

Do You Need a Sequence Model? A Validated Order-Reliance Audit for Temporal Security Detection

Anonymous Author(s)

Affiliation withheld for double-blind review

Abstract—Should you pay for a deep sequence model? LSTMs, temporal CNNs, and transformers are increasingly deployed for behavioural security detection—fed file-system, storage, memory, or API event streams—on the assumption that the temporal order of activity carries discriminative signal worth their extra training, tuning, and deployment cost. That assumption is rarely tested, and our central finding is sobering: *even when temporal order genuinely carries signal, and a sequence model is the more accurate windowed classifier, an order-invariant baseline that simply pools over time matches or beats it on the early-detection metric that governs deployment.* We reach this through a cheap, model-agnostic *order-reliance audit*, which we also *validate* as an instrument: it scores a trained model on correctly ordered windows and again with their timesteps permuted, reporting the *reliance gap* ΔF_1 against a permutation null with a one-sided p -value. On a controlled generator it returns zero when order is irrelevant (specificity) and rises steadily as injected order signal grows (sensitivity). Applying it to crypto-ransomware detection across a synthetic benchmark and two real datasets (RanSAP storage access, RanSMAP memory access), we find order-reliance is *architecture-dependent*: a transformer (global attention, CLS pooling) extracts no temporal signal even when enlarged and is matched by a calibrated order-invariant Random Forest, yet a temporal CNN and an LSTM robustly exploit order (gap 0.04–0.06, all five training seeds significant) and beat the pooled baseline by 0.04–0.12 macro- F_1 (both significant under 5-fold cross-validation), especially for family identification. Judging by the transformer alone, one would wrongly conclude “order does not matter”. On a second domain, malware API-call sequences, the audit fires for every architecture (ΔF_1 0.08–0.15), confirming it detects order when present—yet there a bag-of-calls baseline still wins. So order-reliance, classification accuracy, and early-detection value *dissociate*: a positive audit says a model *uses* order, not that you *need* a sequence model, and across both domains pooled baselines kept winning the deployment metric. The question is best answered empirically and per-architecture; we release the audit as a small, reusable library.

Index Terms—sequence models, temporal modelling, evaluation methodology, order reliance, ransomware detection, malware classification, random forest, transformer

I. INTRODUCTION

Behavioural security work feeds time-ordered signals—file-system events, storage or memory access patterns, system or API calls—to deep *sequence* models, on the premise that the *temporal trajectory* of activity distinguishes malicious from benign work. Sequence models are markedly more expensive to train, tune, and deploy than classical ones, and the security-ML literature repeatedly shows apparent gains often stem from evaluation artefacts rather than real signal [1]. Yet a basic

question is seldom asked before adopting one: *does temporal order actually carry discriminative signal for this task, or do per-window aggregates suffice?*

We make this question testable with an **order-reliance audit**. Given any trained sequence model, we evaluate it on correctly ordered test windows and again on the same windows with their timesteps randomly permuted. A model that relies on the trajectory must lose accuracy when order is destroyed; the *reliance gap* $\Delta F_1 = F_1^{\text{ordered}} - F_1^{\text{shuffled}}$ quantifies this dependence. The audit is model-agnostic, costs one extra inference pass per permutation, and—paired with an order-invariant baseline that pools over time—tells us directly whether sequence modelling is warranted.

A diagnostic is only as trustworthy as its calibration, so our first contribution is to **validate the audit as an instrument**. On a controlled generator whose label lives purely in temporal order, we show the audit returns ≈ 0 when order is irrelevant (specificity) and a value that rises steadily with injected order strength (sensitivity), using the same architecture we audit elsewhere. We then apply it to crypto-ransomware detection across a synthetic benchmark and two real datasets with per-write entropy, RanSAP [9] (storage block writes) and RanSMAP [10] (memory page writes), and to a second, non-ransomware domain, malware family classification from Windows API-call sequences [11]. The headline: a sequence model can rely on order *and* classify better, yet still lose on the deployed metric to a pooled baseline—so the title’s question rarely needs a “yes”. In detail:

- **Order-reliance is architecture-dependent.** On ransomware metadata a transformer extracts no order signal (every gap statistically indistinguishable from zero against its own null) and a pooled Random Forest matches it—so the transformer alone would suggest “order is irrelevant”—yet a temporal CNN and an LSTM *robustly* exploit order (gap 0.04–0.06, significant on all five seeds) and beat the pooled baseline by 0.04–0.12 macro- F_1 , strongest for family identification and on memory data (Fig. 2). An enlarged transformer still finds ≈ 0 , so the blindness is architectural, not undertraining.
- **The audit is validated and travels.** It is calibrated for sensitivity and specificity (Fig. 1) and fires for every architecture on malware API-call sequences (ΔF_1 0.08–0.15)—so the near-zero transformer gap on ransomware reflects that data, not a dead audit.

