A Dual-Head Model for Host based Intrusion Detection on System-Call traces

Matthias Dippold, Sofia Maragkou, Matthias Wess, Thilo Sauter
Institute of Computer Technology (ICT)
TU Wien
Vienna, Austria
mat.dippold@gmail.com, sofia.maragkou@tuwien.ac.at, matthias.wess@tuwien.ac.at, thilo.sauter@tuwien.ac.at

Grigorios Chrysos, Sotiris Ioannidis
Dept. of Electrical and Computer Engineering (ECE)
Technical University of Crete
Chania, Greece
gxrysos@tuc.gr, sioannidis@tuc.gr

Abstract— **Host-based intrusion detection systems (HIDS) need to detect both unseen exploits from day one and recurring known attack families. Maintaining separate anomaly-detection and supervised-classification frameworks increases the computational and operational footprint — a cost that only a few embedded or edge devices can carry. To satisfy both requirements without increasing deployment overhead, we propose a single dual-head Convolutional Neural Network (CNN) operating over a sliding window of embedded system-call tokens. Head 1 is an autoregressive next-token prediction module that emits sequence-level anomaly scores derived from token-wise negative log-likelihood (NLL) estimates, while Head 2 is a supervised attack-classification module using a shared latent representation. The two heads jointly support concurrent anomaly detection and supervised attack classification within a single model.**
The proposed framework is evaluated using the ADFA-LD Linux system-call benchmark [1]. The proposed framework achieves mean AUROC scores of 0.916 in the unsupervised setting and 0.979 in the supervised setting, outperforming the baselines reported in [2] under the reshuffled evaluation protocol in both settings. These results demonstrate that unified sequential representation learning can simultaneously support both zero-day anomaly detection and supervised attack classification refinement within a single operational HIDS framework, thereby reducing the operational complexity of practical HIDS deployments.

Keywords-host-based intrusion detection; anomaly detection; deep learning; convolutional neural networks; zero-day attacks

I. INTRODUCTION

For decades, the growth of network traffic volume has been accompanied by a corresponding increase in cyberattacks, making Intrusion Detection Systems (IDS) the standard mechanism for detecting malicious activity before it reaches downstream applications. Two principal detection frameworks dominate intrusion detection research and deployment. Signature-based detection frameworks match incoming events against curated databases of malicious patterns. While highly effective against previously observed attacks, such systems are inherently limited to known threats, leaving zero-day exploits and modified variants undetected until new signatures are deployed [1]. Anomaly-based detection frameworks address this issue by modelling normal system behaviour from historical benign data and flagging deviations as potential threats, enabling zero-day coverage without prior signatures [3], [4]. Machine Learning (ML) techniques have become the dominant implementation strategy for both paradigms [5], [6].

Practical HIDS operate across two distinct phases of the threat lifecycle. Supervised approaches require representative labelled samples from all target attack classes during training. However, systems are typically deployed under cold-start conditions, where benign traces are abundant but labelled attack samples are scarce and become available only gradually as detected incidents are analysed retrospectively [2], [5]. Anomaly-based methods sidestep this by training on benign data alone, but at the cost of elevated false-positive rates that increase operational overhead [7].

System-call traces are inherently sequential and malicious behaviour often exhibits temporally ordered execution patterns and contextual dependencies between events. Consequently, effective intrusion detection depends not only on statistical syscall composition, but also on preserving temporal context within execution traces. While recurrent architectures have traditionally dominated sequential intrusion detection research, convolutional sequence models provide an efficient alternative due to their parallelizable structure, reduced computational complexity and ability to capture local temporal patterns within syscall sequences. Moreover, convolutional sequence models avoid the sequential training bottlenecks of recurrent architectures and they can offer improved efficiency compared to attention-based models in medium-length syscall contexts.

To address these challenges, we propose a dual-head CNN architecture that combines anomaly-based and supervised intrusion detection, both operating on a shared intermediate trace representation. The model is deployable from day zero using only benign traces and it progressively incorporates labeled attack families as they become available, without requiring architectural redesign or sacrificing zero-day detection capability. The proposed approach is evaluated on the ADFA-LD benchmark, which contains attack traces specifically designed to resist trivial detection [1].

Manuscript received May 25, 2026; revised July 22, 2026; accepted August 21, 2026; published August 25, 2026.
Issue category: Special Issue on DSD/SEAA 2026 on Works in Progress (WiP) Session, Krakow, Poland, Sept. 2026.
Paper category: Full
DOI: doi.org/10.64552/wipiec.v12i2.143

To the best of our knowledge, this is the first HIDS framework explicitly designed to unify cold-start anomaly detection and progressively supervised attack classification within a shared sequential learning architecture. The main contributions of this work are summarized as follows:

- This work introduces *Conv1D-based CNN architecture* for system-call intrusion detection that jointly models anomaly detection and supervised attack classification within a shared latent space, integrating autoregressive next-token prediction with discriminative learning.
- A lifecycle-aligned training strategy is developed to support cold-start deployment using only benign traces, while progressively incorporating labelled attack families during fine-tuning without compromising zero-day detection performance, as measured by AUROC.
- Experimental results on ADFA-LD reveal consistent improvements, achieving superior AUROC and F1 performance vs. strong prior baselines in both unsupervised and supervised settings.
- Last, this work provides empirical evidence that system-call ordering and temporal context are critical factors for effective intrusion detection, highlighting the importance of sequential structure beyond static syscall statistics.

The rest of this paper is organized as follows: Section II reviews related work; Section III describes the methodology; Section IV presents the experimental setup; Section V reports the results; Section VI discusses findings and limitations; and Section VII concludes.

