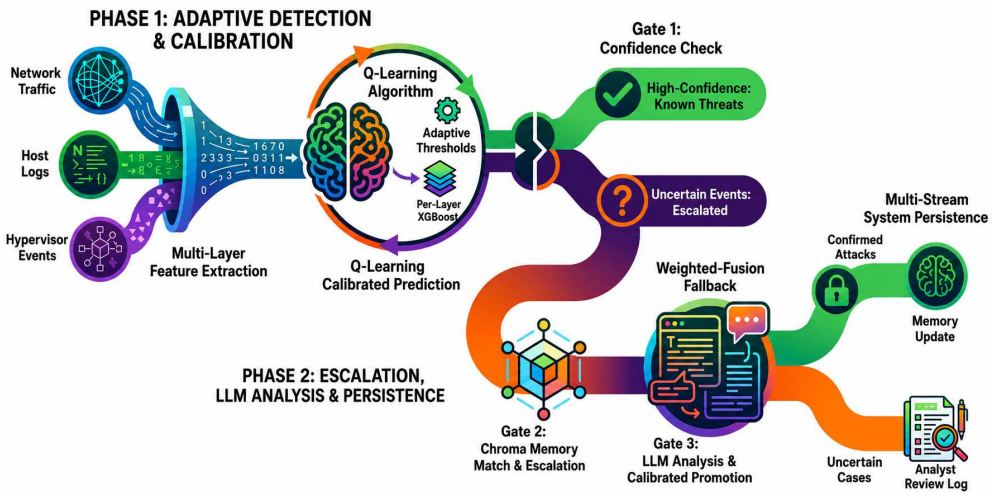

Fig. 1 High-level Architecture of the Proposed Multi-Layer Cloud IDS with adaptive thresholding, Chroma memory, and LLM-assisted decision making

means retaining unresolved cases for analyst review rather than forcing an automatic label. Furthermore, existing intrusion detection systems lack mechanisms for deciding whether alerts should be handled internally or escalated to LLM analysis based on model confidence, and they do not integrate historical attack memory to resolve uncertain low-confidence attack events [23]. Existing LLM-assisted IDS studies also rarely evaluate whether stored attack knowledge can be reused in subsequent executions to reduce repeated LLM invocations for semantically similar uncertain events. To address the limitations of static decision systems, recent research has explored Reinforcement learning (RL) as an adaptive decision-making approach. In the RL approach, the system learns from its experiences and continuously adjusts its decisions in response to dynamic environments [25]. In particular, deep Q-learning (a type of RL) provides continuous learning through trial-and-error interactions, enabling improved intrusion detection performance over time [26].

Recent studies on reinforcement learning-assisted intrusion detection have shown that RL-based models with online uncertainty estimation outperform static threshold models in heterogeneous environments, supporting the use of adaptive threshold learning for multi-layer IDSs [27]. *Therefore, there remains a clear gap in developing a comprehensive cloud IDS framework that integrates multi-layer detection, adaptive (i.e., dynamically learned confidence threshold) based routing, memory-assisted decision support, and LLM-based contextual analysis within a single system.*

To address the above-mentioned issues, we proposed a multi-layer cloud IDS framework (Fig. 1). The architecture of the proposed IDS comprises two phases: *1) Adaptive detection and calibration, and 2) Escalation LLM analysis and persistence.* The first phase performs adaptive detection using an Extreme gradient boosting (XGBoost) model [28] with Q-learning-based threshold calibration, while the second phase handles

low-confidence events through memory matching, LLM-based analysis, and persistent logging with feedback-driven retraining. In this phase, confirmed attack decisions are stored in ChromaDB and subsequently reused during warm execution to reduce repeated LLM calls for semantically similar uncertain events. The framework is evaluated under two pipeline implementations: a static threshold-based IDS pipeline, which serves as a baseline, and the proposed adaptive IDS pipeline, both described in detail in Section 3. The adaptive pipeline is further evaluated under cold and warm settings to quantify the effect of persistent attack-memory reuse on LLM workload reduction. *The main contributions of this paper* are summarised as follows:

1. A unified multi-layer cloud-based intrusion detection framework that integrates network, host, and hypervisor-level monitoring within a single detection architecture.
2. An RL-based adaptive threshold mechanism to automatically learn layer-specific confidence thresholds for intrusion detection decisions.
3. An uncertainty-aware intrusion detection strategy that integrates ML detection with LLM-assisted analysis for low-confidence events.
4. A ChromaDB-based persistent attack-memory mechanism that stores confirmed attack knowledge during cold execution and reuses it during warm execution to reduce repeated LLM invocations.
5. A comparative evaluation with existing intrusion detection approaches and between static and adaptive decision mechanisms (including cold as well as warm executions) to demonstrate the effectiveness of the proposed framework.

The remainder of this paper is organized as follows. Section 2 presents the literature review on cloud IDS and related ML-based approaches. Section 3 describes the proposed multi-layer intrusion detection framework and adaptive decision mechanism. Section 4 presents the implementation details and discussion of the proposed system. Section 5 provides the results evaluation and performance analysis. Finally, Section 6 concludes the paper and outlines future research directions.

2 Literature Review

In the existing literature, several studies have explored IDS using Deep learning (DL) and classical ML approaches. The study [29] evaluated a transformer-based NIDS using the Canadian Institute for Cybersecurity Intrusion Detection System 2018 (CICIDS2018) dataset for traffic-based attacks such as Denial of service (DoS), distributed DoS (DDoS), botnet, and infiltration. Their model achieved an accuracy of up to 99% for DoS and botnet attacks, but showed poor performance for infiltration and brute force attack patterns in cloud network environments. Similarly, in [30], researchers proposed a DL-based HIDS for containerised cloud environments using system call monitoring at the kernel level, and the Leipzig intrusion detection dataset (LID-DS-2019).

It has also been observed that XGBoost is widely used for the development of IDS due to its high detection performance and scalability for cloud environments. A real-time XGBoost-based IDS with hyperparameter optimisation has been proposed in [31] with improved detection performance and low computational overhead. Similarly, [32]

proposed an XGBoost-based IDS classification model that achieved 98.7% accuracy and outperformed several traditional ML models. These studies highlight XGBoost as an efficient and scalable approach for cloud-based IDS. Gavric et al. proposed a lightweight, time-driven HIDS for detecting DoS attacks on resource-constrained IoT devices using only ten system and network utilization features [33]. The framework achieved over 99.9% accuracy with Random Forest while maintaining constant $O(1)$ complexity. However, it requires device-specific training and was evaluated on a relatively small Raspberry Pi-based IoT testbed. Iturbe-Araya and Rifà-Pous examined the effect of different optimization metrics on unsupervised anomaly detection across four imbalanced smart-home datasets [34]. Their results showed that MCC-guided hyperparameter optimization produced more stable and balanced models than optimization based on accuracy or AUC-ROC. However, the study is limited to static IoT datasets and does not address adaptive decision-making in multi-layer cloud environments.

Moreover, hybrid learning models have been explored for cloud-based IDS using a convolutional neural network-recurrent neural network (CNN-RNN) model evaluated on the network security laboratory-knowledge discovery in databases (NSL-KDD) dataset [35], demonstrating effective spatial and temporal traffic pattern learning. The study [36] proposed the optimized bidirectional convolutional long short-term memory (OBCLSTM (Bi-CNN-LSTM)) model evaluated on multiple datasets, including NSL-KDD99, Ton Duc Thang University intrusion detection system (TUIDS), University of New South Wales-network benchmark 2015 (UNSW-NB15), and botnet of things-internet of things (BoT-IoT), achieving a 99.1% detection rate and highlighting model generalizability for cloud environments. Additionally, a deep autoencoder combined with CNN was used for low-rate DDoS detection [37], achieving 95.32% accuracy by leveraging unsupervised feature learning for detecting stealthy cloud-based attacks. Collectively, these studies show that hybrid learning models perform well for frequent and well-represented attack types, but their performance decreases for rare or new attacks, highlighting the need for adaptive and multi-layer intrusion detection frameworks. Agate et al. proposed MIDES, a multi-stage network intrusion detection system that combines a binary Gatekeeper, specialized multi-class classifiers, and a self-adaptive agent that selects between weighted voting and meta-classification using classifier expertise and prediction confidence [38]. Evaluations on CIC-IDS2017 and CSE-CIC-IDS2018 showed strong multi-class detection performance and reduced classification time. However, the framework is limited to network-level intrusion detection and does not address cloud-specific host or hypervisor monitoring.

Recent studies have started integrating LLMs into IDS to handle uncertain or ambiguous cases that traditional models cannot effectively classify. For example, a two-tier LLM-based IDS is proposed by [39] for software-defined networking (SDN) and Kubernetes environments that combine statistical anomaly detection methods such as exponential moving average (EMA) and auto-regressive integrated moving average (ARIMA) with on-demand packet capture. The system was evaluated using real attack traffic, including user datagram protocol (UDP) floods, port scans, synchronize (SYN) floods, secure shell (SSH) brute-force, slowloris, and HTTP/2 attacks. Results showed faster detection in SDN (1-1.5s) compared to Kubernetes (6–30s), but

the approach introduced response delays due to application programming interface (API) communication and LLM inference time. Whereas, in [40], an IDS based on transformer language models was introduced for detecting evolving and unknown network attacks through continuous learning. Also, researchers in [41] developed a hybrid intrusion detection framework combining network and host-based features using a two-stage collaborative classifier to improve detection accuracy and reduce computational complexity.

Similarly, a federated learning-based IDS framework was developed in [42] where low-confidence alerts from the base detector are forwarded to an LLM for secondary analysis using external threat intelligence and a federated Retrieval augmented generation (RAG)-based incident knowledge base. The system also feeds LLM-verified alerts back into the training process to improve detection. the results showed improved recall for underrepresented attacks and reduced false positives, but the approach still relies on fixed confidence thresholds and lacks a calibrated decision mechanism for borderline LLM cases. Furthermore, CyberRAG, an agentic RAG-based system combining a central LLM with three fine-tuned transformer classifiers was proposed in [43] for structured query language (SQL) injection, cross-site scripting (XSS), and server-side template injection (SSTI) attacks, achieving 94.92% accuracy with RAG compared to 84.75% without RAG. The basic limitation of the study is dependency on predefined classifiers and static knowledge bases, restricting generalisation to novel attack types.

Intrusion detection at the hypervisor layer has also gained increasing attention. Several studies have explored ML, federated learning, bio-inspired optimisation, and RL approaches for detecting hypervisor-level attacks such as hyperjacking, side-channel attacks, and VM escape attacks [44–47]. These studies demonstrate that hypervisor-level monitoring can significantly enhance cloud security by detecting virtualisation-specific threats. However, most existing approaches focus on isolated hypervisor detection mechanisms and do not integrate hypervisor monitoring within a unified multi-layer IDS framework, highlighting the need for integrated multi-layer cloud intrusion detection architectures. Since attacks and operational conditions evolve over time, intrusion detection mechanisms must also adapt. In this regard, RL has emerged as a promising approach for adaptive IDS to overcome the static nature of traditional models.

Kheddar et al. reviewed RL- and DRL-based intrusion detection across network, host, cloud, edge, IoT, smart-grid, and industrial environments [48]. Their analysis shows that RL can improve IDS adaptability, automate response decisions, and reduce false alarms. However, practical deployment remains limited by interpretability, scalability, real-time overhead, adversarial robustness, and continued reliance on outdated or simulated datasets. Hammar and Stadler formulated intrusion prevention as a partially observable multiple-stopping problem and proposed T-SPSA, an RL algorithm for learning threshold-based defensive policies [49]. The method determines when defensive actions should be initiated using IDS alerts and login activity and was trained through simulation before being validated in a Docker-based emulation environment. The learned policies outperformed PPO, HSVI, static IDS-response rules, and a single-threshold change-detection approach. However, the study focuses

on intrusion-response timing in a general IT infrastructure rather than confidence calibration across multiple cloud detection layers.

