SeqLog: Surface-Invariant Detection and Event Localization in System Logs

Murilo Salem[1], **Daniel Pontes**[1], **Luísa Böhm**[1], **Lúcio Vagner Carvalho**[1], **Theo Ferrugem**[1], **Anderson Ferrugem**[1]

[1] Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico (UFPel)
Pelotas – RS – Brasil

{mcsalem, dhspbarretos, lcbohm, lvcarvalho, tdferrugem, ferrugem}@inf.ufpel.edu.br

Abstract. *Log-based attack detectors are brittle to surface-token variation (commands, hosts, paths) that preserves operational meaning, flag windows without localizing the responsible events, and lose cross-domain F1 under source-selected decision thresholds. We address these failures in a sequence-classification pipeline (SeqLog) with three components. First,* Surface-Invariance Consistency Training *(SICT) pairs each log window with a semantics-preserving lexical rewrite and penalizes prediction divergence via a symmetric KL term. Second, an* attention-MIL pooling head *matches the weak-label structure of log windows ("positive if any event is malicious") and yields event-level localization from window labels alone. Third,* label-free threshold transport *addresses cross-domain operating-point mismatch by transporting the source predicted-positive rate to a quantile of the target score distribution, using zero target labels. On a controlled two-domain benchmark, the model reaches F1* $0.979/0.976$ *on unseen signatures, localization AUC* 0.966 *in both domains, and recovers cross-domain F1 from* 0.259 *to* 0.865 *on* WINDOWS $\rightarrow$ LINUX *(oracle* 0.872*). On public corpora (*HDFS*,* BGL*,* AIT-LDS*), the pipeline stays competitive and reproduces the localization and transport patterns where their preconditions hold.*

1. Introduction

System logs are key evidence for detecting cyber attacks, yet many log-based detectors remain coupled to the lexical forms seen during training (IPs, hostnames, commands, paths) rather than the operational behavior those tokens express. This gives adversaries an inexpensive evasion surface: swapping tooling or identifiers preserves the attack procedure, but rule-based and lexical models treat `wget http://1.2.3.4/payload` and `curl -O http://5.6.7.8/stage1` as unrelated observations.

Pretrained transformers can model contextual relationships across sequences of log events [Guo et al. 2021, Le and Zhang 2021], yet high in-domain accuracy leaves three limitations unresolved: (i) robustness to semantics-preserving lexical variation is an incidental benefit of pretraining rather than an explicit objective; (ii) detection is formulated at the window level whereas analysts need the responsible events, and localization is rarely quantified beyond qualitative attention maps; (iii) cross-domain F1 at a source-selected threshold conflates ranking degradation with operating-point mismatch, since high area under the receiver operating characteristic curve (ROC-AUC) and average precision (AP) with poor target F1 indicates an unsuitable threshold rather than probabilistic miscalibration.

SeqLog, a reproducible normalization–templating–windowing pipeline, separates robust log detection into three complementary problems, one per limitation:

1. **Surface-Invariance Consistency Training (SICT).** Each window is paired with a semantics-preserving rewrite and a symmetric KL-divergence term enforces consistent predictions across views (Section 3.2); disjoint training and evaluation pools rule out substitution memorization.
2. **Attention-based multiple-instance learning (MIL) for event localization.** Treating a window as a bag of events labeled positive if any event is malicious, a gated attention-MIL head [Ilse et al. 2018] yields event-level scores from window labels alone, evaluated with event-level AUC and hit@1 against ground truth withheld from training (Section 3.3).
3. **Label-free threshold transport.** When ROC-AUC and AP stay high despite poor target F1, we set the target threshold at the quantile matching the source predicted-positive rate over unlabeled target scores, adapting the operating point without recalibrating probabilities (Section 3.4).

A leak-resistant evaluation protocol combines unseen-signature holdout, disjoint-pool surface mutation, hard negatives, and bidirectional cross-domain transfer. The synthetic benchmark isolates these failure modes while public corpora probe portability and limits; generalization stress, not saturated in-domain accuracy, is therefore our primary evidence.