- **Order-reliance, accuracy, and onset value dissociate.** On the API task all models rely on order yet a bag-of-calls forest wins on accuracy; on ransomware the temporal CNN both relies on order and out-classifies the forest *on family identification*, yet the order-invariant forest detects infection *onset* earliest at matched false-positive budgets (§VII-C). The audit measures whether a model *uses* order; a paired baseline on the deployed metric decides whether you *need* it—and across both domains pooled baselines keep winning that metric.
- **The hard, architecture-independent problem is cross-family generalisation:** all models collapse on unseen TeslaCrypt and WannaCry.

Our contributions are (i) a *validated*, model-agnostic order-reliance audit and a small suite of fair-evaluation protocols, released as a reusable library; (ii) the empirical, statistically rigorous finding that order-reliance in ransomware metadata is architecture-dependent, with a methodological lesson—audit several architectures before declaring a task order-invariant; and (iii) a reproducible benchmark spanning a controllable generator and three real datasets across two security domains. Code, data pipelines, seeds, and a one-command reproduction are released.¹

Decision recipe. Before paying for a sequence model: (i) train it and run the order-reliance audit; (ii) on the metric you will *deploy*, compare it against an order-invariant baseline that pools over time; (iii) do both for *more than one* architecture. Adopt sequence modelling only if a model both *uses* order *and* beats the pooled baseline on that metric—in our study, no real task cleared this bar.

II. BACKGROUND AND RELATED WORK

Behavioural ransomware detection. CryptoDrop [6] and UNVEIL [7] detect ransomware from file-system behaviour and the entropy of written data; ShieldFS [8] monitors I/O request packets and entropy at the filesystem layer. These establish entropy and write/rename bursts as strong indicators but largely use hand-crafted thresholds or classical models.

Sequence models and evaluation pitfalls. Following transformers [2], sequence architectures are now common in malware and intrusion detection. They add substantial cost and are prone to over-estimated performance under loose evaluation; Arp *et al.* [1] catalogue pitfalls (spatial/temporal leakage, base-rate neglect, inappropriate baselines). Our audit is in that spirit: a lightweight, mandatory check of a specific, untested assumption.

Lineage of the audit. The reliance gap is a permutation-based importance measure—to *temporal order* what permutation feature importance [3] is to a feature. It is inspired by the NLP word-order literature: Sinha *et al.* [4] pre-train masked language models on shuffled sentences and find downstream performance largely preserved; Pham *et al.* [5] show 75–90% of fine-tuned BERT predictions unchanged when words are permuted. We adapt this to security time-series, add a

permutation-null test and an order-invariant baseline, and—unlike that work—first validate the probe’s sensitivity and specificity on a controlled generator.

Real behavioural datasets. RanSAP [9] provides storage-access traces (block reads/writes with per-block entropy) for seven ransomware families and benign software; RanSMAP [10] adds memory-access traces. Both include high-entropy benign tools (a legitimate encryptor, an archiver, a secure-delete tool), making the benign class genuinely hard. For the second domain we use Mal-API-2019 [11], 7,107 Cuckoo-Sandbox API-call sequences over eight malware families. All are externally collected datasets we did not design.

III. THE ORDER-RELIANCE AUDIT

Let a detector consume a window $X = (x_1, \dots, x_T)$ of T per-timestep feature vectors and output class probabilities. For a trained model f , define the time-permutation $\pi(X) = (x_{\sigma(1)}, \dots, x_{\sigma(T)})$ for a random permutation σ of the non-padding timesteps—this leaves the per-window multiset (hence every pooled statistic) unchanged and destroys only order. Because the per-timestep marginals are preserved, a non-zero gap reflects *use of order* rather than mere off-distribution inputs—a confound the $\alpha=0$ specificity control (§IV) rules out directly, since the gap is ≈ 0 there under identical shuffling. Drawing K independent permutations $\{\pi_k\}$ gives a *permutation null*; we report

$$\Delta F_1 = F_1(f, D) - \frac{1}{K} \sum_k F_1(f, \pi_k(D)), \quad (1)$$

the central 95% of that null as a reference band for $\Delta F_1=0$ (not a confidence interval on the estimate, which the execution bootstrap supplies), and a one-sided permutation p -value $(1 + \#\{k : F_1(\pi_k) \geq F_1^{\text{ord}}\})/(K+1)$, all with the *same* trained model. $\Delta F_1 \approx 0$ with large p means the model’s decision is order-invariant; a large, low- p gap means it relies on order.