II. Related Work

Related host-based intrusion detection research has explored anomaly detection, supervised classification and hybrid learning approaches, often under different assumptions regarding sequence modeling, supervision and deployment lifecycle adaptation.

A. Benchmarks and Evaluation Protocols

System-call HIDS research relies on a small set of public benchmarks, including UNM, DARPA, ADFA-LD [1], and LID-DS [8], each capturing only a limited subset of benign and malicious behaviour observed in production environments [5], and are often used under incompatible train/test protocols that complicate direct comparison across studies [9]. Several recent supervised approaches report very high scores on the original ADFA-LD split, for example He et al. report F1 = 0.972 and AUROC = 0.992 [10], and Shamim et al. report F1 = 0.990 and AUROC = 0.992 [11]. Nevertheless, Kim et al. show that the original Train and Validation pools are not drawn from the same distribution, so models trained on the released Train split generalize poorly and the original partition is unsuitable for deep-learning evaluation [2]. To address this, Kim et al. propose a reshuffled protocol that merges the released Train and Validation benign-sequence pools before re-partitioning into train/validation/test, making the split suitable for representation learning. More recently, Landauer introduced a reproducible 1%-train / 99 %-test ADFA-LD benchmarking pipeline for consistent comparison across methods [9], while LID-DS extended syscall-based evaluation with richer execution metadata, including thread information, syscall arguments and return values [8].

B. Representation Strategies for System-Call Traces

System-call traces can be represented as discrete, variable-length token sequences, and prior HIDS approaches differ primarily in how these sequences are encoded. Recent deep-learning approaches increasingly rely on dense sequential embeddings; however, most of them continue to separate anomaly detection and supervised classification into independent deployment stages. Embedding-based pipelines map each trace to a fixed vector that retains order. Kim et al. [2] represent ADFA-LD traces using a fixed embedding, i.e., Doc2Vec [12], RNN autoencoders and denoising RNN autoencoders, while anomaly detection is performed by measuring deviation from benign embeddings. Once labeled attacks become available, supervised classifiers are trained on the same latent vectors. Similarly, Gungor et al. [13] combine DeepSVDD-trained CNN embeddings with isolation-based novelty detection on LID-DS [8]. Despite substantial differences in representation strategy and architectural complexity, most prior approaches continue to treat anomaly detection and supervised attack classification as separate operational stages. In many embedding-based and deep sequential pipelines, the anomaly detector is trained exclusively on benign traces during deployment initialization, while supervised classifiers are introduced later as independent models once labeled attack families become available. Consequently, integrating supervised knowledge often replaces the original anomaly detector rather than extending it within a unified lifecycle-aware detection framework.

C. Hybrid and Lifecycle-Aware Intrusion Detection

Recent intrusion detection research has increasingly explored hybrid architectures that combine multiple detection objectives, representation strategies, or decision mechanisms within a shared framework. Li et al. propose a dual-head network with a shared parallel CNN–BiLSTM backbone feeding separate binary attack-detection and multiclass attack-type classification heads that are trained jointly under supervision, with the detection head gating the type prediction at decision time in network intrusion detection [14]. Similarly, Kamal et al. employ a serial Autoencoder–Dense–Transformer pipeline with adaptive resampling and class weighting to improve supervised detection robustness under imbalanced traffic distributions [15]. Consequently, these methods remain unsuitable for cold-start HIDS deployment where labeled attack traces are initially unavailable.

Other works seek to improve robustness against previously unseen attacks by combining discriminative and generative anomaly signals. Cao et al. combine OpenMax-based open-set recognition with variational autoencoder reconstruction scoring to identify samples that deviate from the training distribution [16]. On the host side, SeHIDS [17] adjusts model size as system

behaviour evolves over time, but still trains a separate signature-based and anomaly-based detector for each deployment. Nevertheless, these approaches continue to maintain separate anomaly-based and signature-based detection components rather than integrating both objectives within a unified learning framework. Collectively, prior hybrid and adaptive intrusion detection systems continue to treat anomaly detection and supervised attack classification as distinct operational stages. As labeled attack families become available, supervised components are typically introduced as independent replacement models rather than extensions of the original anomaly detector. Consequently, existing hybrid intrusion detection systems remain only partially aligned with the operational lifecycle requirements of real-world HIDS deployment.

III. Methodology

The proposed framework combines anomaly detection and supervised attack classification over sequential system-call traces within a dual-head architecture, supporting both cold-start and progressively supervised phases of a host-based IDS deployment. Under the practical lifecycle of HIDS, the system must (i) identify previously unseen attacks from benign traces alone, (ii) incorporate labeled attack knowledge to improve precision on recurring families, and (iii) preserve anomaly-based detection capability after supervised fine-tuning.

A. Dataset

Following Landauer [9], the ADFA-LD benchmark [1] was selected for evaluation of the proposed framework. ADFA-LD contains 175 distinct syscall identifiers distributed across 833 benign training traces, 4372 benign validation traces, and 746 attack traces spanning six attack families: Adduser, Hydra-FTP, Hydra-SSH, Java-Meterpreter, Meterpreter and Web-Shell. Among the syscall-based intrusion datasets surveyed by Landauer, ADFA-LD exhibits the highest normalized N-gram entropy and does not expose trivial detection shortcuts based on trace length or previously unseen syscall tokens.