2. Related Work

Log-based detection has long relied on parsers such as Drain [He et al. 2017] and the LogPai toolkit [Zhu et al. 2019], which recover event templates that mitigate surface variability; term frequency–inverse document frequency (TF-IDF) and n-gram models over those templates remain strong, if lexically coupled, baselines. The underlying datasets themselves were consolidated by Loghub [Zhu et al. 2023], whose labeled HDFS and BGL corpora we reuse for external validation. A second line adds recurrence or self-attention over token and template sequences: DeepLog [Du et al. 2017], LogAnomaly [Meng et al. 2019], LogBERT [Guo et al. 2021], and NeuralLog [Le and Zhang 2021]. These models optimize in-distribution accuracy and, at best, evaluate robustness to lexical variation post hoc. Closest to our motivation, LogRobust [Zhang et al. 2019] showed that unstable and previously unseen log events degrade template-count detectors and mitigated this with semantic embeddings, and SwissLog [Li et al. 2022] couples robust parsing with attention-based detection and localization for interleaved logs. Both anticipate parts of our problem setting; SeqLog differs in making invariance an explicit training objective with disjoint rewrite pools, and in deriving event-level localization from weak window labels under a quantitative protocol. More recently, large language models have been adapted to log anomaly detection, as in LogLLM [Guan et al. 2024], with surveys such as LLM4Log [Ma et al. 2026] cataloguing rapid progress. These efforts largely report window- or sequence-level accuracy; our contributions (invariance objective, weak-label localization metric, operating-point transport) are orthogonal to the choice of backbone and compose with stronger encoders.

We make invariance explicit through consistency regularization, where penalizing prediction divergence between two views of an input is well established

in semi-supervised vision and natural language processing (NLP) [Xie et al. 2020, Liang et al. 2021]. SICT instantiates this idea with domain-specific, semantics-preserving log rewrites, a setting in which, to our knowledge, consistency training has not previously been applied.

The shift from window-level scoring to interpretable structure motivates our use of attention-based MIL pooling [Ilse et al. 2018], which learns instance weights from bag labels and naturally supports log-window classification and event-level localization. Unlike prior log models that classify flattened windows, we use MIL both architecturally and as a quantitative localization mechanism. Cross-domain robustness also depends on the decision rule: while parameter-efficient methods such as LoRA [Hu et al. 2022] reduce representation-transfer costs, unsupervised thresholding and prior correction address decision and distribution shifts [Otsu 1979, Saerens et al. 2002]. We therefore distinguish threshold adaptation from probabilistic calibration and evaluate the latter separately using Brier score and expected calibration error (ECE).

3. Method

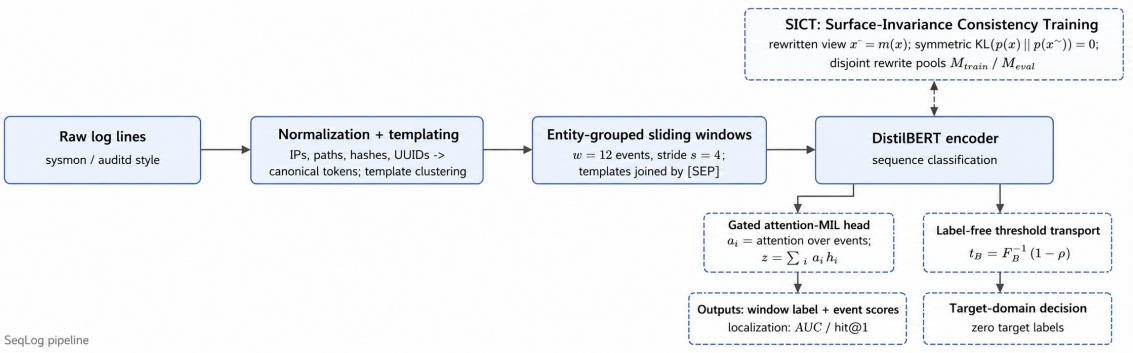


Figure 1. SeqLog pipeline. Logs are normalized, clustered into templates, and grouped into entity-disjoint windows. A DistilBERT encoder is trained with SICT consistency on rewritten views; a gated attention-MIL head yields window scores and event-level localization weights; label-free threshold transport maps the source operating point to the target score distribution.

3.1. Pipeline and Notation

Raw log lines are first normalized: IP addresses, paths, hashes, universally unique identifiers (UUIDs), and numbers are mapped to canonical tokens. Lines are then clustered into templates and grouped per entity (host) into sliding windows of w events with stride s. A window $x = (e_1, \ldots, e_w)$ inherits its label from its events, $y = \mathbf{1}[\exists i : y_i = 1]$, so a window is malicious whenever any event in it is. Only y is available for training; the event labels y_i are held out for evaluation, and all splits are entity-disjoint. The serialized window, its templates joined by `[SEP]`, feeds either the classical baselines (TF-IDF with logistic regression, an ordered template n-gram model, and a rule heuristic) or the transformer classifier, optionally adapted with LoRA. Figure 1 summarizes the pipeline.

