

AERCF: What Survives Distribution Shift? An Empirical Evaluation of an Adaptive, Explainable and Resilient Intrusion Detection and Response Architecture

Olanrewaju Adams Agunloye; Aristotle Anana; Simon Mathias; Ezeikel Bakare

Abstract

Background. Machine-learning intrusion detection is routinely reported with near-perfect accuracy, yet deployments disappoint. Frameworks that promise adaptivity, explainability and resilience compound the problem: each property is usually demonstrated in isolation and in-distribution, where it is easiest to satisfy.

Objective. We specify AERCF, an architecture that fuses unsupervised and supervised detectors, sets thresholds by extreme-value theory (EVT), explains decisions with SHAP, hardens against adversarial perturbation, adapts its threshold from feedback, and maps scores onto a tiered containment lattice. We then ask which of these properties survive the move from the training distribution to new data and new attacks.

Methods. We implemented the detection, calibration, explanation and adaptation layers in Python and evaluated them on NSL-KDD with five seeds, using the held-out KDDTest+ (22,544 records, 17 attack types absent from training) and the harder KDDTest-21. Seven experiments measured detection under shift, EVT calibration, explanation faithfulness and stability, white-box and transferred PGD attacks, four threshold-adaptation policies on a constructed non-stationary stream, tiered containment operating points, and inference cost. The cloud-native containment layer (eBPF telemetry, pod quarantine, token revocation) is specified but not evaluated.

Results. In-distribution, the supervised detectors and the fusion reached AUROC 1.000 and detection rates above 99.9% at 1% FPR. On KDDTest+, the best detector (XGBoost) detected 81.3% of attacks while its false-positive rate rose to 3.85% against a 1% target; on KDDTest-21 it rose to 16.4%. Stacked fusion did not beat its best member (AUROC 0.958 vs 0.967) and its weights changed sign across seeds. EVT thresholds met their targets on held-out validation data but exceeded them 29-fold on KDDTest+ at $\alpha = 0.1\%$. The variational autoencoder lost only 3.5 percentage points on novel attack types, against 13.3 for XGBoost. SHAP explanations were exact, faithful (removing the top three features flipped 53.7% of detections, against 1.3% for random features) and stable across retraining (global rank correlation 0.97–0.99). Adversarial training raised detection under PGD at $\varepsilon = 0.1$ from 49.8% to 63.1%, and an OR-combination of XGBoost and the autoencoder kept 79.0% detection under transferred attacks that cut XGBoost alone to 61.0%. All three adaptive threshold policies underperformed a fixed threshold; one, a feedback controller on a 2% audit sample, drove detection to zero. At an isolation threshold calibrated for 0.1% FPR, 2.78% of benign test records would have been isolated; requiring corroboration by an unsupervised detector halved this to 1.35% while retaining 54.1% of attacks.

Conclusions. Explanation and robustness to perturbation held up; calibration, fusion and naive adaptivity did not. A credible adaptive defence must treat its own false-positive rate as unknown after deployment, prefer diverse detector families over learned fusion, and gate disruptive containment on corroboration. Code and raw results are released.

Keywords: Intrusion Detection; Distribution Shift; Extreme Value Theory; Explainable AI; SHAP; Adversarial Robustness; Adaptive Thresholds; Automated Containment; NSL-KDD.

Statement of empirical status (read before the results)

What is measured. Every table and figure in §6 reports outputs of the Python implementation released as Supplement S1, run on NSL-KDD in a two-vCPU sandbox with five random seeds. No number is simulated or estimated unless a caption says so.

What is not measured. The containment layer (§4.5) — eBPF telemetry, Kubernetes pod quarantine, token revocation and a production cluster — was not built or run; containment results in §6.6 are *operating points* computed from detector scores, not measurements of containment actions. NSL-KDD is a 1999-era derived dataset with known limitations (§8); results on it are

evidence about mechanisms, not about performance on modern networks. The adaptive-threshold stream in §6.5 is constructed by resampling real test records under a stated prevalence schedule.

The companion visualisation. The public AERCF repository contains an interactive browser visualisation of the architecture. It draws detector scores and attributions from fixed random distributions conditioned on the true label (Appendix B); it is a teaching aid, and no result in this paper is taken from it.