B. System Architecture

The proposed framework consists of a shared sequential convolutional encoder feeding two complementary detection heads operating on a common latent representation z, as illustrated in Fig. 1: an autoregressive next-token decoder producing a Negative Log-Likelihood (NLL) anomaly score, and a supervised binary classifier for known attack families. The two scores are combined through a score-fusion stage into the deployment-time detection score p_{fusion}. The input representation and core components are described below.

1) Input Representation: Each system-call trace is represented as a discrete sequence over a vocabulary of 176 token identifiers comprising the 175 distinct ADFA-LD syscall IDs together with a reserved PAD (padding) symbol at index 0. To convert variable-length execution traces into fixed-size inputs suitable for convolutional processing, a sliding window of length W and stride $S = 1$ is applied across each sequence. For each prediction position t, the input window is defined as

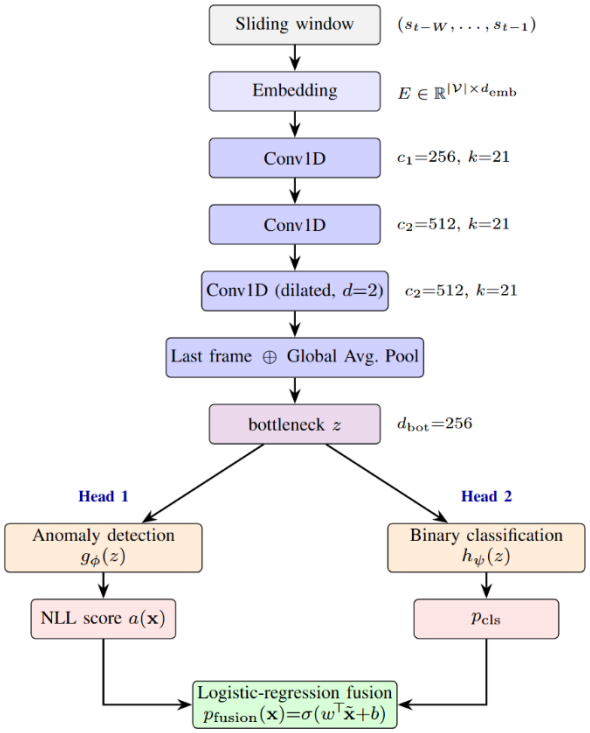


Figure 1. Dual-head CNN over a sliding window of system-call token embeddings. Head 1 (left) is a next-token-prediction decoder providing an NLL anomaly score; Head 2 (right) is a binary classifier on the shared bottleneck. The two scores are combined into a single score p_{fusion} at deployment.

$\boldsymbol{x}_t = (s_{t-W}, \ldots, s_{t-1})$, where $\boldsymbol{x}_t$ is an integer vector of length W containing the syscall tokens immediately preceding the target position. Each token window is mapped through a learned embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d_{emb}}$ producing an embedded representation $\boldsymbol{x}_t$ to $X_t \in \mathbb{R}^{W \times d_{emb}}$. The resulting embedding sequence forms a one-dimensional signal of length W with d_{emb} channels, which is processed by the Conv1D encoder.

2) Shared CNN Encoder: The shared encoder $f_\theta: \mathbb{R}^{W \times d_{emb}} \to \mathbb{R}^{d_{bot}}$ consists of three Conv1D blocks of kernel size k, where the first two blocks use c_1 and c_2 output channels respectively, while a third dilated Conv1D block with dilation $d = 2$ enlarges the temporal receptive field. Each convolutional block is followed by ReLU activation and batch normalization. The final temporal representation is obtained by concatenating the last feature frame with its global-average pooled representation, followed by layer normalization and linear projection into a bottleneck embedding $z \in \mathbb{R}^{d_{bot}}$. Hyperparameters are selected based on the sensitivity analysis described in Sec. IV-C. The final configuration uses $W = 32$, $S = 1$, $c_1 = 256$, $c_2 = 512$, $k = 21$, $d_{bot} = 256$, $d_{emb} = 256$, corresponding to an approximately 10 M-parameter backbone.

3) Head 1 – Autoregressive Anomaly Detection: The anomaly-detection head $g_\phi: \mathbb{R}^{d_{bot}} \to \mathbb{R}^{|\mathcal{V}|}$ performs autoregressive next-token prediction over syscall sequences.

Given an input window $\boldsymbol{x}_t = (s_{t-W}, \ldots, s_{t-1})$ the decoder predicts the subsequent syscall token s_t. Since the prediction target lies outside the observed window, no causal masking is required within the encoder. The anomaly loss is the expected per-window Negative Log-Likelihood [19], $\mathcal{L}_A = \mathbb{E}_{(\boldsymbol{x}_t, s_t)}[NLL(s_t \mid \boldsymbol{x}_t)]$, computed over training windows extracted exclusively from benign traces. During inference, per-window NLL values are aggregated into sequence-level anomaly scores using mean, maximum and p95 statistics per sequence for calibration and evaluation.

4) Head 2 – Supervised Binary Classification: The supervised classifier $h_\psi: \mathbb{R}^{d_{bot}} \to [0,1]$ consists of a single-layer MLP with sigmoid activation operating on the shared bottleneck representation. The supervised loss $\mathcal{L}_B$ is formulated as binary cross-entropy over (window, attack-flag) pairs. To reduce class imbalance during fine-tuning, training batches are sampled using a balanced loader that matches the number of benign and labeled attack windows.