Huang and Zhu proposed RADAMS, a Q-learning-based alert management strategy for reducing analyst overload during Informational Denial-of-Service attacks [50]. The method models operator attention, stress, expertise, and alert criticality to selectively de-emphasize feint alerts, reducing the estimated attack risk by up to 20%. Although evaluated in an industrial control system setting, the study demonstrates the value of RL-based adaptive alert routing and human-in-the-loop escalation.

The study in [51] developed a deep Q-network (DQN)-based security mechanism for achieving high accuracy and low computational overhead, while [52] used RL to optimise classifier ensemble decisions, improving detection performance. In general, these studies show that RL enables adaptive decision-making and dynamic optimisation in IDS.

The reviewed literature highlighted several limitations, including potentially costly LLM escalation, reliance on curated knowledge bases, insufficient hypervisor-level threat coverage, and static threshold configurations, collectively exposing a clear research gap. *Overall*, our work addresses these limitations through a comprehensive IDS pipeline for cloud environments. The pipeline is built using an XGBoost classifier with adaptive RL-based thresholding and LLM-assisted decision analysis. Unlike prior fixed-threshold or single-stage escalation approaches, the proposed framework applies F0.5-calibrated LLM attack confirmation, where the F0.5 score gives greater importance to precision than recall during attack confirmation. It also applies weighted-fusion fallback for borderline ATTACK predictions that do not satisfy the direct LLM confidence threshold, and review-based abstention for unresolved cases. It further incorporates a ChromaDB-based persistent attack-memory component that stores confirmed attacks during cold execution and reuses them during warm execution to reduce repeated LLM invocations for semantically similar uncertain events.

3 Proposed Multi-Layer Adaptive Cloud IDS Framework

This section presents the proposed multi-layer cloud intrusion detection framework for security monitoring across network, host, and hypervisor layers in cloud environments. The proposed adaptive IDS pipeline is divided into two primary phases, which are discussed in detail in the following subsections.

3.1 Phase 1: Adaptive Detection & Calibration

In Phase 1, we propose a Q-learning-based cloud intrusion detection pipeline for layer-specific confidence-threshold selection. As shown in Fig. 2, the framework integrates XGBoost with an LLM and memory pipeline to support high-confidence detection of previously observed attacks and confidence-aware analysis of low-confidence or ambiguous events within the three-layer IDS. Rather than forcing all uncertain events into automatic labels, the framework routes low-confidence cases to memory matching and calibrated LLM-assisted analysis, while unresolved cases are retained in the

review/abstention bucket for analyst review. XGBoost is selected as the base classifier in this study due to its high classification accuracy, computational efficiency, and robustness on tabular network data [53, 54].

Phase 1 consists of two main processes: *1) multi-layer feature extraction* and *2) calibrated prediction*. In the first process, raw data is collected from each layer, including network traffic, host logs, and hypervisor events, which is transformed into structured numerical representations using layer-specific feature extraction techniques. For the network layer, flow-based statistical features from the CICIDS2018 dataset are utilized. For the host layer, textual logs are converted into numerical vectors using term frequency-inverse document frequency (TF-IDF) . For the hypervisor layer, event-based logs are parsed and encoded into structured representations.

In the second process, *calibrated prediction*, each layer performs XGBoost-based prediction, generating class labels together with confidence scores, where the confidence is defined as the predicted class probability. This is followed by Q-learning-based adaptive thresholding, which determines an effective confidence threshold for each layer by iterative learning, based on prediction outcomes. The Q-learning threshold model was empirically trained for twenty episodes, as this configuration consistently produced stable learned thresholds during experimentation (Section 4). Since the Q-learning calibration adjusts only confidence thresholds rather than learning complex attack-classification strategies, stable threshold decisions can be achieved with a relatively small number of training episodes [55]. For each layer, the Q-learning model selects a threshold from a predefined threshold set, uses it to route each event as KNOWN (confidence is equal to or greater than the learned threshold) or UNCERTAIN (confidence is less than the learned threshold), receives reward based on correctness/risk/escalation cost and uncertain-band control, and updates Q-values with the Bellman rule as shown in Eq. (1), originally introduced in the Q-learning framework [56].

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \right] \tag{1}$$

where:

- $Q_t(s_t, a_t)$ is the current Q-value for selecting action a_t in state s_t at iteration t,
- $Q_{t+1}(s_t, a_t)$ is the updated Q-value after observing the reward and next state,
- s_t is the current state (discretized confidence statistics) of the classifier,
- a_t is the selected threshold for event routing,
- r_t is the reward from the routing decision,
- α is the learning rate, and
- γ is the discount factor.

After the training episodes finish, it runs a greedy rollout to pick the most frequently selected threshold as the learned threshold for Gate 1, as shown in Fig. 2. It continuously monitors confidence scores, uncertainty indicators, and alert patterns produced by the NIDS, HIDS, and hypervisor layers. Based on the observed system behavior, the Q-learning mechanism dynamically adjusts confidence thresholds to balance detection reliability and alert overhead. In other words, it enables each IDS layer to adaptively

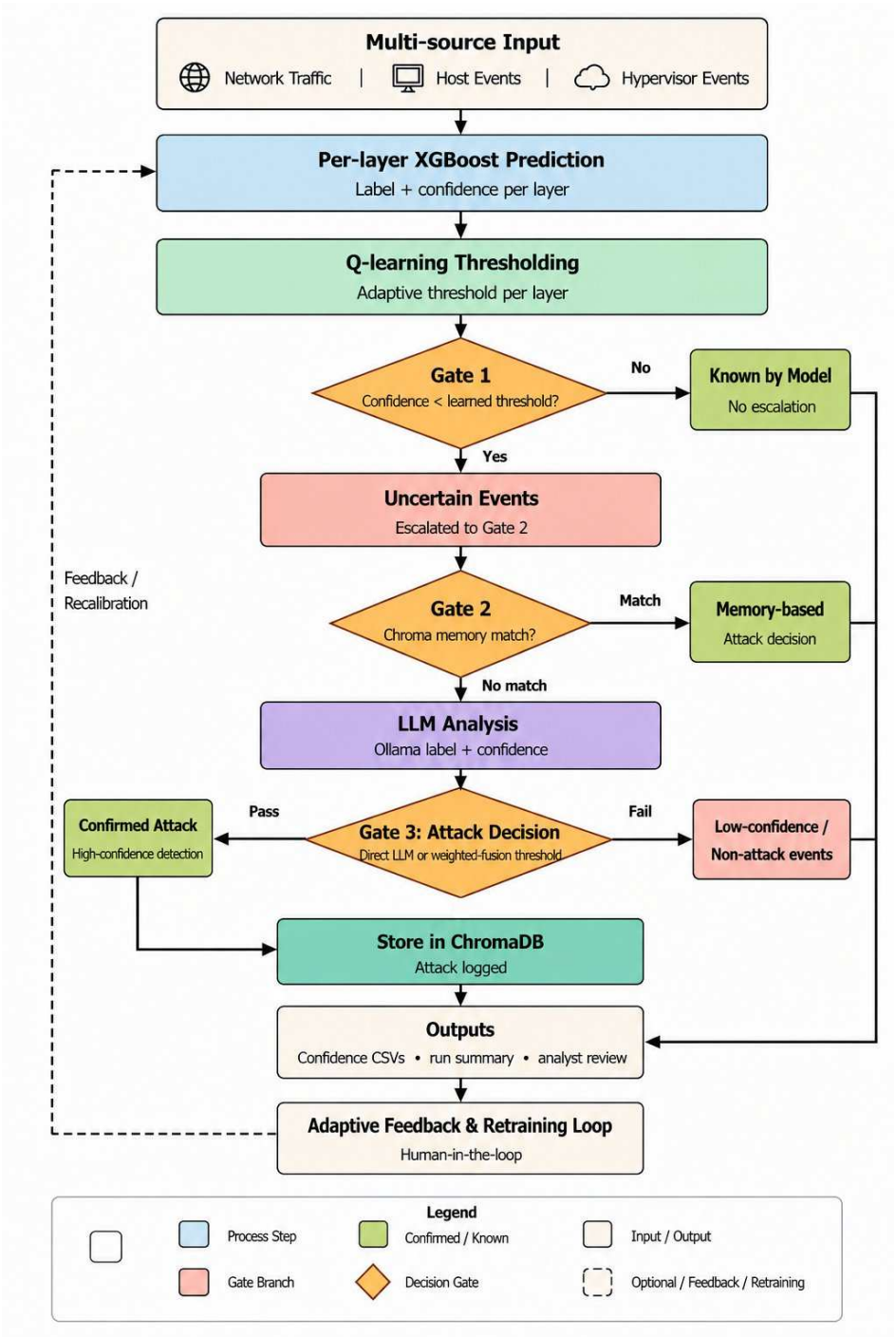

Fig. 2 Proposed Q-learning-calibrated IDS pipeline with three adaptive decision gates, Chroma vector memory, LLM analysis (Ollama), and a periodic retraining feedback loop.

decide for itself whether it is confident enough to classify the event and when to escalate to LLM for assistance. The output of this mechanism reduces unnecessary escalation to LLM while staying reliable under different conditions.

The Q-learning threshold model represents each layer using confidence-based state information, including the mean confidence, confidence variance, and the ratio of uncertain events computed over a sliding window of recent predictions. The available actions correspond to selecting confidence thresholds from a predefined range of candidate threshold values. A reward mechanism is used to guide threshold selection by encouraging correct high-confidence decisions and discouraging misclassifications or unnecessary LLM escalations. In particular, correct high-confidence predictions receive positive rewards, whereas misclassifications and unnecessary escalations are penalized. Through iterative interaction with the detection pipeline, the Q-learning-based threshold calibration mechanism gradually learns an effective threshold adjustment policy that improves routing performance without modifying the underlying detection models. This process reduces uncertainty cost, defined as the computational overhead and response delay caused by escalating ambiguous events for advanced LLM-assisted analysis, as shown in Fig. 2. During this phase, the Gate-1 confidence check determines whether an event should be directly accepted or escalated. If the confidence exceeds the learned threshold, the event is classified as a high-confidence prediction with no escalation; otherwise, the low-confidence event proceeds to the Gate-2 memory-matching stage in Phase 2.

3.2 Phase 2: Escalation, LLM Analysis & Persistence

Phase 2 includes three processes: *1) escalation*, *2) LLM analysis*, and *3) persistence*, as shown in Fig. 1. The escalation process starts with a memory-based matching gate, called Gate 2, where each escalated uncertain event from Phase 1 is compared against the ChromaDB vector memory. If a similarity match is found, defined as the semantic similarity between the embedding of the current event and the embeddings of previously confirmed ATTACK events stored in memory, the event is classified as ATTACK through a memory-based decision; otherwise, it is forwarded to Gate 3 for LLM-based final validation. The flow can be seen in Fig. 2.

The memory match is based on three factors: distance, support, and meta-confidence. The distance measures semantic similarity, support shows how many consistent neighboring attack records back the match, and meta-confidence summarizes the reliability of that memory evidence, where the current event embedding is compared against previously confirmed attack embeddings [57–59]. Initially, during the first runtime execution of the pipeline, the Chroma memory remains empty, as no previously confirmed attack embeddings are available. Therefore, during cold execution, uncertain events do not benefit from prior memory matches and are forwarded to the LLM analysis stage. During warm execution, confirmed attacks stored from earlier runs can be reused by Gate-2 to reduce repeated LLM calls. With the confirmed attacks stored over time, memory becomes progressively populated, enabling Gate-2 to directly classify events as attacks through memory-based decisions for high-confidence matches. The events with low similarity or no reliable match are forwarded to LLM analysis for contextual re-evaluation.