Who evaluated. The author designed the architecture and the experiments. Supplement S1 contains everything needed for independent replication.

I. INTRODUCTION

Machine-learning intrusion detection has a reporting problem. Classifiers trained and tested on random splits of the same capture routinely reach accuracies above 99%, and the field has long recognised that such numbers say little about behaviour on a different network, a later week, or an attack the model has not seen [3], [4]. Benchmark datasets compound the problem: NSL-KDD was itself constructed to remove redundancy that had inflated results on KDD Cup 99 [2], and later corpora have needed their own corrections [5], [44].

Architectures that promise to be *adaptive*, *explainable* and *resilient* make the problem harder rather than easier, because each property is a claim about behaviour away from the training distribution. A calibrated threshold is useful only if its false-positive rate holds on new traffic. An explanation matters most when an analyst must decide whether to trust an unexpected alert. Robustness is a claim about inputs an adversary has chosen. Adaptation is by definition a response to change. Demonstrating any of these on a random in-distribution split tests the claim where it is least likely to fail.

This paper specifies one such architecture, AERCF, and evaluates each of its claimed properties where it can fail. AERCF combines unsupervised and supervised detectors through a learned fusion, sets alert thresholds by extreme-value theory, explains supervised decisions with TreeSHAP, hardens a differentiable detector with adversarial training, adapts its threshold from feedback, and maps scores onto a tiered containment lattice whose most disruptive actions — host-level packet drop, pod quarantine, credential revocation — target cloud-native workloads. We implemented everything except the containment actions and measured it on NSL-KDD’s two held-out test sets, which contain 17 attack types absent from training.

1.1 Research questions

Table 1. Research questions.

RQ	Question	Experiment	Section
RQ1	How much detection performance survives the move from in-distribution validation to held-out test data and novel attack types, and does fusion help?	E1	§6.1
RQ2	Do extreme-value thresholds deliver their target false-positive rate after the distribution changes?	E2	§6.2
RQ3	Are SHAP explanations faithful to the model and stable across retraining?	E3	§6.3
RQ4	How much do adversarial training and detector diversity buy against feature-space PGD attacks, including transferred and adaptive ones?	E4	§6.4
RQ5	Does threshold adaptation improve on a fixed threshold when attack prevalence changes, and how can it fail?	E5	§6.5
RQ6	What do these results imply for automated containment, and what does it cost to run?	E6, E7	§6.6–6.7

1.2 Contributions

- **C1 — Architecture.** A four-layer specification (sense, detect, calibrate-and-explain, decide-and-contain) that joins anomaly-detection and supervised families, EVT calibration, SHAP rationale and a monotone containment lattice with a corroboration rule (§4).
- **C2 — Shift-first evaluation.** A reproducible implementation and a five-seed evaluation in which every property is tested on held-out and novel-attack data, with bootstrap intervals and per-category breakdowns (§5–6).

- **C3 — Negative results that matter for design.** Learned fusion that does not beat its best member and is unstable; EVT thresholds that hold in validation and fail by up to $29\times$ under shift; and three adaptive threshold policies that underperform a fixed one, including a controller that drives detection to zero (§6.1, 6.2, 6.5).
- **C4 — Positive results that also matter for design.** Faithful and stable explanations; a measured robustness gain from adversarial training; and an OR-combination of supervised and unsupervised detectors that resists transferred attacks the fusion does not (§6.3, 6.4).
- **C5 — Containment guidance.** Operating points showing why disruptive containment must be tiered and corroborated, and acceptance criteria for validating the cloud-native containment layer (§6.6, §9).

II. BACKGROUND AND RELATED WORK

2.1 Evaluation pitfalls in ML-based intrusion detection

Sommer and Paxson argued that intrusion detection is unlike other ML domains because errors are expensive, attacks are adversarial and the benign class is open-ended [3]. Arp et al. catalogued ten recurring pitfalls in security ML, among them sampling bias, data snooping, inappropriate baselines and lab-only evaluation, and found them in a large majority of the papers they reviewed [4]. Dataset critiques show that labelling and feature-extraction errors can dominate reported differences between methods [5], [44]. Distribution shift and concept drift are the general form of the problem [29]; for malware, Transcend showed that classifier confidence decays on new families and proposed statistical rejection [30], and work on dataset shift detection recommends that models fail loudly when their inputs move [31]. This paper adopts that stance: every claim is tested on data the model has not seen.