5) Score Fusion: During inference, the sequence-level anomaly score $a(\boldsymbol{x}) = \overline{NLL}$ and the classifier probability $p_{cls} = h_\psi(f_\theta(\boldsymbol{x}))$ are combined through a logistic-regression fusion layer trained on the validation set. The raw fusion input vector $[log(1 + a(\boldsymbol{x})), p_{cls}]$ is first standardised and subsequently processed by an L_2-regularised, class-balanced logistic regression model, yielding the deployment score $p_{fusion}(\boldsymbol{x}) = \sigma(w^\top \tilde{\boldsymbol{x}} + b)$. The anomaly feature is log-compressed to reduce the impact of absolute-scale drift between validation and test partitions. The final decision threshold τ is selected on the validation set by maximising F1-score.

C. Two-Stage Training Strategy

Stage A corresponds to unsupervised cold-start deployment and trains the encoder together with the autoregressive anomaly-detection head exclusively on benign windows for 80 epochs using the Adam optimizer [20] at a peak learning rate of 5×10^{-4}. After each epoch, sequence-level NLL is evaluated on the validation set (benign sequences together with all attack sequences) and the checkpoint achieving the highest validation NLL-AUROC is retained. Including unknown-family attacks in this selection signal slightly inflates the reported zero-day sensitivity, since the held-out families are seen during checkpoint choice; downstream calibration of the fusion layer and decision threshold τ (Sec. III-B5) uses only the known subset, matching the operational lifecycle.

Stage B performs 30 epochs of supervised fine-tuning once labeled attack families become available. The encoder parameters use a backbone learning rate of 1×10^{-5}, ten times lower than the classifier head (1×10^{-4}). The two heads are optimized jointly under $\mathcal{L} = \mathrm{BCE}(h_\psi(z), y) + \lambda_{\mathrm{recon}} \cdot \mathbb{E}_{\mathbf{x} \sim \mathrm{normals}}[\mathrm{NLL}(g_\phi(z), \mathbf{x})]$ with $\lambda_{\mathrm{recon}} = 0.5$. The reconstruction term is computed only on benign windows, so the anomaly head never learns to model attacks. Each batch contains an equal number of benign and labeled-attack windows. The checkpoint with the highest composite score cls-AUROC + 0.5 · NLL-AUROC is kept, subject to a hard floor of NLL-AUROC ≥ 0.75. Stage B is activated only when labeled families exist; otherwise the system runs under Stage A alone.

IV. Experimental Setup

A. Evaluation Metrics

AUROC serves as our primary metric, being threshold-independent and robust to class imbalance [18]. The primary AUROC is computed in pooled binary form, scoring all attack traces against all benign traces, and is the most realistic real-world metric because a deployed detector faces novel zero-day attacks that cannot be attributed to any known family and must simply be flagged as anomalous; the per-attack-family breakdowns across the six ADFA-LD families are also reported for comparability with prior work [2]. F1-score, precision, and recall are additionally reported at a validation-derived threshold τ. Because benign NLL scores drift slightly between validation and test, a threshold tuned on the validation set is not perfectly calibrated on the test set. AUROC, which depends only on score ordering, is unaffected by this shift. Preservation of zero-day detection after finetuning with three out of six attack classes is assessed by comparing unsupervised AUROC before and after fine-tuning.

B. Evaluation Protocols

Because the released Train/Validation partition is unsuitable for deep-learning, as discussed in Sec. II, two complementary protocols are adopted in this work.

1) Landauer 1%-train split: This split of the ADFA-LD dataset was originally intended for classical sequence-based detectors. Under this setup, 1% of benign traces are used for training, while the remaining benign traces together with the full attack pool are reserved for testing [9]. We adopt this protocol with minimalistic training pool to benchmark against classic detectors. While [9] reports only F1 (not AUROC) under this protocol, our corresponding comparison in this work is likewise F1-based (Sec. V-B).

2) Kim-Style Reshuffled Protocol: The primary evaluation protocol follows the reshuffled deep-learning-oriented setup proposed by Kim et al. [2]. The original Train and Validation benign pools are first merged (5205 traces), randomly reshuffled at sequence level, and partitioned into a fixed 70/15/15 train / validation / test split. This produces 3643 benign training traces, 781 validation traces, and 781 test traces, substantially increasing the effective training pool compared to the original 833-trace Train partition while remaining within the upper range of the training sizes swept by Kim et al. Attack traces are distributed proportionally across the same partitions, resulting in 522/112/112 attack sequences for train/validation/test respectively. During the unsupervised anomaly detection stage, attack traces are excluded entirely from training. Labeled attack families are incorporated only during the supervised fine-tuning stage described in Sec. III-C. This reshuffled protocol enables stable deep-learning evaluation, supports lifecycle-aligned supervised refinement, and facilitates direct comparison with prior deep-learning

baselines evaluated under equivalent experimental conditions. The matching deep-learning baselines under this protocol are the RNN denoising autoencoder of [2] combined with Isolation Forest (unsupervised) and a Random Forest (supervised). Numeric values are reported in Sec. IV-D.

TABLE I. ARCHITECTURAL SENSITIVITY ANALYSIS (STAGE A VALIDATION AUROC). EACH ROW PRESENTS A SINGLE ARCHITECTURAL PARAMETER. THE PARAMETER COUNT IS HELD CONSTANT WITHIN THE WINDOW-SIZE SWEEP AND INCREASES MONOTONICALLY ELSEWHERE.