The next process in Phase 2 is LLM analysis using Ollama, followed by the Gate 3 final-decision check, as shown in Fig. 2. Gate 3 uses the LLM prediction label and confidence score to make the final classification. At Gate 3, classification follows two calibrated rules. First, a direct confirmation rule is used: an event is accepted as ATTACK only when the LLM predicts ATTACK and its confidence exceeds the layer-specific F0.5-calibrated LLM confidence threshold. The F0.5 measure is used because it gives higher importance to precision than recall, which is suitable for intrusion detection settings where excessive false attack confirmations can increase analyst workload and reduce operational trust [60]. The F0.5 score is defined as:

$$F_\beta(t) = (1+\beta^2)\frac{\text{Precision}(t)\cdot\text{Recall}(t)}{\beta^2\cdot\text{Precision}(t)+\text{Recall}(t)}, \tag{2}$$

where t denotes a candidate LLM confidence threshold and $\beta = 0.5$ is used to prioritize precision over recall during ATTACK confirmation. The layer-specific LLM attack-confirmation threshold is selected as:

$$\tau^{F0.5}_{\text{LLM},\ell} = \arg\max_t F_{0.5,\ell}(t). \tag{3}$$

In Eq. (3), ℓ denotes the IDS layer, and the selected threshold is the value of t that maximizes the layer-specific $F_{0.5}$ score.

During escalation, the LLM assigns a semantic decision label together with a confidence score. If the LLM predicts ATTACK and its confidence exceeds the calibrated F0.5 threshold, the event is accepted as a confirmed ATTACK. If the LLM predicts BENIGN, the event is treated as non-attack. Events that are neither confidently confirmed as ATTACK nor BENIGN are retained for review. The direct LLM decision rule can be expressed as:

$$\hat{y}_{\text{LLM}} = \begin{cases} 1, & \text{if } d_{\text{LLM}} = \text{ATTACK} \wedge c_{\text{LLM}} \geq \tau^{F0.5}_{\text{LLM},\ell}, \\ 0, & \text{if } d_{\text{LLM}} = \text{BENIGN}, \\ \text{REVIEW/UNSURE}, & \text{otherwise.} \end{cases} \tag{4}$$

where $\hat{y}_{\text{LLM}}$ denotes the final LLM-based decision, d_{LLM} denotes the semantic decision generated by the LLM, c_{LLM} represents the LLM confidence score, and $\tau^{F0.5}_{\text{LLM},\ell}$ is the F0.5-calibrated LLM attack-confirmation threshold for layer ℓ.

If the LLM predicts ATTACK but its confidence score falls below the layer-specific F0.5-calibrated threshold $\tau^{F0.5}_{\text{LLM},\ell}$, the case is processed using a weighted-fusion fallback mechanism, which combines the base-classifier confidence and the LLM confidence. If the resulting fusion score meets the layer-specific fusion threshold, the event is classified as a confirmed ATTACK; otherwise, it is routed to the review/abstention bucket, as shown in Fig. 2. Since fusion is applied only to cases where the LLM predicts ATTACK but its confidence is below the F0.5-calibrated threshold, the LLM is assigned a higher weight ($w_l = 0.80$), while the classifier is assigned a smaller weight

($w_m = 0.20$). The weighted-fusion score is computed as [61]:

$$S_{\text{fusion}} = w_m \, C_{\text{model}} + w_l \, C_{\text{LLM}}, \tag{5}$$

where S_{fusion} denotes the weighted-fusion score, C_{model} is the base-classifier confidence, C_{LLM} is the LLM confidence, and w_m and w_l are the weights assigned to the classifier and LLM, respectively, with $w_m + w_l = 1$. The fallback decision rule is:

$$\hat{y}_{\text{fusion}} = \begin{cases} 1, & \text{if } d_{\text{LLM}} = \text{ATTACK} \wedge S_{\text{fusion}} \geq \tau_{\text{fusion},\ell}, \\ \text{REVIEW}, & \text{otherwise.} \end{cases} \tag{6}$$

where $\tau_{\text{fusion},\ell}$ denotes the layer-specific fusion threshold, which is set equal to the corresponding layer's F0.5-calibrated LLM attack-confirmation threshold. The fusion weights used in this study, $w_m = 0.20$ and $w_l = 0.80$, were selected empirically based on the validation behavior of the uncertain-event subset. This choice is consistent with prior work showing that weighted score fusion is a valid approach for combining heterogeneous classifier outputs and confidence scores [62, 63].

The post-gate stages perform attack persistence by storing confirmed attack events (i.e., events classified as attacks at Gate-3) in the Chroma attack database (ChromaDB), uncertainty auditing by logging Gate-1 uncertain cases, and output finalization by generating confidence comma-separated values (CSVs) files, run summaries, and analyst review records [64, 65]. The review bucket is retained as structured JavaScript object notation lines (JSONL) files, so that unresolved events remain traceable and auditable rather than being discarded. These persistent outputs close the loop by updating the knowledge base for the future retraining stage, where accumulated validated events are used for periodic recalibration of thresholds/weights and for retraining an independent XGBoost-based classifier to maintain detection performance over time. To improve reproducibility and provide a concise overview of the operational workflow, Algorithm 1 summarizes the overall decision-making process of the proposed cloud IDS framework.

Algorithm 1 Workflow of the Proposed Multi-Gated Cloud IDS Framework

Require: Multi-source cloud event stream E from network, host, and hypervisor layers; layer-specific Gate-1 thresholds τ_ℓ learned via Q-learning training and greedy rollout; F0.5-calibrated LLM attack-confirmation thresholds $\tau^{F0.5}_{\text{LLM},\ell}$ and fusion thresholds $\tau_{\text{fusion},\ell}$

Ensure: Final decision per event: *Known by Model*, *Memory-based Attack Decision*, *Confirmed Attack*, *Non-attack*, or *Review / Abstention*

Adaptive Threshold Learning
for each layer $\ell \in \{\text{NIDS}, \text{HIDS}, \text{Hypervisor}\}$ **do**
    Train Q-learning model and extract τ_ℓ through greedy rollout
end for
Multi-Gated Event Processing
for each incoming event $e_i \in E$ **do**
    Identify source layer ℓ and extract layer-specific features
    Generate prediction $\hat{y}_m$ and confidence score C_m using the layer-specific XGBoost classifier
    if $C_m \geq \tau_\ell$ **then**
        Gate 1: Accept $\hat{y}_m$ as **Known by Model** with no escalation
    else
        Gate 1: Mark e_i as **Uncertain Event** and escalate to Gate 2
        if reliable Chroma memory match is found using distance, support, and meta-confidence **then**
            Gate 2: Classify e_i as **Memory-based Attack Decision**
            Record decision in output logs
        else
            Gate 2: Submit e_i to LLM (Ollama) analysis
            Obtain LLM label d_{LLM} and confidence c_{LLM}
            Compute weighted-fusion score $S_{\text{fusion}} = w_m C_m + w_l c_{\text{LLM}}$
            if $d_{\text{LLM}} = \text{ATTACK}$ and $c_{\text{LLM}} \geq \tau^{F0.5}_{\text{LLM},\ell}$ **then**
                Gate 3: Classify e_i as **Confirmed Attack** using the direct F0.5-calibrated LLM rule
                Store confirmed attack in ChromaDB ▷ Attack logged
                Record decision in output logs
            else if $d_{\text{LLM}} = \text{ATTACK}$ and $S_{\text{fusion}} \geq \tau_{\text{fusion},\ell}$ **then**
                Gate 3: Classify e_i as **Confirmed Attack** using weighted-fusion fallback
                Store confirmed attack in ChromaDB ▷ Attack logged
                Record decision in output logs
            else if $d_{\text{LLM}} = \text{BENIGN}$ **then**
                Gate 3: Classify e_i as **Non-attack**
                Record decision in output logs
            else
                Gate 3: Route e_i to **Review / Abstention** bucket
                Record decision in output logs
            end if
        end if
    end if
end for
Generate confidence CSVs, run summary, and analyst review files
Use confirmed attacks and review feedback for periodic threshold recalibration and XGBoost retraining

To provide a comparative baseline for evaluating the proposed adaptive framework, the study considers two pipeline implementations: the proposed adaptive IDS pipeline based on RL-driven threshold optimisation and uncertainty-aware decision routing, illustrated in Fig. 2, and a static threshold-based three-layer IDS pipeline,

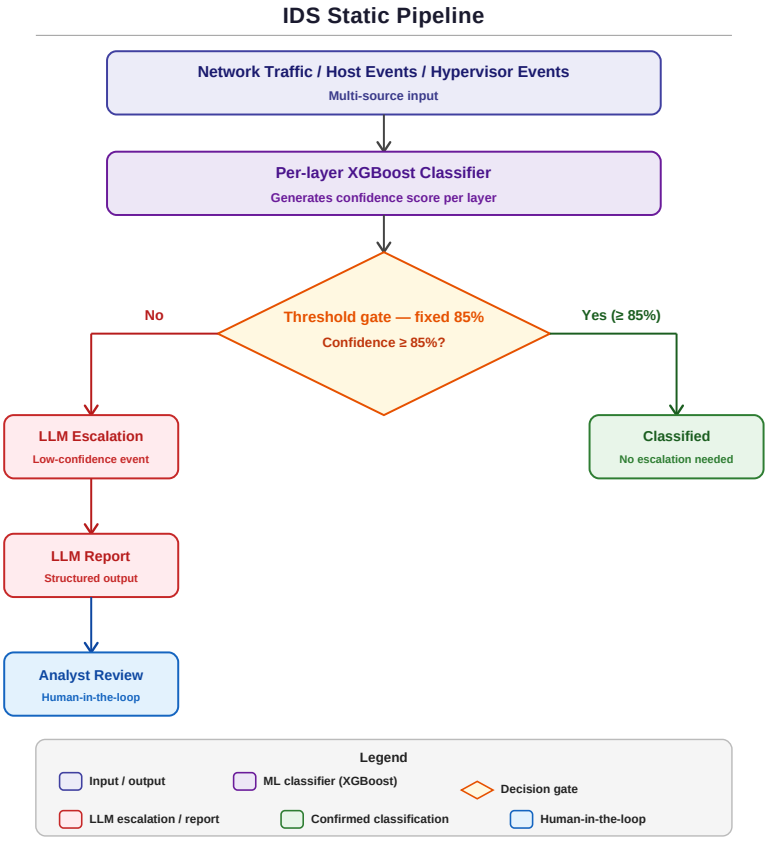

Fig. 3 Static thresholding baseline IDS pipeline using a fixed 85% XGBoost confidence threshold for intrusion detection.

shown in Fig. 3. The baseline pipeline represents a complete three-level security implementation comprising network, host, and hypervisor-based IDS, each deployed on a separate layer. Both pipelines perform intrusion detection as a binary classification task, distinguishing between benign and malicious activity across network, host, and hypervisor layers, enabling consistent and comparable evaluation under matched experimental conditions. The static threshold is fixed, as adopted from recent work [42], which applies a similar fixed-threshold approach. Fig. 3 illustrates the workflow of the baseline pipeline, where the XGBoost classifier processes inputs from each layer. The classifier assigns a confidence score (the predicted probability that a given event is malicious or benign) to each flow. If the score is equal to or exceeds the static threshold, the event is classified, without escalation to the LLM. On the contrary, if the confidence score falls below the static threshold, the event is escalated to the LLM for further analysis. However, a fixed threshold fails to adapt to the varying traffic patterns across layers, leading to unnecessary LLM escalations or missed detections. Overall, the proposed framework integrates supervised detection with LLM-based semantic reasoning,

providing adaptive and explainable intrusion analysis across cloud network, host, and hypervisor layers, with RL-driven threshold adaptation, F0.5-calibrated attack confirmation, review-based abstention, and persistent ChromaDB-based attack-memory reuse.

4 Experimental Implementation and Evaluation Setup

The proposed multi-layer intrusion detection framework pipelines (Section 3) were implemented in a Python-based environment using PyCharm as the development environment. The experiments were executed on an Ubuntu 24.04.4 LTS remote workstation equipped with an Intel Core i5-8600K CPU running at 3.60 GHz, 6 CPU cores, 64 GB RAM, 4 GiB swap, 1 TB disk storage, and an NVIDIA GeForce RTX 3080 GPU with 10 GB memory. The LLM component was implemented using Ollama version 0.19.0. The XGBoost classifiers were implemented using XGBoost version 3.1.2 with Python 3.12.3. The final experimental implementation was based on the V5 F0.5 all-layer calibrated pipeline, in which Gate-1 applies Q-learning-based confidence thresholds, Gate-2 performs ChromaDB memory matching, and Gate-3 applies F0.5-calibrated LLM attack confirmation with weighted-fusion fallback and review-based abstention. For consistency, each layer is evaluated on the first 5,000 events from its held-out test split, which provides stable estimates while keeping runtime manageable for repeated experiments and LLM escalation analysis.