Complementary fair-evaluation protocols. The gap is read alongside an *order-invariant baseline* (a classifier on time-pooled features), *execution-level splits* (no execution in both train and test), *matched false-positive* onset curves, and *clustered/paired bootstrap* tests for every “A beats B” comparison (windows within an execution are correlated); we release these as a small library (§VIII).

IV. VALIDATING THE AUDIT

A near-zero gap is only meaningful if the audit would have reported a large one had order mattered. We therefore calibrate it on a controlled binary task whose label lives *only* in temporal order: class 1 is a rises-then-falls ($\wedge$) trajectory of per-window levels, class 0 the falls-then-rises ($\vee$) trajectory of the *identical* level multiset, so any time-pooling model is at chance and only order separates them. An order-signal strength $\alpha \in [0, 1]$ mixes each example between its class trajectory (probability α) and a random permutation. We audit the *same* transformer used throughout.

Fig. 1 reports the result over five seeds. The reliance gap is ≈ 0 at $\alpha=0$ (mean -0.007 , $p=0.86$), rises *monotonically* through $\alpha=0.8$ ($-0.007 \rightarrow +0.50$), then plateaus near $+0.5$

¹Anonymised artifact (code, data pipelines, seeds, one-command reproduction): <https://anonymous.4open.science/r/ransomtrace>; de-anonymised at camera-ready.

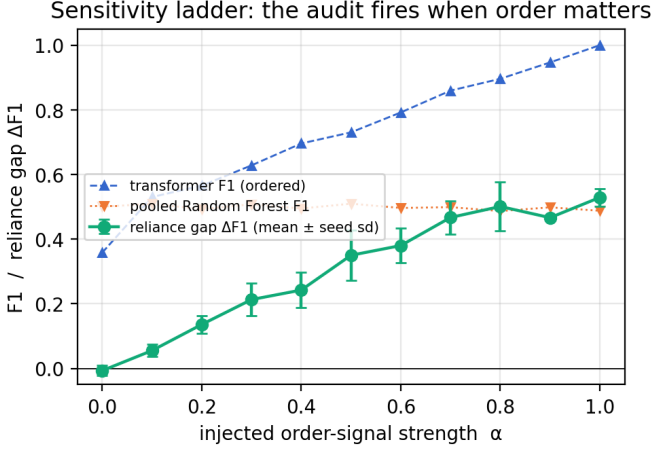


Fig. 1. The audit is a validated instrument. As injected order-signal strength α grows, the reliance gap rises from ≈ 0 (specificity) toward $\approx +0.5$ (sensitivity, plateauing once order saturates), while a time-pooled Random Forest stays at chance. Same transformer architecture as the security experiments; mean $\pm$ seed sd over five seeds, $K=200$.

as the order signal saturates ($+0.53$ at $\alpha=1$); it becomes significant from $\alpha=0.1$ ($\Delta F_1 \approx 0.055$), while the pooled Random Forest stays at chance (≈ 0.50) for every α . This establishes **specificity** (also confirmed by a memoryless control, $\Delta F_1 = -0.0002$, $p=0.96$) and **sensitivity**. It also proves the transformer *order-capable*: carrying sinusoidal positional encodings, a permutation-invariant readout would post a gap of exactly zero on this identical-multiset task, yet its gap reaches $+0.53$ at $\alpha=1$ —so the near-zero ransomware gaps are an empirical inductive-bias result, not a definitional artefact of an order-blind architecture. The smallest reliably detected rung, $\alpha=0.1$ ($\Delta F_1 \approx 0.055$), is our calibrated *detection floor*. Because it is calibrated on a *pure-order* task and on the *transformer’s* ladder, on real, order-entangled tasks and for the CNN and LSTM we treat it as a guide read jointly with seed-stable significance, not a hard threshold; that the audit fires far above it for all three architectures on the API domain supports this cross-architecture use.

V. BENCHMARK, DATASETS, AND FEATURES

Synthetic benchmark. A controllable generator of metadata-only traces (no malware is executed) with benign, suspicious, and ransomware classes, supporting a temporal “kill-chain” mode and a *memoryless* twin (3000 traces). It includes out-of-distribution “hard” variants (benign high-entropy workflows, low-and-slow encryption) and *counterfactual* trace groups that differ only in temporal structure, to stress order-reliance specifically.