Parameter (range)	Min → Peak → Max	AUROC range
Window W	1 …32 …256	0.734 / 0.863 / 0.838
Kernel k	3 …31	0.825 / 0.858
Channels c_1/c_2	8/16 …256/512	0.787 / 0.881
Bottleneck d_{bot}	4 …512	0.810 / 0.851
Embedding d_{emb}	4 …512	0.822 / 0.856

C. Architectural Sensitivity Analysis and Final Parameters

The architectural configuration adopted throughout this work is described in Sec. III-B2, and it is selected through a one-factor-at-a-time sensitivity analysis over five primary architectural parameters: window size W, kernel size k, convolutional channels c_1/c_2, bottleneck dimension d_{bot}, and embedding dimension d_{emb}. Each parameter is varied independently, while all remaining hyperparameters are held fixed. Model selection is based on unsupervised validation AUROC. Table I summarises the complete sensitivity-analysis results. The sensitivity analysis favours larger model capacity across all architectural dimensions except window size. Unlike the other parameters, W controls how many past syscall tokens are visible to the model per prediction step, so its sweep is directly tied to the sequential structure of ADFA-LD anomalies and is analysed further in Sec. V-A. Convolutional channel capacity produces the largest performance improvement, increasing validation AUROC from 0.787 to 0.881, while kernel performance saturates near $k = 21$ and the bottleneck representation saturates at $d_{\text{bot}} = 256$. In contrast, the embedding dimension has comparatively limited influence on performance, which is consistent with the relatively small syscall vocabulary of ADFA-LD (176 tokens). The final retained configuration (Sec. III-B2) is used consistently across all experiments, together with the two-stage training schedule of Sec. III-C.

D. Comparison Baselines

Baselines are grouped by the evaluation protocols defined in Sec. IV-B, so that head-to-head comparisons in Sec. V involve only methods sharing the same train/test partition. Under the Landauer 1 %-train protocol, the strongest baseline reported by Landauer [9] on ADFA-LD is an edit-distance detector with $F1 = 0.343$, which serves as the reference point for the cold-start comparison. Under the Kim-style reshuffled protocol, the deep-learning baselines of Kim et al. [2] are used: an RNN denoising autoencoder combined with Isolation Forest, reaching a macro-average per-attack AUROC of 0.825 (best 0.883, Java-Meterpreter) in the unsupervised setting, and the same encoder combined with a 100-tree Random Forest, reaching a macro-average per-attack AUROC of 0.967 (best 0.985, Hydra-FTP) under supervised training.

E. Evaluation Scenarios

Four evaluation scenarios are considered to emulate different stages of the operational HIDS lifecycle. **S1** reproduces the Landauer 1 %-train / 99 %-test protocol for direct comparison with classical sequence-based anomaly detectors. **S2** evaluates the unsupervised Stage A under the Kim-style reshuffled protocol with $|\mathcal{K}| = 0$ known attack families, across five random seeds. **S3** extends the same reshuffled protocol to the partially supervised Stage B configuration with $|\mathcal{K}| = 3$ randomly selected known attack families. **S4** corresponds to the fully supervised Stage B configuration with all six attack families available $|\mathcal{K}| = 6$, evaluated across five random seeds. Additionally, we investigate the sequential manifestation of anomalies in two experiments to answer whether ADFA-LD anomalies primarily reflect sequential structure or simple token-frequency artifacts [9]. First, the context window size W is varied while keeping the parameter count fixed, isolating temporal context from model capacity. Second, syscall traces are independently permuted per sequence, preserving token composition and sequence length while destroying temporal order and local n-gram structure. Both analyses are presented in Sec. V-A.

V. RESULTS

A. Sequential Manifestation of Anomalies

This section summarizes the experiments designed to evaluate the role of temporal context and syscall ordering in ADFA-LD anomaly detection.

1) Window-Size Sweep at Constant Parameter Count: The first experiment evaluates the contribution of sequential context by varying the window size W while maintaining a constant parameter count. Convolutional channel dimensions are adjusted accordingly so that all evaluated configurations contain exactly $325{,}425$ trainable parameters. As shown in Fig. 2, validation AUROC increases from 0.734 at $W = 1$ to 0.863 at $W = 32$, before gradually decreasing to 0.838 at $W = 256$. As the model capacity remains constant throughout the sweep, the observed improvement can be attributed to the availability of sequential temporal context rather than increased model expressiveness. These results provide initial evidence that anomaly detection on the ADFA-LD dataset depends on sequential structure rather than solely on marginal token statistics. The gradual degradation beyond $W = 32$ reflects signal dilution, where anomalous behavioural motifs occupy relatively short temporal spans and the larger windows introduce increasing amounts of benign context.

2) Per-Sequence Token Permutation: While the window-size sweep demonstrates that larger temporal context improves detection, it does not directly establish whether syscall ordering itself is necessary. The second probing experiment changes sequential structure by independently permuting syscall order within each trace while preserving the sequence length, the token composition, and the class labels. Consequently, only

temporal dependencies and local n-gram structure are removed, whereas all compositional statistics remain unchanged. The complete anomaly-detection pipeline is then retrained on the permuted dataset using identical architecture, hyperparameters,

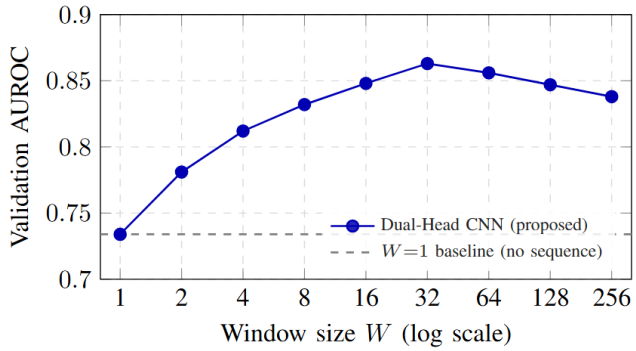


Figure 2. Stage A validation AUROC versus window size W with constant parameter count ($n_{params} = 325{,}425$). AUROC peaks at $W = 32$; the rise from $W = 1$ (unigram) is therefore attributable to sequential context rather than to additional model capacity.