4.1 Static Threshold Baseline IDS

This subsection first evaluates the implementation of the three-layer intrusion detection pipeline using a static confidence threshold. This implementation serves as a baseline for comparison with the proposed IDS pipeline, where a fixed confidence threshold of 0.85 is applied across all layers (NIDS, HIDS, and hypervisor-IDS), as described in Section 3. The value 0.85 is selected to provide a strict static-confidence baseline and to remain consistent with the fixed-threshold setting used in a recent cloud IDS study [42].

4.1.1 Network-Based IDS (NIDS)

In this study, the publicly available CICIDS-2018 dataset [66] is used for the development of NIDS as it provides realistic, labelled network traffic covering diverse attack categories including DoS, DDoS, brute-force, and web attacks with rich flow-based features suitable for ML-based network intrusion detection. To detect network traffic attacks, a binary classification setting is adopted, where all attack types in the dataset are grouped into a single attack class, while normal traffic represents the benign class (also described in Section 3). The XGBoost model is trained using the top 40 features selected through feature-importance ranking and validation, where this feature subset was retained because it preserved detection performance while reducing computational cost compared with using the full feature set. Model performance is evaluated on unseen test data using standard metrics, including precision, recall, and F1-score.

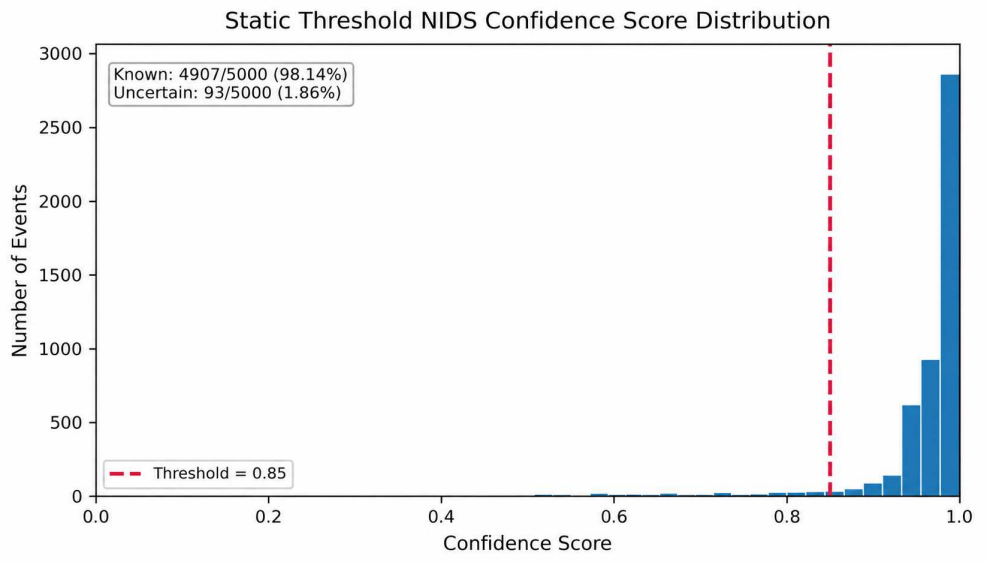

Fig. 4 Static confidence-score distribution of the NIDS component using a fixed 0.85 threshold.

For NIDS, flows with confidence above the selected threshold (0.85) are treated as known events by XGBoost, while those below the threshold are escalated to the LLM. The confidence distribution and routing outcomes are shown in Fig. 4. The red dashed vertical line denotes the confidence threshold. Bars to the right are high-confidence (known) predictions, and bars to the left are low-confidence (uncertain) predictions routed to the LLM. In the evaluation subset of 5,000 flows, 4,907 flows (98.14%) are classified as high-confidence known events, while 93 flows (1.86%) are identified as low-confidence events and escalated to the LLM based on the fixed 0.85 confidence threshold. The example in Table 1 presents an uncertain NIDS event extracted from the structured JSONL log file, which records all flows escalated to the LLM during the 5,000-flow evaluation.

4.1.2 Host-Based IDS (HIDS)

To implement the host-based IDS pipeline, experiments are conducted on the LID-DS 2019 dataset [67], a benchmark for evaluating Linux HIDS using system-call traces. This dataset is selected for its labeled host-level data that captures both normal and attack behaviors, aligning with the requirements of the host layer. Unlike structured network-flow datasets, LID-DS-2019 consists of host-level log records that exhibit greater semantic ambiguity and behavioral overlap between benign and attack activities, making host-level intrusion detection inherently more challenging. Consistent with the unified setup across all layers, the classifier performs binary classification, predicting benign or attack classes. Events with confidence below 0.85 are treated as uncertain HIDS events and escalated to the LLM, whereas those with confidence greater than or equal to 0.85 are treated as known events by the XGBoost classifier.

Table 1 Description of uncertain NIDS event promoted as ATTACK by LLM

Field	Description
Event Type	Suspected FTP Brute-Force Activity (NIDS)
ML Confidence Score	0.5807
Routing Decision	*Uncertain* → LLM analysis
Observed Flow Entry	`[NIDS FLOW] Dst Port=21, Protocol=6, Flow Duration=2, Tot Fwd Pkts=1, Tot Bwd Pkts=1, Flow Pkts/s=1000000, ...`
LLM-Based Interpretation	The high packet rate and short flow duration suggest a potential brute-force attack on port 21, which is commonly used for FTP.
LLM Output	label=`ATTACK`, confidence=`0.80`, attack_type=`FTP brute force`

For proper evaluation, a sample of 5,000 events is used to ensure stable estimates while maintaining computational efficiency for repeated evaluations and LLM escalation analysis. Of these 5,000 events, 2,141 events (42.82%) are classified as high-confidence known events, while 2,859 events (57.18%) are marked as uncertain and escalated to the LLM under the fixed 0.85 static threshold. The HIDS prediction confidence score distribution, prediction outcome distribution, and details are shown in Fig. 5. Low-confidence events identified in Fig. 5 are marked as uncertain and forwarded to the LLM layer for further analysis. Confirmed attack outputs generated through LLM-assisted analysis are stored in ChromaDB, while unresolved cases remain in structured JSONL review logs for audit and future analysis. An example case scenario is presented in Table 2.

Table 2 Description of uncertain HIDS event via LLM

Field	Description
Event Type	Uncertain Brute-Force activity (CWE-307)
ML confidence score	0.38
Routing decision	*Uncertain* → LLM analysis
Observed Log Entry	`apache2 ... > poll fds=12:41 timeout=5000`
LLM-Based Interpretation	The `apache2` process repeatedly enters a polling state with extended timeouts, which is atypical for normal web request handling. This behavior is commonly associated with automated clients probing authentication endpoints at a low rate to evade detection. Although a single log entry does not explicitly indicate credential failures, the execution pattern and timing characteristics suggest a potential brute-force probing attempt. Correlation with adjacent authentication logs is recommended for confirmation.

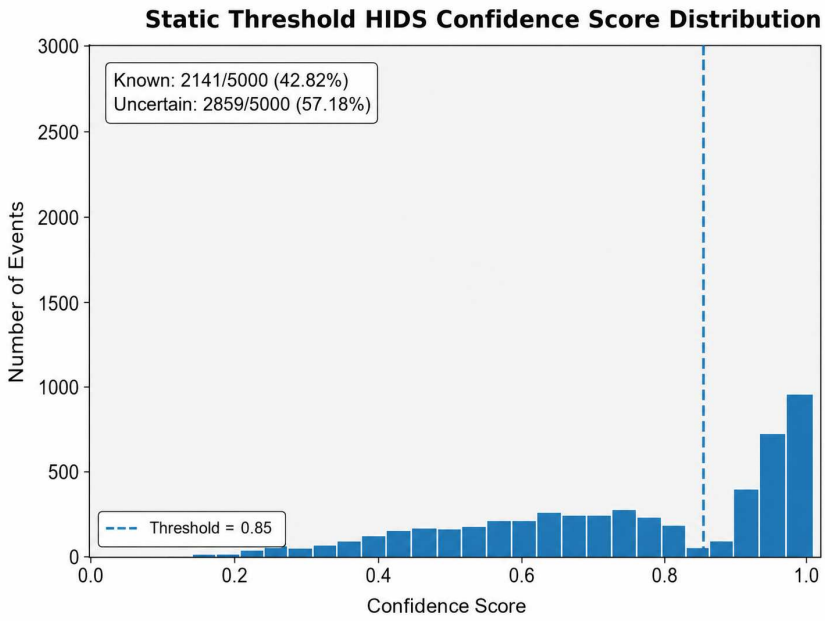

Fig. 5 Static confidence-score distribution of the HIDS component using a fixed 0.85 threshold.

4.1.3 Hypervisor based IDS

In order to implement the complete IDS pipeline, we have generated a synthetic hypervisor-based dataset for validation of our methodology. The dataset expresses a natural imbalance with a high ratio of noise, with 74% benign and 26% malicious events. Please note: despite an extensive search, we did not find any publicly available and labeled dataset for hypervisor-level intrusion detection. This is mainly due to the sensitive and privileged nature of hypervisor logs, which are rarely released by cloud providers. Therefore, we constructed a realistic synthetic hypervisor dataset to enable controlled and reproducible evaluation of our proposed approach. The summary of the generated hypervisor-based dataset is shown in Table 3.

Following the same 5,000-event evaluation and static-threshold protocol defined earlier for NIDS and HIDS, the hypervisor layer produced 4,841 known events (96.82%) and 159 uncertain events (3.18%), as shown in Fig. 6. As described earlier, events below the threshold are escalated to the LLM; confirmed attack outputs are stored in ChromaDB, while unresolved or inconclusive events remain in structured review logs for future audit and analysis.

Table 4 presents a representative low-confidence hypervisor event with an LLM-generated explanation. Inconclusive cases are retained for review rather than being automatically stored as confirmed attacks. The LLM explanations presented in this work are based on the actual responses recorded in the LLM audit logs. For readability and space considerations, they have been slightly paraphrased while preserving their original meaning and intent.

Table 3 Synthetic Hypervisor Dataset Summary

Property	Value
Records	25,000
Hypervisor Type	VMware ESXi, KVM, Xen, Hyper-V (Type-1)
Target Layer	Hypervisor
Features / Columns	24 total columns
Class Labels (for detection)	Binary ground truth: 2 classes (0=benign, 1=attack)
Distribution of Data	normal: 12500; vm lateral-movement: 2541; vm escape: 2,501; snapshot abuse: 2500; hypervisor dos: 2488; hyper-jacking: 2470
Pipeline Split (current)	Runtime random split in pipeline: 80/20 train/test

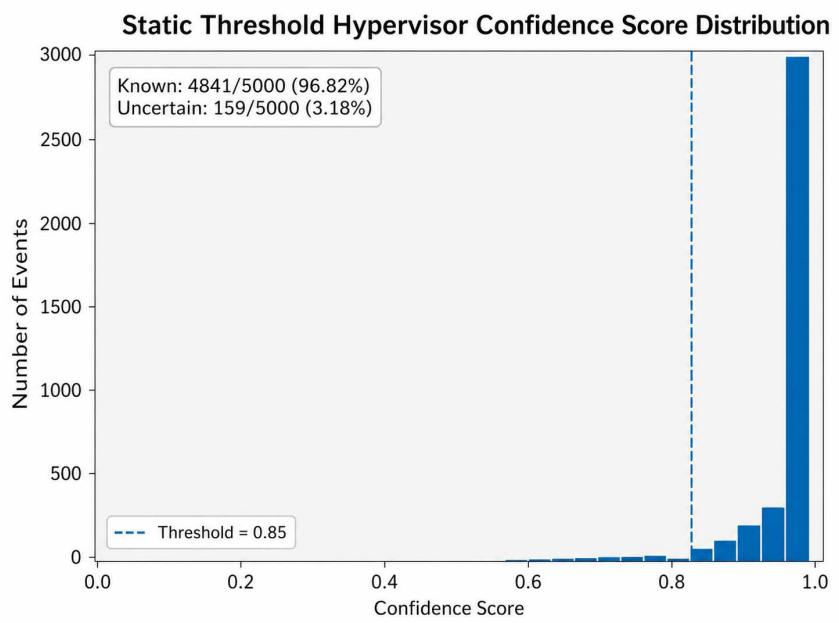

Fig. 6 Static confidence-score distribution at the hypervisor layer using a fixed 0.85 threshold.