3.2. Surface-Invariance Consistency Training (SICT)

The goal of SICT is to make a window and a lexically rewritten but operationally identical version of it receive the same prediction. Let M be a pool of semantics-preserving

rewrites, for instance `wget`→`curl -O`, `schtasks`→`at.exe`, or identifier renames, chosen so that they survive template normalization and therefore actually reach the model. For each training window x we sample an augmented view $\tilde{x} = m(x)$ with $m \sim M_{\text{train}}$, and minimize

$$\mathcal{L} = \tfrac{1}{2}\big[\mathrm{CE}(p_\theta(x), y) + \mathrm{CE}(p_\theta(\tilde{x}), y)\big] + \lambda\, \tfrac{1}{2}\big[\mathrm{KL}(p_\theta(x) \,\|\, p_\theta(\tilde{x})) + \mathrm{KL}(p_\theta(\tilde{x}) \,\|\, p_\theta(x))\big], \tag{1}$$

where p_θ is the model's predicted probability parameterized by θ, CE denotes the cross-entropy loss, KL the Kullback–Leibler divergence, y the window label, and λ the weight of the consistency term. The loss combines a supervised term on both views with a symmetric KL term that penalizes any divergence between their predictions. The augmentation is resampled every epoch. Crucially, the rewrite pool is split into two disjoint halves: Equation (1) draws only from M_{train}, while the robustness evaluation of Section 5.2 applies only M_{eval}. Gains under mutation therefore cannot be attributed to memorizing the substitutions seen at test time.

3.3. Attention-MIL Pooling with Event Localization

The window label is precisely the standard multiple-instance assumption, which lets us treat each window as a bag of event instances. The encoder runs once over the serialized window, and token embeddings are mean-pooled per event (events being `[SEP]`-delimited) to yield $h_1, \ldots, h_w$. Gated attention [Ilse et al. 2018] then scores each event and aggregates them into a single bag embedding:

$$a_i = \frac{\exp\{w^\top(\tanh(Vh_i) \odot \sigma(Uh_i))\}}{\sum_j \exp\{w^\top(\tanh(Vh_j) \odot \sigma(Uh_j))\}}, \qquad z = \sum_i a_i h_i, \tag{2}$$

where V and U are learned projection matrices, w is a learned scoring vector, σ is the sigmoid function, and $\odot$ denotes the element-wise product. A linear head classifies the bag embedding z. The attention weights a_i serve as per-event localization scores, obtained without any event-level supervision. Because our benchmark retains ground-truth event labels, we can evaluate these scores quantitatively rather than inspect them by eye: localization AUC (score versus event label, pooled over windows that contain both classes after per-window min-max normalization) and hit@1 (whether the top-scored event is in fact malicious). As a baseline, we extract the same per-event scores from the `[CLS]` attention of the vanilla transformer. Finally, SICT composes naturally with the MIL head: Equation (1) is simply applied to the bag logits.

3.4. Label-Free Threshold Transport

Let a model be trained on domain A and evaluated on domain B using a threshold t_A selected on source-domain data. Target-domain F1 may collapse even when ROC-AUC and AP remain high. This combination indicates that the model retains useful ranking information but that the source-selected operating point is unsuitable for the target score distribution. It does not, on its own, demonstrate probabilistic miscalibration, which must instead be assessed using metrics such as Brier score and ECE.

We therefore adapt the decision rule rather than the predicted probabilities. Let ρ be the predicted-positive rate obtained by applying t_A to a held-out source validation set

V_A. We set the target threshold to the empirical $(1-\rho)$-quantile of the unlabeled target score distribution,

$$t_B = \hat{F}_B^{-1}(1-\rho), \qquad \rho = \frac{1}{|V_A|} \sum_{x \in V_A} \mathbf{1}[p_\theta(x) \geq t_A], \tag{3}$$

where $\hat{F}_B$ is the empirical cumulative distribution of target scores and $\mathbf{1}[\cdot]$ is the indicator function. Equation (3) changes only the operating point: it neither transforms the model scores nor guarantees calibrated target-domain probabilities. Its main assumption is that the source predicted-positive rate is a useful approximation to the desired target operating rate, which is most plausible when source and target positive prevalence are comparable. We additionally evaluate two prevalence-free alternatives that partition the target score distribution without labels, namely Otsu's criterion [Otsu 1979] and a two-component Gaussian mixture model (GMM) over the logits, and compare them with n-shot labeled threshold selection and a fully labeled oracle.