TABLE II. PER-SEQUENCE SHUFFLE (STAGE A). TOKEN MULTISETS AND LENGTHS ARE UNCHANGED; ONLY POSITIONAL STRUCTURE IS REMOVED. AUROC DROPS SHARPLY ACROSS ALL ATTACK FAMILIES.

Metric	Ordered	Scrambled
Stage A best val AUROC	0.893	0.405
Test AUROC (fusion)	0.869	0.583
Final next-token cross-entropy	0.089	1.599

splits and random seeds. The performance degradation is reported in Table II: Stage A validation AUROC drops from 0.893 on the original ordered sequences to 0.405 on the scrambled sequences, while test AUROC decreases from 0.869 to 0.583. Similarly, the final next-token cross-entropy increases substantially from 0.089 to 1.599, indicating that little predictable structure remains once temporal order is removed. The normal/attack NLL separation further supports this interpretation. Under ordered traces, attacks produce higher mean NLL than benign traces (3.28 vs. 0.75), whereas under shuffled traces the separation reverses (2.44 vs. 2.70). This inversion suggests that non-sequential compositional statistics alone are insufficient for reliable anomaly discrimination and may actively mislead the detector once temporal structure is removed.

Combining the window-size sweep and the per-sequence permutation experiment results provide direct evidence that ADFA-LD anomalies manifest primarily through sequential execution patterns rather than token-frequency statistics. All remaining experiments use the same retained architectural configuration without further modification.

B. *Landauer 1% Train Split – Classical-Baseline Comparison*

Under the Landauer 1 %-train / 99 %-test protocol, the proposed framework is evaluated in Stage A mode using the NLL anomaly score without supervised fine-tuning. As summarized in Table III, the detector reaches $F1 = 0.403$ ($P = 0.379$, $R = 0.431$), improving over the strongest classical baseline of Landauer [9] (edit distance, $F1 = 0.343$) by approximately 6 F1 points under the same constrained 1 %

TABLE III. LANDAUER 1 %-TRAIN / 99 %-TEST UNSUPERVISED ADFA-LD PROTOCOL.[A]

Method	F1	P	R
Ours, $\lvert\mathcal{K}\rvert = 0$	**0.403**	0.379	0.431
Landauer 2024, edit distance	0.343	–	–

[A]Trained on only 1 % of benign traces; not directly comparable to the reshuffled-split tables. Landauer 2024 reports only F1 under this protocol.

TABLE IV. UNSUPERVISED STAGE A ($|\mathcal{K}| = 0$) ON THE [2] RESHUFFLED 70/15/15 SPLIT. ALL VALUES ARE AUROC. POOLED IS TAKEN OVER ALL ATTACKS VS. ALL NORMALS; AVERAGE AND BEST ARE COMPUTED OVER THE SIX ADFA-LD ATTACK FAMILIES.

Method	Pooled	Average	Best
Ours, $\lvert\mathcal{K}\rvert = 0$	0.916	**0.913**	**0.942**
Kim 2021, RNN-DAE + IF	–	0.825	0.883

training regime. This indicates that the learned sequential representation captures substantially richer behavioural structure than lightweight statistical sequence matching alone.

C. *Reshuffled Split – Stage A (Unsupervised)*

Under [2] reshuffled protocol, Stage A is evaluated across five independent random seeds. Since the Stage A training setup is based on fully unsupervised learning, the number of known attack families $|\mathcal{K}|$ has no effect on the training stage. The pooled binary test AUROC across the five runs is 0.916 ± 0.040; the corresponding mean $F1 = 0.636 \pm 0.125$ ($P = 0.531 \pm 0.166$, $R = 0.827 \pm 0.031$) at the validation-derived threshold. Kim et al. [2] do not report a pooled binary AUROC, so direct comparison on this metric is not possible. On the per-attack axes Kim does report, our framework reaches a macro-average per-attack AUROC of 0.913 and a best per-attack AUROC of 0.942 (Hydra-FTP), exceeding the corresponding [2] values of 0.825 average / 0.883 best on every individual attack family (Table IV).

D. *Reshuffled Split – Stage B Lifecycle Evaluation*

The Stage B fine-tuning phase progressively incorporates labelled attack families through the supervised classifier while preserving the anomaly-detection capability of the shared encoder, as described in Sec. III-C. Two deployment schemes are reported, emulating the lifecycle by progressively expanding the set of known attack families: $|\mathcal{K}| = 3$, where only three of the six ADFA-LD attack families are known and the remaining three families are zero-day at test time, and $|\mathcal{K}| = 6$, corresponding to a fully supervised deployment scenario. For the $|\mathcal{K}| = 3$ setting, each of five runs randomly selects three of the six known attack families, while the other three remain zero-day at test time. Under this configuration, the primary fusion score achieves mean $F1 = 0.654 \pm 0.097$ and mean pooled binary $AUROC = 0.940 \pm 0.024$. As $|\mathcal{K}| = 3$ is partially supervised, it is not directly comparable to either of [2] fully

unsupervised or fully supervised baselines. Crucially, supervised fine-tuning under $|\mathcal{K}| = 3$ does not erode the unsupervised anomaly path. To isolate this, the Stage B model's NLL score is re-evaluated on the test partition restricted to

TABLE V. SUPERVISED STAGE B AUROC ON THE [2] RESHUFFLED 70/15/15 SPLIT. POOLED IS TAKEN OVER ALL ATTACKS VS. ALL NORMALS; MACRO AND BEST ARE COMPUTED OVER THE SIX ADFA-LD ATTACK FAMILIES.