The distribution of known and uncertain events across all three layers (NIDS, HIDS, and hypervisor-IDS) at the fixed confidence threshold of 0.85 is presented in Table 5. The results show substantial variation in uncertainty routing across the three datasets. While the NIDS and Hypervisor layers classify 98.14% and 96.82% of events as known, respectively, the HIDS layer routes 57.18% of its events as uncertain. This

Table 4 Description of Hypervisor uncertain event via LLM

Field	Description
Event Type	Uncertain Event U2
ML Confidence Score	0.58
Routing Decision	*Uncertain* → LLM analysis
Model/Stage Explanation	This event reflects ambiguous system-level behavior with partial similarity to early-stage hypervisor attack patterns, such as irregular VM execution flow and transient low-level resource access. However, key attack characteristics such as sustained malicious activity, coordinated follow-up events, or clear exploit signatures are not observed. Consequently, the statistical model reports low confidence, and the event is escalated for semantic analysis to avoid premature or unsafe mitigation actions.
LLM Verdict	Suspicious but inconclusive; manual review or contextual correlation advised.

indicates that a single fixed threshold does not adapt well to heterogeneous layer-specific data characteristics. This observation motivates the need for the proposed adaptive threshold-learning mechanism introduced in the following subsection.

Table 5 Known and uncertain event distribution using a fixed threshold of 0.85

Layer	Threshold	Known	Known (%)	Uncertain	Uncertain (%)
NIDS	0.85	4,907	98.14	93	1.86
HIDS	0.85	2,141	42.82	2,859	57.18
Hypervisor	0.85	4,841	96.82	159	3.18
Overall	–	**11,889**	**79.26**	**3,111**	**20.74**

4.2 Q-Learning-Based Adaptive IDS Pipeline

This subsection presents the complete Q-learning-calibrated IDS pipeline, building on the framework described in Section 3. To address the limitations of static thresholding, an RL-based Q-learning approach is employed to autonomously determine layer-specific confidence thresholds.

4.2.1 RL-Based Confidence Threshold Calibration

Following the same held-out 5,000-event evaluation protocol used for the static-threshold baseline, the RL-based thresholding is evaluated for each IDS layer to ensure a fair and direct comparison. The underlying XGBoost classifier remains unchanged. As described in Algorithm 1, Gate 1 applies the learned layer-specific confidence threshold to route each event either as a known event or as an uncertain event. Table 6 shows the learned threshold values and routing percentages obtained for the proposed IDS pipeline.

Table 6 Learned thresholds and routing percentages for the proposed adaptive IDS pipeline

Layer	Threshold	Known	Known (%)	Uncertain	Uncertain (%)
NIDS	0.79	4,933	98.66	67	1.34
HIDS	0.78	2,465	49.30	2,535	50.70
Hypervisor	0.73	4,888	97.76	112	2.24
Overall	–	**12,286**	**81.91**	**2,714**	**18.09**

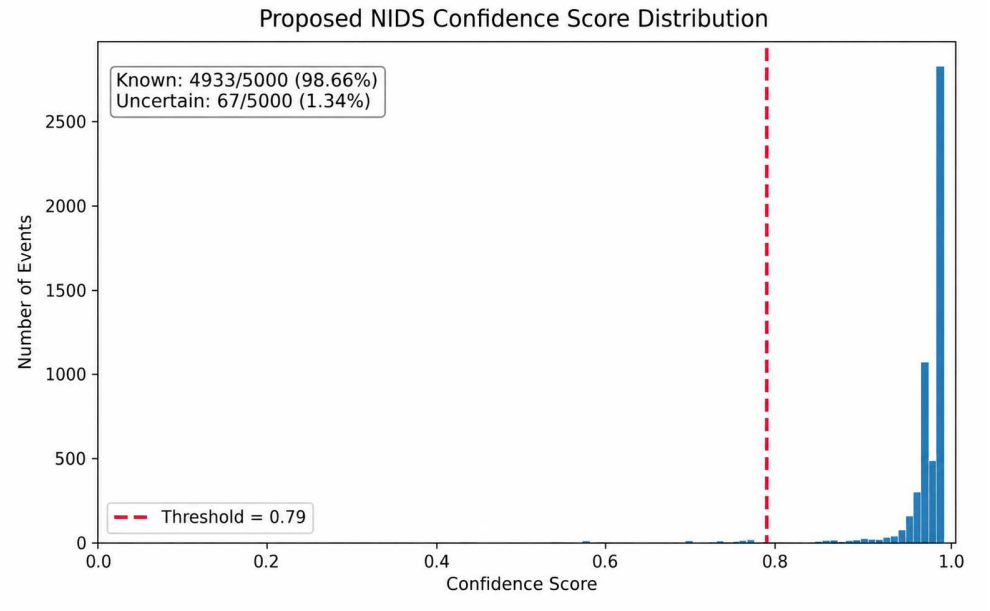

Fig. 7 Proposed adaptive NIDS confidence-score distribution using the learned threshold of 0.79.

The results obtained by implementing the proposed approach on the network layer show that the learned threshold of 0.79 maintains high-confidence detection while reducing unnecessary escalation. Under this threshold, 4,933 out of 5,000 events (98.66%) are classified as known, while only 67 events (1.34%) are forwarded to the LLM as uncertain cases. The NIDS confidence for the events, shown in Fig. 7, depicts that most of the network traffic events have a very high confidence close to 1.0, whereas only a small number of low-confidence events fall below the learned confidence threshold. This indicates that the learned NIDS threshold is operationally efficient, maintaining high-confidence acceptance while limiting uncertain escalations.

For the HIDS layer, the learned threshold converged to 0.78, resulting in 2,465 known events (49.30%) and 2,535 uncertain events (50.70%). Compared with the static threshold, the adaptive threshold reduces the number of uncertain HIDS events from 2,859 to 2,535, but the host layer still remains the most challenging due to the semantic ambiguity and behavioral overlap present in system-call traces. The HIDS confidence, shown in Fig. 8, demonstrates that in host-based traffic, the confidence

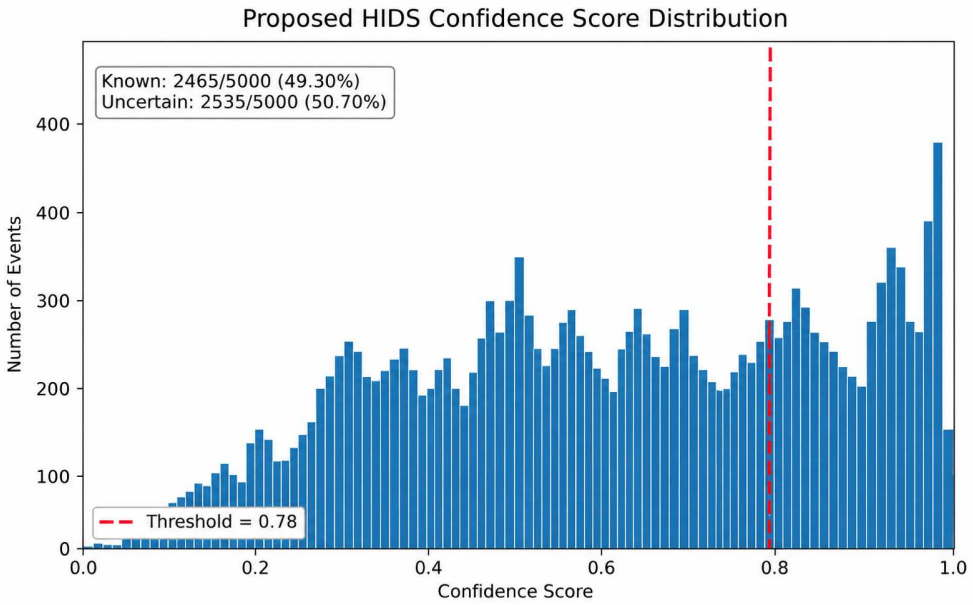

Fig. 8 Proposed adaptive HIDS confidence-score distribution using the learned threshold of 0.78.

scores are more widely spread, with notable events lying in the mid-confidence range, showing the ambiguous nature of host-based logs. This behaviour is mainly due to the characteristics of the LID-DS-2019 dataset, which contains host-level log data with greater ambiguity than structured network-flow data. Consequently, confidence scores are more widely distributed, resulting in a larger proportion of uncertain events. This highlights the importance of adaptive thresholding for host-level intrusion detection.

In the case of the hypervisor layer, the learned threshold converged to 0.73 and generated 4,888 known events (97.76%) and 112 uncertain events (2.24%). This shows that the learned threshold accepts the majority of high-confidence hypervisor events while reducing escalation from 159 static uncertain events to 112 adaptive uncertain events. The distribution of prediction confidence scores produced by the RL-based pipeline is presented in Fig. 9, which shows that the learned threshold separates the small set of uncertain events from the large majority of high-confidence detection events. This enables largely autonomous hypervisor-level intrusion detection with minimal escalation.

Comparing the proposed RL-based pipeline with the static-threshold pipeline, as summarized in Table 7, shows that adaptive thresholding reduces the overall uncertain workload from 3,111 events under the fixed 0.85 baseline to 2,714 events under the learned thresholds. This corresponds to a reduction of 397 events, or 12.76%, before memory reuse is applied. The comparison shows that the benefit of adaptive thresholding differs across layers because each dataset has a different confidence-score distribution. The NIDS and hypervisor layers already produce highly confident predictions, while the HIDS layer remains more uncertain because host-level logs are

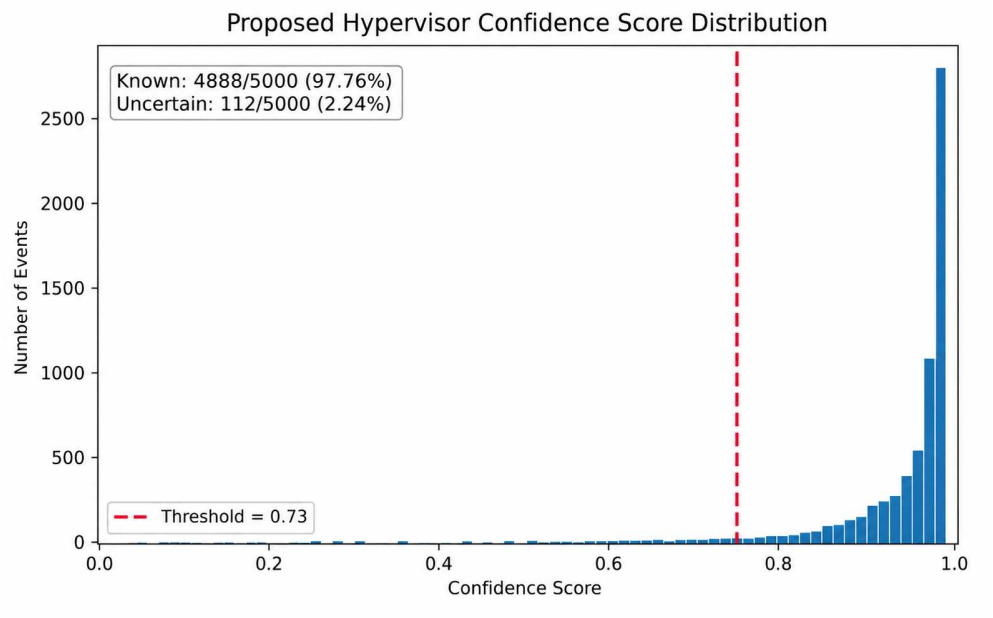

Fig. 9 Proposed adaptive hypervisor confidence-score distribution using the learned threshold of 0.73.