4. Experimental Setup

Our primary testbed is a controlled synthetic generator with lexically disjoint LINUX and WINDOWS domains, where all hosts emit benign events and half also contain a contiguous 4–10-event MITRE-like attack chain. This control enables signature holdout, mutation, hard-negative, and event-level localization protocols, while making clear that absolute in-domain scores reflect synthetic separability rather than real-world difficulty; accordingly, our conclusions emphasize generalization performance. Generator constants were fixed before evaluation: attack chains contain 4–10 events, positive-window prevalence is about 12% (156 of 1,296 pooled test windows), and cross-domain transfer cannot exploit token overlap. All shared design choices, including window size, stride, backbone, LoRA rank, and consistency weight λ (Table 3), were held fixed across variants to isolate the contribution of each component.

Scaled to 160 hosts per domain, the corpus yields 1,296 test windows, 156 of them positive, pooled over seeds. To probe external validity (Section 5.5), we add three public corpora (Table 1), all loaded into the same JSONL schema under the same normalization and windowing.

Table 1. Corpora used in our experiments. The synthetic generator drives the main generalization claims; the public corpora test portability and scope.

Corpus	Label granularity	Source	Role
Synthetic (LINUX/WINDOWS)	per-event	generated	main protocols
HDFS	per-block (failures)	real	portability
BGL	per-line (alerts)	real	portability
AIT-LDS v2	per-line (MITRE phases)	real intrusions	external validity

On this data we evaluate four leak-free generalization protocols (Table 2), each isolating one failure mode while holding the others fixed. The unseen-signature holdout splits attack templates 2:1, so that training sees only the first subset while the test set contains only the held-out third. Surface mutation trains on clean logs and rewrites the test

set with M_{eval}, the half of the rewrite pool disjoint from the one SICT trains on. Hard negatives are benign events that nonetheless carry attack tokens, namely legitimate uses of `curl`, `powershell`, or `schtasks`. The cross-domain protocol trains on one domain and evaluates on the other with no refitting. Each protocol runs with three seeds, and unless stated otherwise tables report means over these seeds together with seed-level standard deviations where informative. Because three seeds do not support meaningful hypothesis tests, we quantify evidence by the magnitude of inter-variant gaps relative to observed seed variability rather than by p-values. We report F1, ROC-AUC, AP, calibration (Brier and ECE), and, where event labels exist, localization AUC and hit@1.

Table 2. Generalization protocols. Each isolates one failure mode while holding the others fixed.

Protocol	Train sees	Test contains
Unseen-signature	first 2/3 of attack templates	held-out 1/3
Surface mutation	clean logs	rewrites from M_{eval}
Hard negatives	standard mix	benign events with attack tokens
Cross-domain	one domain	the other, no refitting

All variants use a DistilBERT backbone from the transformers library with the hyperparameters of Table 3, and every experiment, its aggregation across seeds, and every asset in this paper are regenerated end to end by scripts in the released repository.

Table 3. Training and windowing hyperparameters.

Setting	Value
Backbone	DistilBERT
Window size w / stride s	12 / 4
Max sequence length	256
Epochs / batch size	4 / 16
Learning rate (full / LoRA)	3×10^{-5} / 5×10^{-4}
LoRA rank	8
SICT weight λ	1
Seeds	3

5. Results

5.1. Unseen Attack Signatures

Our central comparison is the detection of attack signatures absent from training, the cleanest test of whether a model has learned structure beyond memorized templates. The contrast across model families is stark. The rule heuristic drops to F1 $= 0.000$ (pure memorization), the sequential template baseline to roughly 0.05, TF-IDF generalizes only partially (0.56–0.88), and the vanilla transformer retains 0.82–0.92 with unseen AP of at least 0.98 (Figure 2). Table 4 ablates our components on this protocol over three seeds. On Linux the progression is monotone, vanilla 0.818 $\rightarrow$ SICT 0.951 $\rightarrow$ MIL 0.954 $\rightarrow$ MIL+SICT 0.979, while on Windows MIL+SICT reaches 0.976 (unseen AP 1.000),

where each component alone yields only 0.88–0.95. Trained invariance and per-event pooling are thus complementary: the combined variant is the best detector of never-seen signatures in both domains and also the most stable, with seed variance of just ± 0.02. This is the paper's strongest detection result and the main basis for our claim of generalization beyond signature matching.