RanSAP / RanSMAP (real). Each execution’s write stream (timestamp, address, size, entropy) is binned into *disjoint* windows of 512 consecutive writes; each window is one timestep with 16 aggregate features (entropy statistics, I/O rate/size, address-locality). A model input is a sliding sequence of 32 such windows: the step-8 stride

overlaps successive *inputs*, but the 32 bins *within* an input never share writes, and the audit permutes timesteps within each input independently—so the order-reliance gap cannot be an artefact of within-input window overlap. We use the win7-120gb-hdd (RanSAP; seven families) and celeron-gen6/ddr4-1600-16g (RanSMAP; six families) configurations. Labels are encoded 0 (benign) and 2 (ransomware): the *suspicious* class 1 exists only in the three-class synthetic generator and is absent from the real datasets, so real-data macro- F_1 is computed over $\{0, 2\}$. Splits are execution-level.

Mal-API-2019 (real, second domain). 7,107 samples over eight malware families; each is the first 100 API calls, one-hot over the 100 most frequent APIs plus an “other” bucket. The order-free baseline is a bag-of-APIs Random Forest on the call-frequency histogram. Each sample is its own group for resampling.

VI. EXPERIMENTAL SETUP

We compare, on identical features: a **transformer** (model dim 64, 4 heads, 2 layers, sinusoidal positional encodings, CLS pooling), a **bi-LSTM** (64, 2 layers), a **temporal CNN** (dilated, 64, 3 blocks), an **order-invariant Random Forest** (300 trees, depth 16) on mean-pooled features, and a flattened **Logistic Regression**. Sequence models use class-balanced cross-entropy, AdamW, gradient clipping, and per-feature standardisation frozen at train time so ordered and shuffled inference are processed identically. Seeds are fixed and recorded; the audit uses $K=200$ permutations and 2000 clustered bootstrap resamples unless noted. To separate architectural limits from undertraining we also audit a larger transformer (dim 128, 4 layers, 40 epochs).

VII. RESULTS

A. Order-reliance is architecture-dependent

We audit the transformer’s reliance gap across six settings, each tested against *its own* permutation null (its width shrinks as the test set grows, so the settings cannot share one band). On binary and 3-class detection the gap is statistically indistinguishable from zero: the two synthetic 3-class controls fall inside the memoryless no-signal band $[-0.0004, +0.0003]$, and the two binary gaps, judged against their own nulls, are non-significant ($p=0.39, 0.91$)—and (Table I) a pooled Random Forest is statistically tied with the transformer. The two family-ID gaps are larger ($+0.0087, +0.0046$) but only marginal against their own wider nulls ($p=0.06, 0.03$; the differing significance of a larger and a smaller gap is precisely this null-width effect), and neither is robust—over five training seeds each averages ≈ 0 and is significant in 1/5 (RanSMAP-family mean -0.001). No setting shows a stable order-reliance gap *for the transformer*.

A single architecture’s invariance is not proof the data lack order signal. Re-running the audit with an LSTM and a temporal CNN overturns the strong reading (Fig. 2). Both show *robust* reliance gaps: on RanSAP family, RanSMAP binary, and RanSMAP family every one of five training seeds

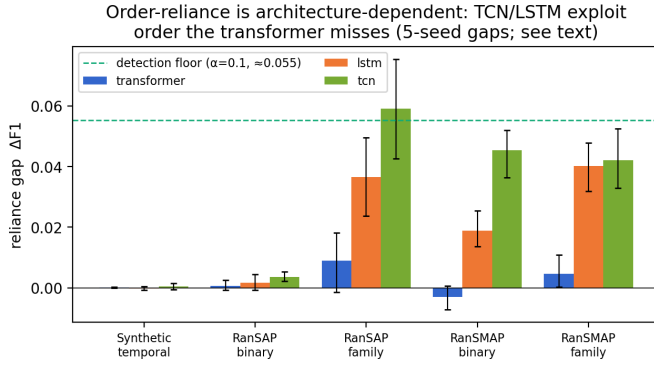


Fig. 2. Order-reliance is architecture-dependent. The transformer’s gap (blue) stays at zero on every setting; the LSTM (orange) and temporal CNN (green) show robust gaps reaching the calibrated detection floor, and also beat the pooled baseline. Single-seed gaps with permutation-null CIs; see §VII-A for the five-seed robustness.