2.2 Detector families

Isolation Forest isolates anomalies with random axis-parallel splits, scoring points by expected path length [6]. Variational autoencoders model the benign distribution and score points by reconstruction probability or the negative evidence lower bound [7], [8]; ensembles of autoencoders underpin online detectors such as Kitsune [43]. Gradient-boosted trees are strong supervised baselines for tabular traffic features [9]. Temporal convolutional networks [11] suit ordered sequences such as packet or system-call streams; NSL-KDD records carry no order, so AERCF's temporal detector is not evaluated here. Stacked generalisation learns to combine base models on held-out predictions [10].

2.3 Thresholds from extreme-value theory

The Pickands–Balkema–de Haan theorem states that exceedances over a high threshold u are approximately generalised-Pareto distributed [12], [13]. Fitting the GPD to the tail of benign scores yields a threshold for any target false-positive rate α , including rates too small for empirical quantiles to estimate reliably; SPOT applies this to streaming anomaly detection [14]. The guarantee is conditional on the fitted tail being the tail of future benign data.

2.4 Explanation

SHAP attributes a prediction to features using Shapley values [15]; TreeSHAP computes them exactly and in polynomial time for tree ensembles [16]. LIME [17] and layer-wise relevance propagation [18] are alternatives. Because many explanation methods can produce plausible but unfaithful attributions, removal-based evaluation such as ROAR [19] and security-specific criteria — descriptive accuracy, sparsity, stability, efficiency [20] — are needed to establish that an explanation reflects the model.

2.5 Adversarial robustness

Gradient-based attacks [21], [22], [23] perturb inputs within a norm ball; projected gradient descent (PGD) with adversarial training remains the standard empirical defence [22]. Defences that appear robust often fail under adaptive attacks designed with knowledge of the defence [24], [25]. In network security, feature-space attacks are an upper bound on what an attacker can achieve, because real traffic must satisfy protocol and semantic constraints [26], and practitioners argue that real attackers rarely compute gradients at all [27]. Randomised smoothing gives certified radii for smoothed classifiers [28]; we do not evaluate certification here.

2.6 Adaptation and automated response

Reinforcement-learning formulations treat threshold or response selection as a sequential decision problem, often a POMDP [33], [34]. Two features of intrusion detection make this delicate: feedback (labels) is sparse and delayed, and the environment includes an adversary who can change prevalence. Classical control offers a sobering baseline: integral feedback on a noisy, biased measurement winds up [35]. On the response side, zero-trust architecture requires per-request authorisation and continuous evaluation [37], the NIST Cybersecurity Framework 2.0 treats detection and response as continuous functions rather than periodic audits [38], and adaptive risk-management frameworks propose context-dependent responses [1]. Shielding constrains a learned policy with a verified safety envelope [36]; AERCF's containment lattice applies the same idea to response actions.

III. PROBLEM FORMULATION

Let $x \in [0,1]^d$ be a normalised flow record ($d = 121$ after encoding) with label $y \in \{0, 1\}$. A detector produces a score $s(x)$, larger meaning more suspicious. For a target false-positive rate α , a threshold θ is chosen on benign validation scores so that

$$P(s(x) \geq \theta \mid y = 0) = \alpha \quad (1)$$

Deployment requires that Eq. (1) hold for future benign traffic, whose distribution $P_t(x \mid y = 0)$ may differ from the validation distribution P_0 . The **realised calibration ratio** $\rho = \text{FPR}_t / \alpha$ measures how far it fails; $\rho = 1$ is perfect. Under EVT, the threshold for α is

$$\theta_{\alpha} = u + (\sigma / \xi) \cdot [(n \alpha / N_u)^{(-\xi)} - 1] \quad (2)$$

where u is the 90th percentile of fitted benign scores, N_u the number of exceedances among n scores, and (ξ, σ) the fitted GPD shape and scale. For probability-valued scores we fit on the logit scale.