Table 7 Static vs proposed uncertainty routing across IDS layers

Layer	Mode	Known (%)	Uncertain (%)	Threshold
NIDS	Static	98.14	1.86	0.85
	Proposed	98.66	1.34	0.79
HIDS	Static	42.82	57.18	0.85
	Proposed	49.30	50.70	0.78
Hypervisor	Static	96.82	3.18	0.85
	Proposed	97.76	2.24	0.73

semantically noisier and more overlapping. *The primary limitation of static confidence thresholds is that they do not account for heterogeneous layer-specific data characteristics, including data structure, semantic ambiguity, and confidence-score distributions. In contrast, the proposed RL-based method learns layer-specific thresholds, allowing the pipeline to balance high-confidence acceptance with controlled LLM escalation across the three IDS layers.*

4.3 Gate-1 and Gate-2 Routing Analysis

After learning the thresholds successfully, events were routed through the Gate-1 and Gate-2 mechanisms described in Section 3 and summarized in Algorithm 1. The learned confidence thresholds were 0.79 for NIDS, 0.78 for HIDS, and 0.73 for the Hypervisor layer. Under these thresholds, Gate-1 produced 2,714 uncertain events

across the three layers, consisting of 67 NIDS events, 2,535 HIDS events, and 112 Hypervisor events.

During the cold execution, no memory-based matches were observed at Gate-2 because Chroma memory did not contain previously confirmed attack embeddings. Consequently, all 2,714 uncertain events bypassed Gate-2 and were forwarded directly to Gate-3 for LLM-based analysis. During the warm execution, the same uncertain-event set was processed again after confirmed attacks from the cold run had been stored in ChromaDB. In this setting, Gate-2 resolved 2,068 uncertain events through memory matching and reduced the number of actual LLM calls from 2,714 in the cold run to 646 in the warm run. This subsection therefore analyzes both cold-run routing behaviour and warm-run memory reuse.

4.4 Gate-3 LLM Validation and Fusion-Based Decision Making

Gate-3 implementation follows the calibrated LLM validation and weighted-fusion mechanism described in Section 3. As described in Eq. (3), the layer-specific LLM attack-confirmation thresholds were 0.71 for the NIDS layer, 0.50 for the HIDS layer, and 0.50 for the Hypervisor layer. These thresholds were selected using F0.5-oriented calibration rather than the earlier precision-constrained calibration. This subsection presents the routing outcomes observed after applying these calibrated thresholds during cold and warm evaluation.

During the cold execution, Gate-3 processed all 2,714 uncertain events because no Gate-2 memory matches were available. For the NIDS layer, 67 uncertain events were forwarded to the LLM. Among these, 42 events were accepted as ATTACK through the direct F0.5-calibrated LLM rule, 4 additional borderline ATTACK cases were accepted through the weighted-fusion fallback, and 21 events were retained for review. For the HIDS layer, 2,535 uncertain events were forwarded to the LLM. Of these, 1,703 events were accepted as ATTACK through the direct LLM rule, 634 events were classified as BENIGN , and 198 events were retained for review. For the Hypervisor layer, 112 uncertain events were analyzed by the LLM; however, none satisfied the calibrated attack-confirmation criteria, and all 112 events were retained for review.

Overall, the cold run produced 1,749 final attack decisions from Gate-3, consisting of 1,745 direct F0.5-calibrated LLM confirmations and 4 weighted-fusion confirmations. In addition, 634 events were classified as benign and 331 events were retained in the review/abstention bucket. These results show that the weighted-fusion fallback was rarely activated, contributing only four borderline NIDS attack confirmations, while most accepted attack decisions were made through the direct calibrated LLM rule.

During the warm execution, Gate-2 memory matching resolved 2,068 uncertain events before LLM invocation. As a result, only 646 events required Gate-3 LLM analysis. In the warm run, all 67 uncertain NIDS events were resolved through ChromaDB memory matching, resulting in zero NIDS LLM calls. For the HIDS layer, 2,001 events were resolved through memory matching, while the remaining 534 events were forwarded to the LLM; among these, 336 were accepted as ATTACK through the direct F0.5-calibrated LLM rule, 143 were classified as BENIGN, and 55 were retained for

review. For the Hypervisor layer, no memory matches were found, and all 112 uncertain events were forwarded to the LLM; all remained in the review bucket.

Overall, the warm run produced 2,404 final attack decisions, including 2,068 ChromaDB memory-based attack decisions and 336 direct LLM-confirmed attacks. It also produced 143 benign decisions and 167 review/abstention cases. No weighted-fusion promotions occurred in the warm run. These results demonstrate that Gate-3 remains conservative for unresolved or low-confidence cases, while Gate-2 memory reuse substantially reduces repeated LLM invocations after confirmed attack knowledge has been stored.

4.5 Post-Detection Persistence and Analyst Review

Post-detection persistence and review follow the mechanism described in Section 3. In the cold run, the review/abstention bucket contained 331 events in total, comprising 21 events from NIDS, 198 from HIDS, and 112 from the Hypervisor layer. These events correspond to cases where the LLM remained uncertain or where the response did not satisfy the calibrated confirmation criteria required for automatic attack promotion.

In the warm run, ChromaDB memory reuse reduced the review/abstention bucket to 167 events, comprising 0 events from NIDS, 55 from HIDS, and 112 from the Hypervisor layer. This reduction is mainly due to the reuse of confirmed NIDS and HIDS attack knowledge stored during the cold run. However, the Hypervisor layer retained 112 review cases because no reliable memory matches or calibrated LLM-confirmed attack decisions were produced for these uncertain hypervisor events. The retained review-bucket events represent cases requiring additional contextual evidence or expert assessment. Therefore, the review mechanism prevents forced automatic classification of unresolved events and supports traceability, auditability, and future feedback-driven refinement of the proposed framework.

5 Performance Evaluation and Analysis

In our study, we found that the proposed RL-based pipeline improves routing efficiency by reducing unnecessary LLM escalations while maintaining reliable detection performance. The evaluation further considers both cold and warm executions to quantify the effect of ChromaDB-based memory reuse on repeated LLM calls.

Observed Escalation Analysis: For the experimental setup with 5,000 test events per layer, the static threshold baseline with a fixed confidence threshold of 0.85 produced uncertain-event counts of 93 for NIDS, 2,859 for HIDS, and 159 for the Hypervisor layer. In contrast, the proposed RL-based adaptive routing produced uncertain-event counts of 67 for NIDS, 2,535 for HIDS, and 112 for the Hypervisor layer. Thus, the total number of uncertain events, denoted as N_{unc}, produced by the proposed adaptive scheme during cold execution is

$$N_{\text{unc}}^{\text{RL,cold}} = 67 + 2535 + 112 = 2714, \tag{7}$$

and for the static scheme is,

$$N_{\text{unc}}^{\text{static}} = 93 + 2859 + 159 = 3111. \tag{8}$$

The reduction in Gate-1 uncertain events achieved by the proposed RL-based adaptive thresholding before memory reuse is therefore,

$$\Delta N_{\text{unc}}^{\text{static}\to\text{cold}} = N_{\text{unc}}^{\text{static}} - N_{\text{unc}}^{\text{RL,cold}} = 3111 - 2714 = 397, \tag{9}$$

where ΔN_{unc} denotes the difference in uncertain-event counts between the static baseline and the proposed adaptive cold run. The percentage reduction is,

$$\text{Reduction}_{\text{static}\to\text{cold}}(\%) = \frac{397}{3111} \times 100 = 12.76\%. \tag{10}$$

Although the adaptive thresholds provide a moderate reduction in Gate-1 uncertainty, the major LLM-workload reduction is achieved during warm execution through ChromaDB memory reuse. In the cold run, all 2,714 uncertain events are forwarded to the LLM. In the warm run, Gate-2 memory matching resolves 2,068 uncertain events before LLM invocation, reducing actual LLM calls to 646.

$$N_{\text{LLM}}^{\text{warm}} = 646, \tag{11}$$

$$\Delta N_{\text{LLM}}^{\text{cold}\to\text{warm}} = 2714 - 646 = 2068, \tag{12}$$

$$\text{Reduction}_{\text{cold}\to\text{warm}}(\%) = \frac{2068}{2714} \times 100 = 76.20\%. \tag{13}$$

Compared with the static baseline workload, the warm adaptive pipeline reduces LLM calls from 3,111 to 646, giving:

$$\Delta N_{\text{LLM}}^{\text{static}\to\text{warm}} = 3111 - 646 = 2465, \tag{14}$$

$$\text{Reduction}_{\text{static}\to\text{warm}}(\%) = \frac{2465}{3111} \times 100 = 79.24\%. \tag{15}$$

Across 15,000 evaluated events, the proposed adaptive pipeline first reduces Gate-1 uncertainty from 3,111 static uncertain events to 2,714 adaptive uncertain events. After confirmed attack knowledge is stored and reused through ChromaDB, the warm run further reduces actual LLM calls to 646. This lowers the LLM workload from 20.74% of all events under the static baseline to 4.31% under the warm adaptive pipeline.

Estimated Cost Reduction Analysis: Since LLM inference APIs are billed per token [68, 69], the total inference cost (C_{total}) scales directly with the number of escalated events. For a routing policy producing N_{LLM} LLM calls, the total cost is:

$$C_{\text{total}} = N_{\text{LLM}} \times c_{\text{event}}, \tag{16}$$

where c_{event} is the cost per escalated event. In the static baseline, the total LLM workload is 3,111 events. In the adaptive cold run, this workload is reduced to 2,714 LLM calls due to the learned Gate-1 thresholds. In the adaptive warm run, ChromaDB

memory reuse further reduces the actual LLM calls to 646. The cost saving of the proposed warm adaptive pipeline over the static scheme can be calculated as,

$$\Delta C_{\text{static}\to\text{warm}} = \left(N_{\text{LLM}}^{\text{static}} - N_{\text{LLM}}^{\text{warm}}\right) \times c_{\text{event}} = (3111 - 646) \times c_{\text{event}} = 2465 \times c_{\text{event}}. \tag{17}$$

Since c_{event} is a fixed cost determined by the choice of LLM provider and prompt design [70], Eq. (17) shows that cost saving scales linearly with the reduction in LLM calls. Compared with the static baseline, the warm adaptive pipeline reduces the LLM workload by 79.24%. The cold-to-warm reduction is 76.20%, showing that the major cost benefit comes from ChromaDB-based memory reuse after confirmed attack knowledge has been stored.

Table 8 LLM workload comparison between static, adaptive cold, and adaptive warm routing.

Mode	LLM Workload	Reduction Baseline	Reduction (%)
Static baseline	3,111	–	–
Adaptive cold	2,714	Static → Cold	12.76
Adaptive warm	646	Static → Warm	79.24

Routing Selectivity and Detection Performance: Table 9 summarizes the warm-run routing selectivity and final decision outcomes of the proposed pipeline after ChromaDB memory reuse. In the adaptive cold run, Gate-1 produced 2,714 uncertain events from the 15,000 evaluated events, all of which were forwarded to the LLM because no ChromaDB memory matches were available. In the warm run, confirmed attack knowledge stored from the cold execution allowed Gate-2 to resolve 2,068 uncertain events before LLM invocation, reducing actual LLM calls from 2,714 to 646.

Table 9 Warm-run routing and final decision outcomes across the three IDS layers.

Layer	Uncertain events	Memory hits	LLM calls	Final attack	Final benign	Review / abstention
NIDS	67	67	0	67	0	0
HIDS	2,535	2,001	534	2,337	143	55
HypIDS	112	0	112	0	0	112
Overall	2,714	2,068	646	2,404	143	167

To further explain how uncertain events are finalized, Table 10 presents the decision-source breakdown for cold and warm executions. The table separates memory-based decisions, direct F0.5-calibrated LLM confirmations, weighted-fusion fallback confirmations, LLM-benign decisions, and review/abstention cases.

The decision-source breakdown shows that the cold run depends entirely on Gate-3, whereas the warm run shifts most uncertain-event handling to ChromaDB memory matching. In the cold run, 1,745 uncertain events are confirmed directly by the

Table 10 Decision-source breakdown for cold and warm uncertain-event processing.