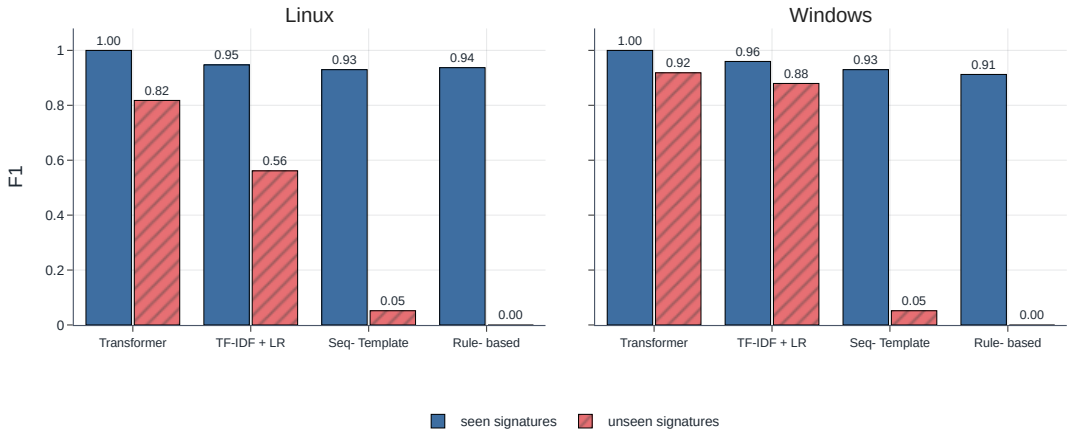


Figure 2. Detection on seen versus unseen attack signatures. The synthetic benchmark is split so that test windows contain attack templates never observed during training. The transformer retains the highest F1 on unseen signatures in both domains, while rule- and template-based baselines collapse, showing that the main gain comes from generalization beyond memorized signatures rather than from in-domain saturation.

Table 4. Component ablation (3 seeds): F1 on unseen signatures and on the surface-mutated test, per variant.

Domain	Variant	F1 (unseen sig.)	AP (unseen sig.)	F1 (mutated)
Linux	Transformer (vanilla)	0.818	0.985	0.958
	+SICT	0.951	0.995	0.961
	+MIL	0.954	0.998	0.980
	+MIL+SICT	0.979	0.999	0.980
Windows	Transformer (vanilla)	0.918	0.981	1.000
	+SICT	0.951	0.992	0.980
	+MIL	0.884	0.989	0.958
	+MIL+SICT	0.976	1.000	0.958

5.2. Surface Mutation and Hard Negatives

Under the evaluation-pool rewrites, every transformer variant holds its clean F1 (ΔF1 $\leq$ 0.02; Table 4, last column), whereas the heuristic loses up to 0.44. With SICT this invariance is trained rather than incidental, and because the train and evaluation pools are disjoint, it provably extends beyond the rewrites seen during training. Against hard negatives, benign events that share attack tokens, such as legitimate `curl` or `powershell`

use, the token-matching heuristic collapses (F1 0.33–0.37), while TF-IDF and the transformer both keep precision near 1.0, suppressing exactly the failure mode that drives alert fatigue. Relative to TF-IDF the robustness gain here is modest; the sharper contrast is against rule- and template-driven matching.

5.3. Cross-Domain Transfer Without Target Labels

Under the source-selected threshold, every model's target-domain F1 falls to 0.22–0.32, while the transferable variants retain AP between 0.87 and 0.95 (Table 5). The preserved AP shows that these models continue to rank many malicious windows above benign ones, whereas the low F1 shows that the source operating point does not transfer. We therefore attribute this discrepancy to threshold mismatch rather than to a complete loss of discriminative signal. This evidence alone should not be interpreted as proof of probabilistic miscalibration; Brier score and ECE provide the separate assessment of that property.

Quantile transport (Equation (3)), using zero target labels, recovers LoRA+SICT from F1 $= 0.259$ to 0.865 on WINDOWS$\rightarrow$LINUX, close to the labeled oracle (0.872) and ahead of 20-shot threshold selection (0.782). Plain LoRA follows (0.777 versus 0.784), and in the harder LINUX$\rightarrow$WINDOWS direction MIL transports best (0.794 versus 0.822; vanilla only 0.429). Two regularities emerge: SICT improves what LoRA transfers, and MIL improves what full fine-tuning transfers, both by constraining the encoder away from source-specific shortcuts (Figure 3). The result should not be read as transformers transferring well in the generic sense, since the gains concentrate in specific variants while vanilla full fine-tuning stays weak. The prevalence-free alternatives (Otsu, GMM) prove unreliable for the neural scores; transporting the source positive rate is the method of choice whenever source and target prevalence are comparable.