is significant and positive, with mean ΔF_1 of 0.039–0.055—around the calibrated detection floor, versus the transformer’s ≈ 0 . The operative criterion is seed-stable significance *together with* an effect size near or above the floor: a gap just below 0.05 that is significant on all five seeds is real, whereas a sub-floor gap that flips sign across seeds (the transformer’s) is not. These order-exploiting models also beat the order-invariant Random Forest on accuracy, and the advantage is *five-seed stable*: with the execution-level split held fixed, the model–RF macro- F_1 is positive on all five seeds for both, with means $+0.084/+0.043/+0.122$ (TCN) and $+0.043/+0.038/+0.112$ (LSTM) on RanSAP-family / RanSMAP-binary / RanSMAP-family (seed sd ≤ 0.019). The single-seed paired bootstrap (Table I) confirms the TCN on all three settings ($p \leq 0.008$) and the LSTM on RanSMAP-family ($p=0.005$; only marginal on the other two, where the 12–14-execution test set underpowers a single split). We settle test-sampling uncertainty directly with 5-fold execution-level cross-validation (every execution tested out-of-fold exactly once): both order-exploiting models beat the forest on all three settings, positive on 5/5 folds, all paired $p \leq 0.001$ (LSTM $+0.100/+0.062/+0.085$, TCN $+0.109/+0.053/+0.102$ for RanSAP-family / RanSMAP-binary / RanSMAP-family), and the LSTM’s two single-split-marginal cases become decisively significant. The transformer stays the order-blind reference: it *trails* the forest on RanSAP-family (-0.044 , $p=0.07$), ties on RanSMAP-binary ($+0.030$, $p=0.10$), and wins only RanSMAP-family ($+0.058$, $p=0.002$); the enlarged transformer still reads ≈ 0 (range -0.005 to $+0.009$), so its blindness is architectural. Two routes to beating the pooled baseline thus coexist: the transformer’s RanSMAP family-ID edge is *capacity*—order-invariant and dataset-specific, since it reverses to a loss on RanSAP—whereas the CNN’s is *temporal*. The data carry exploitable, class-discriminative order signal; whether a model benefits depends on its inductive bias, and the multi-architecture audit is what reveals it.

TABLE I

DETECTION MACRO- F_1 BY ARCHITECTURE (SEED 42, $K=100$; EXECUTION-LEVEL BOOTSTRAP 95% CI; Δ AND PAIRED p VS. THE ORDER-INVARIANT RANDOM FOREST). THE LOWER THREE SETTINGS ARE THOSE WITH ROBUST, SEED-STABLE RELIANCE GAPS; *RanSAP binary* (TOP) IS THE SATURATED DETECTION CONTRAST—ALL ARCHITECTURES SIT NEAR 0.99 AND NONE SIGNIFICANTLY OUT-CLASSIFIES THE FOREST (TCN $+0.011$, $p=0.31$), MATCHING ITS NEGLIGIBLE RELIANCE GAP THERE (§VII-A) AND THE ONSET FINDING THAT THE POOLED FOREST IS ALREADY AN EXCELLENT DETECTOR; IT IS SHOWN SINGLE-SEED, NOT CLAIMED SEED-ROBUST. ON THE THREE ROBUST SETTINGS THE ORDER-EXPLOITING TCN BEATS THE POOLED FOREST AND THE LSTM BEATS IT ON FAMILY ID; THE TRANSFORMER’S LONE WIN (RANSMAP FAMILY) IS ORDER-INVARIANT CAPACITY (ITS RELIANCE GAP IS ≈ 0 , §VII-A). THE TABULATED MACRO- F_1 AND PAIRED p ARE AT THE REFERENCE SEED; ACROSS FIVE TRAINING SEEDS (SPLIT HELD FIXED) THE Δ -VS-RF ADVANTAGE IS POSITIVE ON ALL FIVE FOR BOTH THE TCN AND LSTM ON THOSE THREE SETTINGS, AND A 5-FOLD EXECUTION-LEVEL CV CONFIRMS IT (BOTH IN §VII-A). THE FAMILY AND MEMORY MARGINAL CIs ARE WIDE—ONLY 12–14 TEST EXECUTIONS—SO THE *paired* TEST, NOT THE OVERLAPPING MARGINAL CIs, IS THE RELIABLE COMPARISON.