Run	Layer	Chroma memory	Direct LLM threshold	Weighted fallback	LLM benign	Review / unsure
Cold	NIDS	0	42	4	0	21
	HIDS	0	1,703	0	634	198
	Hypervisor	0	0	0	0	112
	Overall	**0**	**1,745**	**4**	**634**	**331**
Warm	NIDS	67	0	0	0	0
	HIDS	2,001	336	0	143	55
	Hypervisor	0	0	0	0	112
	Overall	**2,068**	**336**	**0**	**143**	**167**

F0.5-calibrated LLM threshold, while only four additional NIDS events are confirmed through the weighted-fusion fallback. In the warm run, 2,068 uncertain events are resolved through memory matching, leaving only 336 direct LLM-confirmed attacks and no weighted-fusion promotions. This confirms that weighted fusion acts only as a conservative secondary confirmation path, while the main warm-run efficiency gain comes from persistent memory reuse.

The warm-run routing results also show that 2,404 uncertain events are finalized as attacks, 143 events are classified as benign, and 167 events remain in the review/abstention bucket. This behaviour is important for operational IDS deployment because unresolved cases remain auditable and available for expert review rather than being forced into unreliable automatic decisions.

Table 11 presents the end-to-end detection performance of the proposed three-layer pipeline under the complete 15,000-event warm-run evaluation. For conservative binary reporting, review/abstention cases are counted as non-attack. Under this setting, the proposed framework achieves an overall accuracy of 94.63%, precision of 94.10%, recall of 93.79%, and F1-score of 93.95%. At the layer level, the NIDS and Hypervisor-based IDS obtain strong F1-scores of 96.31% and 97.26%, respectively, while the HIDS layer achieves an F1-score of 89.86% despite the greater semantic ambiguity and behavioural overlap present in host-level system-call traces.

Table 11 End-to-end detection performance of the proposed three-layer IDS pipeline under warm-run evaluation.

Layer	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)
NIDS	97.88	96.71	95.91	96.31
HIDS	88.66	88.83	90.92	89.86
HypIDS	97.34	98.79	95.78	97.26
Overall	94.63	94.10	93.79	93.95

Cross-layer adaptation statement: The proposed method is evaluated across three heterogeneous environments and datasets: network-flow data (CICIDS2018), host-log data (LID-DS-2019), and synthetic hypervisor events with injected boundary/noise effects. Despite different feature spaces, attack semantics, and noise levels, the same three-stage framework adapts per layer and maintains stable overall performance. In

Table 12 Comparison of the proposed pipeline with related works.

Criterion	Fernández Saura [42]	Kalafatidis [39]	Adjewa [40]	Chen [41]	Proposed Work
Three-layer IDS (NIDS+HIDS+HypIDS)	✗	✗	✗	✗	✓
RL-based adaptive threshold per layer	✗	✗	✗	✗	✓
Selective LLM escalation	✓	✓	✗	✗	✓
Multi-gated attack decision mechanism	✗	✗	✗	✗	✓
Fixed-threshold independence	✗	✗	✗	✗	✓
Explicit escalation-cost analysis	✗	✗	✗	✗	✓
Static-threshold baseline comparison reported	✗	✗	✗	✗	✓
Hypervisor layer coverage	✗	✗	✗	✗	✓

✓= present, ✗= absent.

particular, the LID-DS-2019 dataset consists of host-level log records that are inherently more ambiguous than structured network-flow data, resulting in wider confidence distributions and a higher proportion of uncertain events. Nevertheless, the proposed framework successfully adapts to these differences through layer-specific learned thresholds and uncertainty-aware escalation. This demonstrates that the method is not tuned to a single dataset type but can operate across network, host, and virtualisation monitoring contexts.

5.1 Comparative Analysis with Existing IDS Frameworks

Drawing on the literature review in Section 2, most relevant LLM-assisted intrusion detection studies were identified and compared with the proposed pipeline, as shown in Table 12. The analysis indicates limited overlap, with notable differences in system scope and architectural design. Fernandez-Saura et al. [42] escalate low-confidence alerts to a local LLM within a federated setting, but operate on a single NIDS layer with a fixed confidence threshold of 0.85 and do not quantify escalation cost. Kalafatidis et al. [39] apply a two-tier LLM-enhanced detection strategy for containerised SDN environments, triggering on-demand LLM analysis via a static statistical threshold, with no host or hypervisor coverage and no adaptive routing. Adjewa et al. [40] dynamically identify unknown attack clusters using transformer encoders and Gaussian mixture models, achieving adaptive identification without retraining, but remain confined to network-level detection with no LLM escalation control or threshold learning. Chen et al. [41] combine NIDS and HIDS through a two-stage collaborative classifier with NLP-based host data transformation, extending coverage to two layers, yet still rely on a fixed decision boundary with no RL-based optimisation or LLM escalation gate.

Table 13 Quantitative performance comparison with related IDS studies. NR: Not explicitly reported in the original study; Macro Avg.: macro average F1-score; Near-direct: same dataset with related IDS scope; Partial: partially comparable due to differences in architecture or evaluation setup; Limited: comparison constrained by different datasets or deployment environments. Proposed results are reported under the complete warm-run evaluation.

Study	Dataset(s)	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Comparison scope
Adjewa et al. [40]	CSE-CIC-IDS2018	95.60	≈96	95.60	≈96	Near-direct (same NIDS benchmark family)
Chen et al. [41]	CICIDS2018, NDSec-1	NR	NR	NR	99.94[a]	Partial (same CICIDS2018 benchmark, hybrid setting)
Fernández Saura et al. [42]	CIC-IDS-2017	NR	NR	91.47	95.54	Limited (different dataset, federated setting)
Kalafatidis et al. [39]	Real SDN/Kubernetes testbed	NR	NR	NR	NR	Limited (real-world SDN/Kubernetes deployment)
Proposed NIDS layer	CICIDS2018	**97.88**	**96.71**	**95.91**	**96.31**	Direct NIDS evaluation under warm-run routing
Proposed HIDS layer	LID-DS-2019	**88.66**	**88.83**	**90.92**	**89.86**	Separate HIDS evaluation under warm-run routing
Proposed HypIDS layer	Synthetic hypervisor logs	**97.34**	**98.79**	**95.78**	**97.26**	Novel hypervisor-level evaluation under warm-run routing
Proposed overall pipeline	CICIDS2018 + LID-DS-2019 + Synthetic	**94.63**	**94.10**	**93.79**	**93.95**	Complete three-layer warm-run evaluation

[a]Macro average F1-score as reported in the original study.

In addition to the architectural comparison, Table 13 presents the quantitative performance metrics reported by related IDS studies alongside the proposed framework. The comparison includes reported accuracy, precision, recall, and F1-score values together with the corresponding datasets and deployment scopes. Since the reviewed studies differ in datasets, attack scopes, and deployment settings, direct metric comparison remains constrained. However, the proposed framework is distinct because it evaluates a heterogeneous three-layer cloud IDS pipeline spanning NIDS, HIDS, and hypervisor-level intrusion detection, while integrating RL-based adaptive thresholding, uncertainty-aware escalation, selective LLM invocation, ChromaDB memory reuse, and review-based abstention.

6 Conclusions and Future Work

This study presented a cloud-specific multi-layer IDS integrating XGBoost and LLM reasoning within a confidence-aware multi-gated architecture. Different from the existing approaches that rely on fixed classifiers and static LLM thresholds, the proposed framework employs Q-learning for adaptive threshold calibration per

layer and a weighted-fusion fallback mechanism for low-confidence ATTACK predictions generated by the LLM, enabling more reliable decision control across network, host, and hypervisor layers. Experimental results showed that the proposed pipeline reduced uncertain-event workload compared with the fixed-threshold static baseline and reduced repeated LLM invocations by 79.24% during warm execution through ChromaDB-based attack-memory reuse. The complete warm-run evaluation also demonstrated strong overall detection performance, achieving a 93.95% F1-score across the three-layer evaluation setting. The framework further improved operational efficiency by cautiously promoting uncertain attacks and routing unresolved low-confidence events to analyst review rather than forcing risky automatic decisions. These results demonstrate the practical value of combining RL, confidence calibration, memory-assisted retrieval, and LLM-assistance for cloud intrusion detection. Future work will focus on adversarial robustness, zero-day attack resilience, out-of-distribution detection, and real-time deployment in dynamic cloud environments using an agentic AI-based approach.

Acknowledgment

The authors acknowledge the financial support by the German Federal Ministry of Research, Technology and Space (BMFTR) within the projects > Open6GHub+ < {16KIS2406} and > X-COM < {16KISS007K}. The authors also acknowledge the support from the University of Waikato, Hamilton, New Zealand.

References

[1] Baladari, V.: The role of software developers in transitioning on-premises applications to cloud platforms: Strategies and challenges. Journal of Scientific and Engineering Research **8**(1), 270–278 (2021)

[2] Othman, S.M., Al-mutawkkil, A.Y., Alnashi, A.M.: Survey of intrusion detection techniques in cloud computing. Journal of Applied Sciences and Technology **2**(4), 363–374 (2024) https://doi.org/10.59628/jast.v2i4.970

[3] Arogundade, O.R., Palla, K.: Virtualization revolution: Transforming cloud computing with scalability and agility. IARJSET (2023)

[4] Akhtar, N., Kerim, B., Perwej, Y., Tiwari, A., Praveen, S.: A comprehensive overview of privacy and data security for cloud storage. International Journal of Scientific Research in Science Engineering and Technology (2021)

[5] Alzahrani, A., Alenezi, M., Alshamrani, S.: Machine learning-based intrusion detection systems for cloud environments: Challenges and future directions. Computers & Security **137**, 103530 (2024)

[6] Donat, V., Haertel, C., Staegemann, D., Daase, C., Pohl, M., Dreschel, D., Walia, D.S., Turowski, K.: Examining the impact of cloud computing on organizational

performance: A systematic literature review. In: ICEIS 2025, vol. 1, pp. 375–386 (2025). https://doi.org/10.5220/0013478900003929

[7] Hashim, W., Hussein, N.A.-H.K.: Securing cloud computing environments: An analysis of multi-tenancy vulnerabilities and countermeasures. SHIFRA **2024**, 8–16 (2024)

[8] Tabrizchi, H., Kuchaki Rafsanjani, M.: A survey on security challenges in cloud computing: Issues, threats, and solutions. The Journal of Supercomputing **76**(12), 9493–9532 (2020) https://doi.org/10.1007/s11227-020-03213-1

[9] Ahmed, M., Mahmood, A.N., Hu, J.: A survey of network and host-based intrusion detection systems for cloud computing. Journal of Cloud Computing **12**, 127 (2023)

[10] Ahmadi, S.: Systematic literature review on cloud computing security: Threats and mitigation strategies. Journal of Information Security **15**(2), 148–167 (2024) https://doi.org/10.4236/jis.2024.152010

[11] Li, J., Wang, P., Xu, L.: Security threats and detection techniques in virtualized cloud environments. IEEE Access **11**, 84532–84547 (2023) https://doi.org/10.1109/ACCESS.2023.3298741

[12] Liu, H., Lang, B.: Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences **9**(20), 4396 (2019)

[13] Ahmad, Z., Khan, S., et al.: Network intrusion detection system: A systematic study of machine learning and deep learning approaches. IEEE Access (2021)

[14] Alzahrani, A., Alenazi, M.: Machine learning-based intrusion detection systems for cloud computing: A survey. Journal of Cloud Computing (2022)

[15] Zhang, Y., et al.: Deep learning-based intrusion detection system for cloud computing environments. IEEE Access (2023)

[16] Ferrag, M.A., Maglaras, L.: Security for cloud, iot and edge computing: A survey of intrusion detection systems. Computer Networks **223**, 109560 (2023)

[17] Aliferis, C., Simon, G.: Overfitting, underfitting and general model overconfidence and under-performance pitfalls and best practices in machine learning and ai. Artificial intelligence and machine learning in health care and medical sciences: Best practices and pitfalls, 477–524 (2024)