Table 5. Cross-domain F1 on the target domain by threshold policy. Transport uses zero target labels; n-shot uses n labeled target windows; oracle uses the full labeled target test.

Direction	Model	Source	20-shot	Transport	Oracle
LINUX $\rightarrow$ WINDOWS	TF-IDF+LR	0.222	0.439	0.600	0.612
	Heuristic	0.220	0.590	0.225	0.783
	Transformer	0.224	0.351	0.429	0.456
	Transformer+LoRA	0.254	0.323	0.422	0.441
	Transformer+LoRA+SICT	0.243	0.523	0.409	0.566
WINDOWS $\rightarrow$ LINUX	TF-IDF+LR	0.222	0.480	0.616	0.655
	Heuristic	0.318	0.733	0.318	0.841
	Transformer	0.250	0.340	0.539	0.584
	Transformer+LoRA	0.292	0.621	0.777	0.784
	Transformer+LoRA+SICT	0.259	0.787	0.865	0.872

5.4. Event Localization from Weak Labels

Evaluated against ground-truth event labels on mixed windows, those containing both benign and malicious events, the MIL attention weights, trained with window labels alone, reach localization AUC 0.966 in both domains with hit@1 up to 0.989 (Figure 4). This is

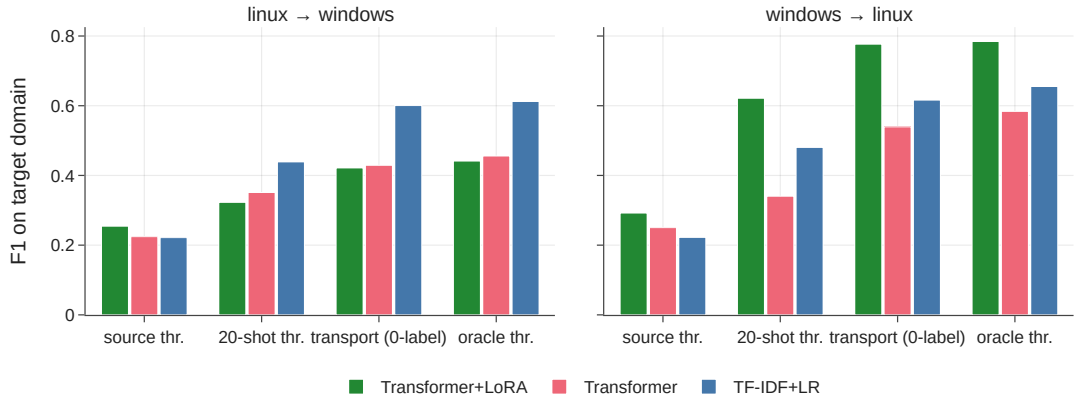


Figure 3. Cross-domain F1 under different threshold policies. In both transfer directions, the source-selected threshold yields poor target-domain F1, while 20-shot threshold selection and label-free quantile transport recover much of the lost performance. Together with the preserved ranking metrics, this result is consistent with substantial operating-point mismatch rather than complete loss of target-domain ranking signal.

both higher and far more stable (std ± 0.01–0.03) than the post-hoc `[CLS]`-attention baseline (0.889 ± 0.16 on Linux, 0.781 ± 0.13 on Windows), whose quality hinges on whatever the classification head happened to attend to. Adding SICT to the MIL head slightly diffuses the attention (Loc-AUC 0.85–0.90), buying detection robustness at a small localization cost. MIL is therefore our clearest architectural contribution: the model does not merely flag the window, it points at the right events, and Section 5.5 shows the same pattern, albeit at lower absolute scores, on real per-line-labelled logs.

5.5. External Validity on Real Corpora

We finally replicate each contribution on public logs at the label granularity it needs, testing which parts of the synthetic story survive noisy operational data rather than claiming universal superiority. Detection stays competitive but not universal: the MIL transformer reaches F1 0.758 on HDFS, 0.998 on BGL, and 0.93/0.82 on the two AIT organizations, against TF-IDF's 0.736/0.973/0.942/0.906, which wins both AIT splits; no model exceeds 0.76 on HDFS, so the synthetic near-saturation does not recur.