Setting	Model	macro- F_1 [95% CI]	Δ vs RF (p)
RanSAP binary	RF	0.986 [0.95, 1.00]	—
	Transformer	0.993 [0.98, 1.00]	+0.008 (0.45)
	LSTM	0.993 [0.98, 1.00]	+0.008 (0.41)
	TCN	0.996 [0.99, 1.00]	+0.011 (0.31)
RanSAP family (7)	RF	0.765 [0.49, 0.80]	—
	Transformer	0.686 [0.42, 0.74]	−0.079 (0.16)
	LSTM	0.796 [0.52, 0.83]	+0.032 (0.15)
	TCN	0.873 [0.59, 0.89]	+0.108 (0.008)
RanSMAP binary	RF	0.901 [0.84, 0.93]	—
	Transformer	0.929 [0.88, 0.94]	+0.028 (0.18)
	LSTM	0.932 [0.88, 0.95]	+0.031 (0.08)
	TCN	0.955 [0.91, 0.97]	+0.054 (0.003)
RanSMAP family (6)	RF	0.787 [0.50, 0.83]	—
	Transformer	0.871 [0.57, 0.89]	+0.084 (<0.001)
	LSTM	0.889 [0.58, 0.91]	+0.101 (0.005)
	TCN	0.902 [0.59, 0.93]	+0.115 (<0.001)

B. Window size

Re-ingesting RanSAP and RanSMAP at 512/128/32 writes per window, the transformer’s gap turns significant at the finest bins only because the test sets grow to 24–38k windows; the *effect* stays negligible ($\Delta F_1 \leq 0.013$, $\sim 4\times$ below the floor), so its order-invariance is robust toward raw events—a case where a bare p misleads and the calibrated effect-size yardstick (§IV) matters.

C. RanSAP: detection, generalisation, onset

On ransomware-vs-benign detection (Table I) the transformer reaches macro- F_1 0.993 vs. 0.986 for the Random Forest—a paired difference within noise ($p=0.45$). Because pure RanSAP traces are high-entropy from the first window, per-trace latency is degenerate, motivating two sharper analyses.

Zero-day family generalisation. Training on six families and testing on the held-out seventh, the Random Forest has the higher mean recall (0.811 vs. 0.776), but the paired difference across families is *not* significant (Wilcoxon $p=0.094$). Both

models detect five families near-perfectly and collapse on unseen TeslaCrypt (0.24/0.11) and WannaCry (0.45/0.37): cross-family generalisation, not architecture, is the bottleneck.

Onset at matched false positives. We compose transition traces (a held-out benign execution followed by a held-out ransomware execution) and measure high-entropy writes lost before the first true alarm—the data a victim cannot recover, hence our deployment metric—calibrating *every* detector’s threshold to a 99.5-percentile benign budget so speed is compared at an equal false-alarm rate (Fig. 3). The order-invariant Random Forest is the *best* early detector: at a tight budget it loses fewer writes than the transformer (2950 vs. 3902, paired $p < 0.001$) and fewer than the order-exploiting TCN at *every* budget (2950 vs. 3467 at 0.5, $p=0.001$; 2406 vs. 3358 at 5.0, $p < 0.001$). This is not an artefact of the *abrupt* benign→ransomware boundary: under a gradual stochastic-mixture ramp (the ransomware-window fraction rising over 16–32 bins), the forest still loses fewer high-entropy writes than the TCN at every budget (all $p < 0.001$), and no sequence model attains a significant early-detection advantage at any budget or ramp length.

The LSTM’s raw streaming probabilities are poorly calibrated (benign 99.5-percentile ransomware probability 0.73), so it never reaches 100% detection at a deployable budget. Post-hoc temperature scaling [12] on a held-out split ($T=1.45$) improves NLL (0.046 → 0.039) yet still yields no 100%-detection point. Because the streaming score is the ransomware posterior of a *three*-class softmax, temperature *can* re-rank windows—so a new operating point is *permitted* to appear and none does; what this rules out is the *scale* explanation, leaving intrinsic per-window separability, which a global temperature cannot repair. We therefore omit the LSTM as *under-ranked for streaming*, not merely miscalibrated. So the architecture-dependence of *order-reliance* does *not* carry over to onset. On the binary onset task order-reliance is negligible for *every* architecture—the TCN’s gap clears its null yet is only $\Delta F_1=0.004$, an order of magnitude below the floor (vs. 0.059 on RanSAP-family)—and on that audit the TCN is at most nominally the better binary classifier (+0.011 macro- F_1 , paired $p=0.31$). The dissociation is thus *across* the evidence base—order-use and classification superiority live on family ID, early-detection value on the binary onset task—not a within-task contradiction. Here the per-window decisiveness a pooled forest has in abundance matters more than modelling the trajectory: “uses order” and “is a better classifier” are not “is a better early detector”. (The transformer’s apparent edge at a single fixed threshold was a *threshold-selection* artefact that vanishes under this control.)