Method	Pooled	Macro	Best
Ours, $\|\mathcal{K}\| = 3$	0.940	–	–
Ours, $\|\mathcal{K}\| = 6$	**0.979**	**0.977**	**0.989**
Kim 2021, RNN-DAE + RF	–	0.967	0.985

benign sequences and the three attack families that were excluded from fine-tuning (the held-out zero-day families), and the resulting NLL-AUROC is compared against the Stage A model evaluated on the exact same subset. Averaged across runs, this Stage A → Stage B change is $\Delta = +0.001$, and every individual run stays within $[-0.001, +0.003]$ of its cold-start value. The safeguards described in Sec. III-C – recon-on-normals, reduced backbone learning rate, and the composite checkpoint criterion with NLL-AUROC floor – successfully prevent classifier-driven encoder drift. This shows that the dual-head design delivers *both* capabilities at once: the supervised F1/AUROC boost on the known families is gained *without* sacrificing the cold-start NLL detector on the unseen ones. This simultaneous-capability property directly satisfies requirement (iii) of Sec. III and is, to the authors' knowledge, the central novelty of the proposed architecture for the IDS lifecycle. The fully supervised $|\mathcal{K}| = 6$ configuration evaluated across five seeds achieves mean F1 $= 0.815 \pm 0.056$ and mean pooled binary AUROC $= 0.979 \pm 0.005$. Although [2] does not report a pooled binary AUROC, our $|\mathcal{K}| = 6$ model achieves a macro-averaged per-attack AUROC of 0.977, with a peak AUROC of 0.989 on Hydra-FTP. These results exceed the supervised RNN-DAE + Random Forest baseline reported in [2], which attains an average AUROC of 0.967 and a best AUROC of 0.985 across the individual attack families evaluated.

Table V reports the supervised lifecycle results, simulating the operational state of a HIDS after a substantial period of analysis. The $|\mathcal{K}| = 3$ row emulates a mid-lifecycle stage in which three families are still effectively zero-day, while $|\mathcal{K}| = 6$ corresponds to all currently catalogued ADFA-LD families being known. Fig. 3 visualizes the monotonic improvement in detection performance as more attack families become known.

E. Calibration and Distribution Drift

The training logs reveal a moderate distribution shift between the validation and test benign NLL distributions. The validation partition exhibits mean NLL $= 0.200$ (p95 $= 1.137$), while the test partition reaches mean $= 0.242$ (p95 $= 1.375$), corresponding to an approximately $1.21\times$ increase across both statistics. Although the observed drift is sufficiently small to preserve ranking-based evaluation through AUROC, it is large enough to affect fixed-threshold calibration. Consequently, validation-derived thresholds are used when reporting operational metrics such as F1-score, precision, and recall, rather than relying on absolute anomaly-score cut-offs.

VI. DISCUSSION

Head-to-head comparisons are limited to baselines evaluated under identical experimental protocols (Sec. III). The reshuffle

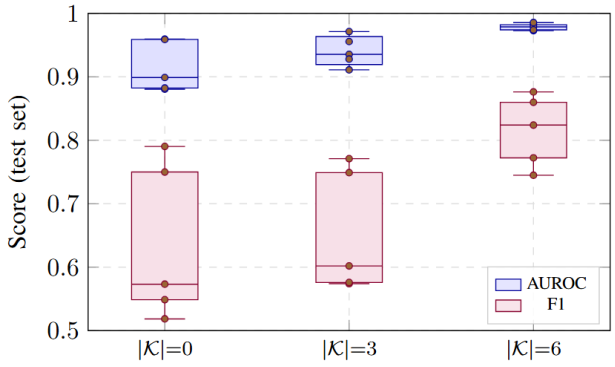


Figure 3. Lifecycle progression – the mean of both test AUROC (blue) and F1 (purple) increases monotonically as more attack families become known. Each scenario shows the distribution across $n = 5$ random seeds; for $|\mathcal{K}| = 3$, each run uses a random three-of-six subset of labelled families.

protocol of [2] seems to be the most appropriate setting for deep models since the original ADFA-LD Train and Validation normal sets are not drawn from the same distribution [2], so training and validating under the original split creates inconsistencies, making it difficult to tell whether poor scores reflect the model limitations or the partition effects.

Our dual-head architecture aligns with the IDS lifecycle: Stage A provides strong unsupervised detection under the matching protocol (Sec. V-C), and each labelled family incorporated in Stage B improves discrimination progressively (Table V). More importantly, the $|\mathcal{K}| = 3$ scenario shows that the supervised gain on known families is achieved *without* degrading the zero-day NLL detector on unseen families ($\Delta \approx 0$, Sec. V-D). This is supported by the reconstruction term in the Stage B loss, which helps preserve benign pattern representations and mitigate forgetting (Sec. III-C). To the best of our knowledge, this is the first system-call HIDS to demonstrate that a single shared encoder can carry both objectives simultaneously across the deployment lifecycle.

The ≈10 M trainable parameters of the encoder represent 99.5% of the total model parameters while the two detection heads + fusion layer, only add under 0.5% of the parameters. For identically sized encoders, two separate networks would nearly double both parameters and computational cost, so sharing the encoder roughly halves the deployment footprint. This efficiency could be optimized further on embedded and edge hosts.

Limitations and outlook: ADFA-LD provides a controlled and widely used benchmark for system-call intrusion detection, enabling reproducible lifecycle evaluation. We report results over five different seeds and random three-of-six subsets with stratified sampling, and use AUROC as the primary metric.