[18] Basiouni, A.M., El Rashid, M., Shaalan, K.: In-context learning in large language models (llms): Mechanisms, capabilities, and implications for advanced knowledge representation and reasoning. IEEE Access (2025)

[19] Xu, H., Wang, S., Li, N., Wang, K., Zhao, Y., Chen, K., Yu, T., Liu, Y., Wang,

H.: Large Language Models for Cyber Security: A Systematic Literature Review (2024)

[20] Jaffal, N.O., Alkhanafseh, M., Mohaisen, D.: Large language models in cybersecurity: A survey of applications, vulnerabilities, and defense techniques. AI **6**(9), 216 (2025) https://doi.org/10.3390/ai6090216

[21] Krishna, A., Zahid, F.: Response-level identification of cloud api misconfigurations using large language models. Pragmatic Cybersecurity **1**(2), 12 (2026) https://doi.org/10.53941/pc.2026.100012

[22] Lazer, S.J., Aryal, K., Gupta, M., Bertino, E.: A survey of agentic ai and cybersecurity: Challenges, opportunities and use-case prototypes. arXiv preprint arXiv:2601.05293 (2026)

[23] Zhang, J., Bu, H., Wen, H., Liu, Y., Fei, H., Xi, R., Li, L., Yang, Y., Zhu, H., Meng, D.: When LLMs meet cybersecurity: A systematic literature review. Cybersecurity **8**(1), 1–41 (2025) https://doi.org/10.1186/s42400-025-00361-w

[24] Watanabe, R., Okada, S., Watarai, K., Mitsunaga, T.: Cloud SecNavigator: RAG approach to bridge gaps and strengthen cloud security practices with RAGAS assessment. In: IEEE ICEET, pp. 1–6. IEEE, ??? (2024). https://doi.org/10.1109/iceet65156.2024.10913535

[25] Louati, F., Ktata, F.B., Amous, I.: Enhancing intrusion detection systems with reinforcement learning: A comprehensive survey of rl-based approaches and techniques. SN Computer Science (2024) https://doi.org/10.1007/s42979-024-03001-1

[26] Alavizadeh, H., Alavizadeh, H., Jang-Jaccard, J.: Deep q-learning based reinforcement learning approach for network intrusion detection. Computers **11**(3), 41 (2022) https://doi.org/10.3390/computers11030041

[27] Tamuka, N., Mathonsi, T.E., Olwal, T.O., Maswikaneng, S., Muchenje, T., Tshilongamulenzhe, T.M.: A comparative analysis of self-aware reinforcement learning models for real-time intrusion detection in fog networks. Future Internet **18**(2), 100 (2026) https://doi.org/10.3390/fi18020100

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

[29] Long, Z., Yan, H., Shen, G., Zhang, X., He, H., Cheng, L.: A transformer-based network intrusion detection approach for cloud security. Journal of Cloud Computing **13**(1) (2024) https://doi.org/10.1186/s13677-023-00574-9

[30] Joraviya, N., Gohil, B.N., Rao, U.P.: Dl-hids: Deep learning-based host intrusion

detection system using system calls-to-image for containerized cloud environment. The Journal of Supercomputing **80**, 12218–12246 (2024) https://doi.org/10.1007/s11227-024-05895-3

[31] Zatika, A., Coffman, J.: Improving ids performance with xgboost hyperparameter optimization and real-time analysis. In: IEEE/ACM 17th UCC. IEEE, ??? (2024). https://doi.org/10.1109/UCC63386.2024.00037

[32] Dhaliwal, S.S., Nahid, A.-A., Abbas, R.: Effective intrusion detection system using XGBoost. Information **9**(7), 149 (2018) https://doi.org/10.3390/info9070149

[33] Gavric, N., Bhandari, G.P., Shalaginov, A.: Towards resource-efficient DDoS detection in IoT: Leveraging feature engineering of system and network usage metrics. Journal of Network and Systems Management **32**(4), 69 (2024) https://doi.org/10.1007/s10922-024-09848-2

[34] Iturbe-Araya, J.-I., Rifà-Pous, H.: Hyperparameter optimization and evaluation metrics for unsupervised anomaly-based cyberattack detection in imbalanced smart home datasets. Journal of Network and Systems Management **33**, 99 (2025) https://doi.org/10.1007/s10922-025-09973-6

[35] Hizal, S., ÇAVUŞOĞLU, E., AKGÜN, D.: A new deep learning based intrusion detection system for cloud security. In: 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp. 1–4 (2021). https://doi.org/10.1109/HORA52670.2021.9461285

[36] Geetha, T.V., Deepa, A.J., Linda, M.M.: Deep learning method for efficient cloud ids utilizing combined behavior and flow-based features. Applied Intelligence **54**(8), 6738–6759 (2024) https://doi.org/10.1007/s10489-024-05505-y

[37] Venkateshwarlu, V., Ranjith, D., Raju, A.: Lrdadf: An ai enabled framework for detecting low-rate ddos attacks in cloud computing environments. In: 5th International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–8 (2023). https://doi.org/10.1109/ICECCT56650.2023.10179834

[38] Agate, V., De Paola, A., Ferraro, P., Lo Re, G.: MIDES: A multi-layer intrusion detection system using ensemble machine learning. International Journal of Intelligent Networks **6**, 204–223 (2025) https://doi.org/10.1016/j.ijin.2025.09.001

[39] Kalafatidis, S., Papageorgopoulos, N., Kartakoullis, A., Ledakis, G.: Two-tier intrusion detection with lightweight statistical monitoring and on-demand llm-based traffic analysis. In: Availability, Reliability and Security (ARES 2025 Workshops), vol. 15998, pp. 55–73. Springer, ??? (2025). https://doi.org/10.1007/978-3-032-00642-4_4

[40] Adjewa, F., Esseghir, M., Merghem-Boulahia, L., Kacfah, C.: Llm-based continuous intrusion detection framework for next-gen networks. In: IEEE IWCMC, pp. 1198–1203 (2025). IEEE

[41] Chen, Z., Simsek, M., Kantarci, B., Bagheri, M., Djukic, P.: Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. Computer Networks **250**, 110576 (2024) https://doi.org/10.1016/j.comnet.2024.110576 . Available online 7 June 2024

[42] Saura, P.F., Bernabé, J.B., Gómez, A.S.: Enhancing federated intrusion detection through llm-driven alert enrichment and collaborative threat information sharing. Future Generation Computer Systems **178**, 108319 (2026) https://doi.org/10.1016/j.future.2025.108319

[43] Blefari, F., Cosentino, C., Pironti, F., Furfaro, A., Marozzo, F.: CyberRAG: An Agentic RAG Cyber Attack Classification and Reporting Tool. arXiv preprint (arXiv:2507.02424v2) (2025)

[44] Jaber, A.N., Rehman, S.U.: Fcm–svm based intrusion detection system for cloud computing environment. Cluster Computing **23**(4), 3221–3231 (2020) https://doi.org/10.1007/s10586-020-03082-6

[45] Alazab, M., Awajan, A., Obeidat, A., Faruqui, N., Bere, A., Ali, S., Wei, W.: Adaptive protocols for hypervisor security in cloud infrastructure using federated learning-based anomaly detection. Engineering Applications of Artificial Intelligence **152**, 110750 (2025) https://doi.org/10.1016/j.engappai.2025.110750

[46] Anumukonda, N.S.K., Yadav, R.K., Raghava, N.S.: Hypervisor based intrusion detection using enhanced radial basis neural network on cloud environment. In: IEEE IATMSI, vol. 2, pp. 1–6 (2024). https://doi.org/10.1109/IATMSI60426.2024.10503284

[47] Qaffas, A.A.: Reinforcement learning driven self-adaptation in hypervisor-based cloud intrusion detection systems (rldac-ids). International Journal of Advanced Computer Science and Applications **15**(7) (2024) https://doi.org/10.14569/IJACSA.2024.0150745

[48] Kheddar, H., Dawoud, D.W., Awad, A.I., Himeur, Y., Khan, M.K.: Reinforcement-learning-based intrusion detection in communication networks: A review. IEEE Communications Surveys & Tutorials **27**(4), 2420–2469 (2025) https://doi.org/10.1109/COMST.2024.3484491

[49] Hammar, K., Stadler, R.: Intrusion prevention through optimal stopping. IEEE Transactions on Network and Service Management **19**(3), 2333–2348 (2022) https://doi.org/10.1109/TNSM.2022.3176781

[50] Huang, L., Zhu, Q.: RADAMS: Resilient and adaptive alert and attention

management strategy against informational denial-of-service (IDoS) attacks. Computers & Security **121**, 102844 (2022) https://doi.org/10.1016/j.cose.2022.102844

[51] Hossain, M.A.: Deep Q-learning intrusion detection system (DQ-IDS): A novel reinforcement learning approach for adaptive and self-learning cybersecurity. ICT Express **11**, 875–880 (2025) https://doi.org/10.1016/j.icte.2025.05.007

[52] Mogollón-Gutiérrez, Ó., Escudero García, D., Sancho Núñez, J.C., DeCastro-García, N.: Reinforcement learning for the optimization of adaptive intrusion detection systems. Engineering Proceedings **123**(1), 2 (2026) https://doi.org/10.3390/engproc2026123002

[53] Gouveia, A., Correia, M.: Network intrusion detection with xgboost. In: Distributed, Parallel and Secure Systems. INESC-ID, Universidade de Lisboa, ??? (2020). https://www.inesc-id.pt/

[54] Shwartz-Ziv, R., Armon, A.: Tabular data: Deep learning is not all you need. Information Fusion **81**, 84–90 (2022) https://doi.org/10.1016/j.inffus.2021.11.011

[55] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge, MA (2018)

[56] Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3–4), 279–292 (1992) https://doi.org/10.1007/BF00992698

[57] Liao, Y., Vemuri, V.R.: Use of k-nearest neighbor classifier for intrusion detection. Computers & security **21**(5), 439–448 (2002)

[58] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, *et al.*: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems **33**, 9459–9474 (2020)

[59] Fayyazi, R., Hoyos Trueba, S., Zuzak, M., Yang, S.J.: Proverag: Provenance-driven vulnerability analysis with automated retrieval-augmented LLMs. IEEE Access (2025) https://doi.org/10.1109/ACCESS.2025.3638251

[60] Christen, P., Hand, D.J., Kirielle, N.: A review of the f-measure: its history, properties, criticism, and alternatives. ACM Computing Surveys **56**(3), 1–24 (2023)

[61] Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(3), 226–239 (1998) https://doi.org/10.1109/34.667881

[62] Xu, Y., Lu, Y.: Adaptive weighted fusion: A novel fusion approach for image

classification. Neurocomputing **168**, 566–574 (2015) https://doi.org/10.1016/j.neucom.2015.05.070

[63] Nguyen, T.T., Luong, A.V., Dang, M.T., Liew, A.W.-C., McCall, J.: Ensemble selection based on classifier prediction confidence. Pattern Recognition **100**, 107104 (2020) https://doi.org/10.1016/j.patcog.2019.107104

[64] Ortega, A., Villegas, P., Auge, D., Marcos, E., Beltran, M.: A taxonomy for threat actors' persistence techniques. Computers & Security **120**, 102820 (2022) https://doi.org/10.1016/j.cose.2022.102498

[65] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., *et al.*: A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion **76**, 243–297 (2021) https://doi.org/10.1016/j.inffus.2021.05.008

[66] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: 4th International Conference on Information Systems Security and Privacy (ICISSP), pp. 108–116 (2018)

[67] Grimm, D., Grimm, M., Krebs, T., Kreuter, M., Hotho, A.: A modern and sophisticated host based intrusion detection data set. In: CD-MAKE 2019 – Machine Learning and Knowledge Extraction (2019)

[68] OpenAI: API Pricing. https://openai.com/pricing. Accessed: March 2026 (2024)

[69] Anthropic: Claude API Pricing. https://docs.anthropic.com/en/docs/about-claude/pricing. Accessed: March 2026 (2024)

[70] IntuitionLabs: LLM API Pricing Comparison (2025): OpenAI, Gemini, Claude. https://intuitionlabs.ai/articles/llm-api-pricing-comparison-2025. Accessed: March 2026 (2025)