D. Second domain: malware API-call sequences

To show the audit travels and fires when order matters on real data, we audit malware family classification from API-call sequences. Here every architecture has a large, significant reliance gap—transformer +0.083, LSTM +0.144, TCN +0.149, all $p=0.005$, far above the detection floor and every ransomware gap. The *same* transformer that read ≈ 0 on

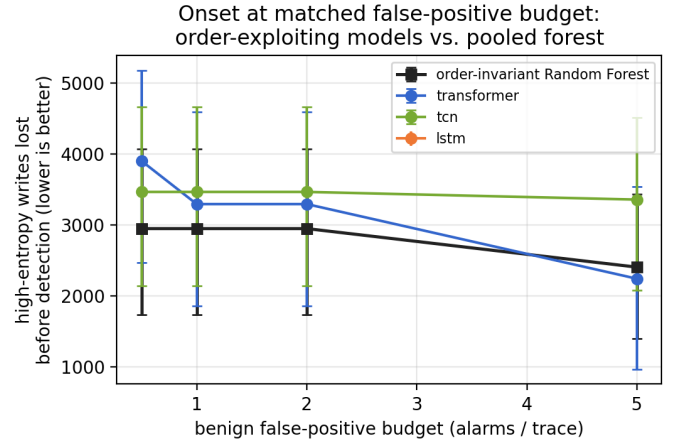


Fig. 3. Onset detection on benign→infection transition traces at matched benign false-positive budgets (per-trace bootstrap CIs). The order-invariant Random Forest loses the fewest high-entropy writes before detection—fewer than the order-exploiting TCN at every budget; the LSTM’s streaming probabilities reach no 100%-detection point at a deployable budget—even after temperature scaling—and it is omitted.

ransomware fires +0.083 here, so its near-zero ransomware gap is a property of that data, not a broken model or a dead audit. Yet a subtlety sharpens the message: all three sequence models *rely* on order, but the order-free bag-of-APIs Random Forest (0.49) still *significantly* outperforms every one on accuracy—by 0.02 to 0.05 macro- F_1 ($p=0.007$ vs. the TCN, $p < 0.001$ vs. the LSTM and transformer). A positive gap does not by itself imply a sequence model is *necessary*: the audit measures whether a model uses order; the paired baseline decides whether you need it. Here both say “the models use order, but a pooled model still suffices”—in contrast to the RanSAP-family CNN (§VII-A), which both uses order and beats the baseline.

VIII. THE ORDERAUDIT LIBRARY

We package the audit as a small, dependency-light Python library (NumPy, scikit-learn, SciPy; no deep learning framework required). A single call, `oa.reliance_audit(model.predict_proba, X_test, y_test, K)`, returns the reliance gap with its permutation-null CI and p -value; `oa.pooled_baseline(.)` supplies the order-invariant baseline, and further helpers provide the clustered and paired bootstrap and the matched-false-positive onset curve. A demo (`orderaudit demo`) reproduces the sensitivity/specificity check, and a one-command script regenerates every table and figure here from recorded seeds. Releasing a reusable, validated instrument—rather than a one-off script—is itself part of the contribution: the “do you need a sequence model?” question recurs across behavioural-security tasks.

IX. DISCUSSION

Audit before architecture—across architectures. Before adopting a sequence model, run the reliance audit, with an

order-invariant baseline and *more than one architecture*. A single model’s invariance is not proof the data lack order signal: the ransomware transformer would have us conclude “use a forest”, yet a temporal CNN extracts real temporal structure. Conversely, a positive gap is not proof you need a sequence model: on API calls all models rely on order yet a bag-of-calls forest wins. The pair (reliance gap, sequence–pooled gap) is the actionable diagnostic. We are candid about the deflationary balance of our evidence: in every task we study the *deployment* verdict is *pool*—even where a sequence model is the better windowed classifier—and we have not yet found a *real* task where the audit green-lights a sequence model for the deployed objective. The affirmative branch is not hypothetical: on the pure-order validation task (§IV) the pooled baseline is pinned at chance while the sequence model classifies perfectly at $\alpha=1$, so there the diagnostic returns *use a sequence model*—the real-data verdict of *pool* is a property of those tasks, not a test that can only say no. Two limits remain: the audit runs on an *already-trained* model, economizing deployment, not the training already paid; and in our cases it is the paired baseline, not the reliance gap, that flips the decision. Its present value is a validated guard against *unnecessary* sequence models.

Why a CNN sees what the transformer misses. A pooled or CLS reader sees essentially the order-invariant *multiset* of per-window levels, whereas a dilated convolution reads *inter-window transitions* a time-permutation destroys. This prediction is falsifiable: giving the global-attention transformer a local ± 2 -step attention band lifts its RanSMAP reliance gap from ≈ 0 to $+0.027$ and $+0.037$ ($p=0.005$, approaching the matched TCN), while mean-pooling with global attention recovers nothing—so the decisive ingredient is attention *locality*, not the readout. “Does this task need a sequence model” is entangled with “which sequence model”.