While this setting ensures comparability and stability of evaluation, it naturally serves as a stepping stone toward deployment-oriented regimes where decisions require explicit thresholding, e.g., via conformal or quantile-based calibration under sliding windows.

The proposed framework is lightweight and dataset-agnostic, relying only on the system-call alphabet and window size W, which enables straightforward transfer to benchmarks such as LID-DS [8]. This motivates future extension and evaluation to more complex and realistic operational settings, including multimodal intrusion detection where system calls are integrated with network and process-level signals within a unified representation space.

Finally, the dual-stage formulation highlights a further research direction in representation learning. While Stage B already leverages supervised signals for known attack families, a natural extension is to introduce explicit margin-aware objectives in latent space (e.g., contrastive or triplet learning over z). This extension would further shape inter-class geometry while preserving the reconstruction anchor, and is expected to improve both supervised discrimination and zero-day robustness within a unified framework.

VII. Conclusion

This work introduces a generic dual-head CNN framework for host-based intrusion detection that unifies the cold-start and progressively supervised phases of the HIDS lifecycle within a single architecture. The proposed solution provides zero-day detection from benign traces while seamlessly incorporating labelled attack families as they become available, without requiring any architectural modifications. Under the reshuffled protocol of Kim et al. [2], the framework achieves strong performance in the unsupervised cold-start setting (Stage A AUROC 0.916) and improves consistently as supervision is introduced, reaching an AUROC of 0.979. Across all settings, it outperforms strong RNN-DAE-based baselines under identical evaluation conditions, while preserving zero-day detection capability for unseen attack families. Beyond benchmark performance, this work demonstrates that anomaly-based and supervised intrusion detection can be unified within a single sequential representation, rather than treated as separate or sequentially replaced paradigms. A shared encoder is sufficient to support both cold-start zero-day detection and incremental supervised refinement within one model. Future directions include evaluation on more recent and complex benchmarks such as LID-DS [8], extension to multimodal fusion of system-call, process, and network event streams within a unified latent space, and adaptive threshold calibration mechanisms to improve robustness under operational drift in deployed systems.

References

[1] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), 2013, pp. 4487–4492.

[2] C. Kim, M. Jang, S. Seo, K. Park, and P. Kang, "Intrusion detection based on sequential information preserving log embedding methods and anomaly detection algorithms," IEEE Access, vol. 9, pp. 58 088–58 101, 2021.

[3] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in Proc. IEEE Symp. Security and Privacy, 1996, pp. 120–128.

[4] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in Proc. IEEE Symp. Security and Privacy, 1999, pp. 133–145.

[5] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," ACM Computing Surveys, vol. 52, no. 6, pp. 128:1–128:35, 2019.

[6] S. A. Abdulkareem, C. H. Foh, M. Shojafar, F. Carrez, and K. Moessner, "Network intrusion detection: An IoT and non IoT-related survey," IEEE Access, vol. 12, pp. 144 608–144 636, 2024.

[7] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," ACM Comput. Surv., vol. 48, no. 1, Sep. 2015. [Online]. Available: https://doi.org/10.1145/2808691

[8] M. Grimmer, T. Kaelble, F. Nirsberger, E. Schulze, T. Rucks, J. Hoffmann, and E. Rahm, "Lid-ds 2021," in CRITIS Conference Proceedings, 2021.

[9] M. Landauer, F. Skopik, and M. Wurzenberger, "A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques," Proc. ACM Softw. Eng., vol. 1, no. FSE, p. Article 61, 2024.

[10] J. He, C. Tang, W. Li, T. Li, L. Chen, and X. Lan, "BR-HIDF: An anti-sparsity and effective host intrusion detection framework based on multi-granularity feature extraction," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 485–499, 2024.

[11] N. Shamim, M. Asim, A. I. Awad, and M. K. Khan, "Anomaly detection in Internet of Things system calls using a centroid-based vector-space model," IEEE Internet of Things Journal, vol. 12, no. 14, pp. 26 868–26 881, 2025.

[12] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in Proc. Int. Conf. Machine Learning (ICML), 2014, pp. 1188–1196.

[13] O. Gungor, I. Kale, J. Zhou, and T. Rosing, "Light-hids: A lightweight and effective machine learning-based framework for robust host intrusion detection," 2025. [Online]. Available: https://arxiv.org/abs/2509.13464

[14] T. Li, X. Zhang, H. Zhao, J. Xu, Y. Chang, and S. Yang, "A dual-head output network attack detection and classification approach for multi-energy systems," Frontiers in Energy Research, vol. 12, p. 1367199, 2024.

[15] H. Kamal and M. Mashaly, "Ae-dtnn: Autoencoder–dense–transformer neural network model for efficient anomaly-based intrusion detection systems," Machine Learning and Knowledge Extraction, vol. 7, no. 3, p. 78, 2025.

[16] Z. Cao, Z. Zhao, W. Shang, and S. Ai, "VAEMax: Open-set intrusion detection based on OpenMax and variational autoencoder," arXiv preprint arXiv:2403.04193, 2024.

[17] M. Baz, "SEHIDS: Self evolving host-based intrusion detection system for IoT networks," Sensors, vol. 22, no. 17, p. 6505, 2022.

[18] T. Fawcett, "An introduction to roc analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006, rOC Analysis in Pattern Recognition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X

[19] I. Ring, John H., C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," Digital Threats: Research and Practice, vol. 2, no. 4, pp. 26:1–26:29, Oct. 2021.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learning Representations (ICLR), 2015.