Because the two AIT organizations share one multi-step attack scenario, they test cross-organization portability: transferred to the other organization with zero target labels, models retain 88–98% of oracle F1 (MIL 0.750/0.847 vs. 0.856/0.874 under quantile transport). With per-line labels available, MIL again localizes attack events from window labels alone (AUC 0.731, hit@1 0.676 on BGL; hit@1 around 0.72 on AIT), reproducing the weak-label localization claim at lower absolute separability than in synthetic data.

Threshold transport across HDFS↔BGL (positive prevalence 49% vs. 16%) behaves exactly as predicted: HDFS→BGL lifts F1 from 0.294 to 0.416 with zero target labels (oracle 0.627), whereas the reverse direction, where source prevalence badly undershoots the target, degrades; labeled threshold selection remains safer under strong prevalence shift. SICT's premise of interchangeable tooling tokens (`nmap`/`masscan`) misses

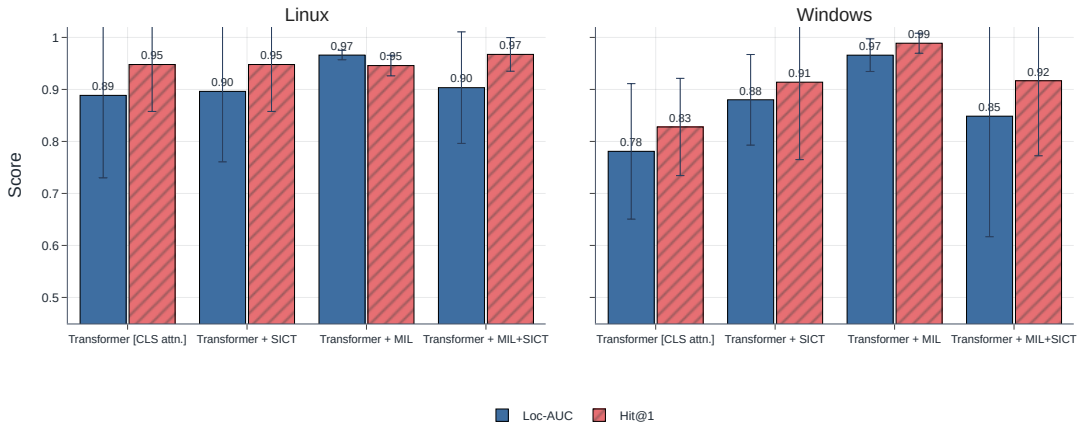


Figure 4. Event localization from weak window labels. We compare localization AUC and hit@1 on mixed windows for the vanilla transformer, SICT, MIL, and MIL+SICT variants. MIL yields the strongest and most consistent localization in both domains, showing that the model can identify malicious events inside flagged windows without event-level supervision during training.

AIT's structural HTTP/DNS signal: the rewrite touches fewer than 3% of test lines and leaves F1 unchanged (clean = mutated ≈ 0.93), so we scope SICT to command- and endpoint detection and response (EDR)-style logs and report this as a boundary rather than a win.

In-domain, all strong models saturate (transformer F1 0.994/1.000, TF-IDF 0.974/0.994) with low calibration error (ECE ≤ 0.002). Calibration is conceptually separate from transport: Equation (3) relies only on the source predicted-positive rate and the ordering of target scores, not on calibrated probabilities. In-domain accuracy does not separate the approaches; the generalization protocols do.

6. Discussion and Limitations

6.1. Role of Each Component

The three contributions address distinct stages of the detection pipeline: SICT modifies training, MIL pooling, and threshold transport the decision rule. MIL yields the most consistent gains, preserving window-level detection while enabling event localization from window labels, whereas threshold transport provides the strongest cross-domain improvement by converting ranking quality into higher target-domain F1 without target labels. SICT mainly helps with unseen signatures and semantically equivalent behaviors expressed through different commands or tools, but is less useful when meaningful rewrites are unavailable. Overall, these methods offer limited gains in easy in-domain synthetic settings, where TF-IDF remains competitive and efficient, but become most valuable under unseen attacks, lexical variation, misleading benign tokens, and cross-domain shifts.