What the baseline saves. Cost cuts the same way: on RanSMAP family-ID the pooled forest—already the equal-or-better detector on the metric you deploy—trains on CPU in 3 s without a GPU (vs. 10–21 s for the sequence models on a GTX 1650) and runs $4.6\text{--}6.3\times$ faster per window at inference (38 vs. 174–239 μs); its only disadvantage is a larger serialised model (23.8 MB). The sequence models charge a real compute premium—and a GPU dependency—for an order-reliance they often do not even use.

Where to invest. The persistent, architecture-independent failure is zero-day family generalisation; effort is better spent on features and training that transfer across families.

X. THREATS TO VALIDITY

We use representative subsets of one hardware configuration per real dataset; absolute numbers may shift with scale, though the reliance gap is reported with CIs and verified across seeds. The real-data test sets are small (12–14 executions for family ID), so marginal accuracy CIs are wide; we therefore rely on the *paired* execution bootstrap and on five-seed sign-stability (the split held fixed), and on 5-fold execution-level cross-validation, which confirms the accuracy advantage over the *full* execution set (LSTM and TCN paired $p \leq 0.001$

on all three settings, §VII-A). The onset analysis uses an abrupt benign→ransomware boundary; the gradual-ramp variant (§VII-C) shifts the absolute writes-lost counts but not the ordering. Our sequence models are modestly sized; we mitigate the “undertrained model” confound by also auditing an enlarged transformer and validating the audit on a controlled generator. The malware features use the first 100 API calls and a top-100 vocabulary; richer representations may shift absolute accuracy but not the audit’s strong firing. Both ransomware datasets originate from one group’s instrumentation; replication on independently collected traces is valuable. Finally, we run many audits; rather than a formal false-discovery correction we adopt a stricter rule—a reliance gap is real only when significant on *all five* seeds *and* near or above the calibrated floor—which is exactly how the transformer’s lone $p=0.03$ family gap and its fine-window “significant” gaps are correctly rejected.

XI. CONCLUSION

We introduced and *validated* an order-reliance audit that asks, cheaply, whether a detection problem needs a sequence model. It shows order-reliance in ransomware metadata is architecture-dependent—a transformer finds none, but a CNN and an LSTM robustly exploit order—yet on every task we study a pooled baseline still matches or beats the sequence model on the deployed metric. The discipline is to audit temporal reliance, against a pooled baseline and across architectures, before paying for temporal models; we release the audit, benchmark, and a one-command reproduction.

REFERENCES

- [1] D. Arp *et al.*, “Dos and Don’ts of Machine Learning in Computer Security,” in *Proc. USENIX Security Symp.*, 2022.
- [2] A. Vaswani *et al.*, “Attention is All You Need,” in *Proc. NeurIPS*, 2017.
- [3] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [4] K. Sinha, R. Jia, D. Hupkes, J. Pineau, A. Williams, and D. Kiela, “Masked Language Modeling and the Distributional Hypothesis: Order Word Matters Pre-training for Little,” in *Proc. EMNLP*, 2021, pp. 2888–2913.
- [5] T. M. Pham, T. Bui, L. Mai, and A. Nguyen, “Out of Order: How Important Is The Sequential Order of Words in a Sentence in Natural Language Understanding Tasks?,” in *Findings of ACL-IJCNLP*, 2021, pp. 1145–1160.
- [6] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, “CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data,” in *Proc. IEEE ICDCS*, 2016.
- [7] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, “UN-VEIL: A Large-Scale, Automated Approach to Detecting Ransomware,” in *Proc. USENIX Security Symp.*, 2016.
- [8] A. Continella *et al.*, “ShieldFS: A Self-healing, Ransomware-aware Filesystem,” in *Proc. ACSAC*, 2016.
- [9] M. Hirano, R. Hodota, and R. Kobayashi, “RanSAP: An Open Dataset of Ransomware Storage Access Patterns for Training Machine Learning Models,” *Forensic Science International: Digital Investigation*, vol. 40, 2022.
- [10] M. Hirano and R. Kobayashi, “RanSMAP: Open Dataset of Ransomware Storage and Memory Access Patterns for Creating Deep Learning Based Ransomware Detectors,” *Computers & Security*, vol. 150, 2025.
- [11] F. O. Catak and A. F. Yazi, “A Benchmark API Call Dataset for Windows PE Malware Classification,” *arXiv:1905.01999*, 2019.
- [12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On Calibration of Modern Neural Networks,” in *Proc. ICML*, 2017, pp. 1321–1330.