6.2. Position Relative to Prior Findings

Our results align with, and extend, recent findings. The unseen-signature collapse of rule- and template-count baselines reproduces the instability documented by LogRo-

bust [Zhang et al. 2019], with SICT adding a training-time consistency remedy instead of post-hoc embedding smoothing. SwissLog [Li et al. 2022] likewise finds localization signal in attention aggregation, though unsupervised; our MIL head derives it from weak window labels under explicit ranking metrics (AUC, hit@1). LLM-based detectors such as LogLLM [Guan et al. 2024], as surveyed by LLM4Log [Ma et al. 2026], still evaluate window- or sequence-level accuracy; the failure modes studied here are complementary and merit reporting alongside headline accuracy for any backbone. Our external-corpora pattern, where lexical baselines can win and neural gains concentrate under shift, echoes the mixed Loghub benchmarking picture [Zhu et al. 2023].

Practically, TF-IDF suffices when logs are stable and labels abound; the neural stack pays off under signature churn, for triage that needs event localization (hit@1 up to 0.99 synthetic, around 0.72 on real dense-attack windows), and across environments, where threshold transport recovers most of the lost F1 without target labels, provided prevalences are comparable.

6.3. Threats to Validity

The primary benchmark is synthetic, so conclusions emphasize relative behavior under entity-disjoint, leak-resistant protocols rather than absolute scores. HDFS, BGL, and AIT-LDS support the main findings: detection remains competitive, threshold transport shows similar recovery/prevalence dependence, and MIL localizes real attack events from window labels alone, suggesting these patterns are not purely synthetic.

Scope remains limited. SICT fits command- and EDR-style logs better than AIT's HTTP/DNS-heavy setting, where TF-IDF performs better. Evaluation covers only two AIT organizations and subsampled HDFS/BGL partitions, and three seeds limit statistical resolution. Threshold transport is label-free but transductive and assumes similar predicted-positive prevalence across domains; large shifts, as in HDFS↔BGL, reduce performance, while streaming would require online quantile estimation.

7. Conclusion

We reframed robust log-attack detection as three separable problems, namely generalizing beyond seen signatures, localizing malicious events from weak labels, and transporting the operating point across domains, and validated each under leak-free protocols. The strongest evidence comes from the controlled generalization tests: MIL+SICT detects unseen signatures, MIL supplies quantitative event localization, and threshold transport recovers much of the cross-domain F1 lost when a source-selected operating point is applied to the target domain. Experiments on public corpora show these effects carrying partway to real logs while also marking their boundaries: lexical baselines can still win, and SICT matters only where the log surface admits meaningful rewrites. Treating representation robustness, event localization, probability calibration, and operating-point selection as distinct, separately testable problems is what makes each gain measurable and its limits honest.

References

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *CCS*, pages 1285–1298.

Guan, W., Cao, J., Qian, S., Gao, J., and Ouyang, C. (2024). LogLLM: Log-based anomaly detection using large language models. *arXiv preprint arXiv:2411.08561*.

Guo, H., Yuan, S., and Wu, X. (2021). LogBERT: Log anomaly detection via BERT. In *IJCNN*, pages 1–8.

He, P., Zhu, J., Zheng, Z., and Lyu, M. (2017). Drain: An online log parsing approach with fixed depth tree. In *ICWS*, pages 47–56.

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *ICLR*.

Ilse, M., Tomczak, J., and Welling, M. (2018). Attention-based deep multiple instance learning. In *ICML*, pages 2127–2136.

Le, V. and Zhang, H. (2021). Log-based anomaly detection without log parsing. *arXiv preprint arXiv:2108.01955*.

Li, X., Chen, P., Jing, L., He, Z., and Yu, G. (2022). SwissLog: Robust anomaly detection and localization for interleaved unstructured logs. *IEEE Transactions on Dependable and Secure Computing*.

Liang, X., Wu, L., Li, J., Wang, Y., Meng, Q., Qin, T., Chen, W., Zhang, M., and Liu, T.-Y. (2021). R-Drop: Regularized dropout for neural networks. In *NeurIPS*.

Ma, Z., Yang, J., and Chen, T.-H. (2026). LLM4Log: A systematic review of large language model-based log analysis. *arXiv preprint arXiv:2604.16359*.

Meng, W., Liu, Y., Zhu, J., et al. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, pages 4739–4745.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

Saerens, M., Latinne, P., and Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. (2020). Unsupervised data augmentation for consistency training. In *ICLR*.

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., and Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. In *ESEC/FSE*, pages 807–817.

Zhu, J., He, S., He, P., Liu, J., and Lyu, M. R. (2023). Loghub: A large collection of system log datasets for AI-driven log analytics. In *ISSRE*.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., and Lyu, M. (2019). Tools and benchmarks for automated log parsing. In *ICSE-SEIP*, pages 121–130.