Containment uses a lattice of tiers $T = \{\text{monitor} < \text{step-up} < \text{isolate}\}$ with thresholds $\theta_1 < \theta_2$ chosen for $\alpha_1 = 1\%$ and $\alpha_2 = 0.1\%$. A **corroboration rule** permits *isolate* only if at least one detector from a different family (here, an unsupervised detector) also fires at its own 1% threshold:

$$\text{tier}(x) = \text{isolate} \text{ iff } s_F(x) \geq \theta_2 \wedge \exists k \in U : s_k(x) \geq \theta_k \quad (3)$$

As with any monotone lattice, a learned or rule-based component may escalate a tier but cannot lower one assigned by a deterministic rule, so a verified envelope bounds the behaviour of learned parts [36].

IV. THE AERCF ARCHITECTURE

Figure 1 shows the four layers. Solid boxes were implemented and evaluated; dashed boxes are specified for future validation.

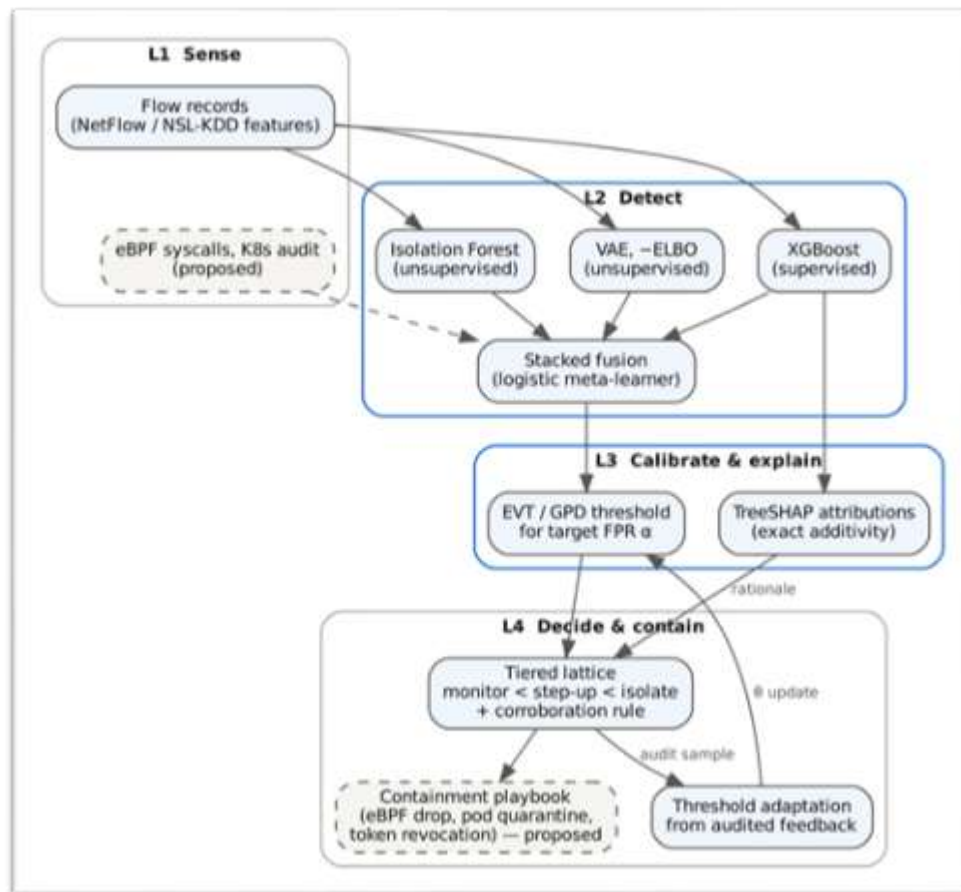


Figure 1. AERCF architecture. L1–L3 and the decision logic of L4 were implemented and evaluated on NSL-KDD. BPF/Kubernetes telemetry and the containment playbook (dashed) are specified in §4.5 but were not built or measured.

4.1 L1 — Sense

AERCF consumes flow-level features. For this evaluation they are the 41 NSL-KDD features: heavy-tailed counts and byte volumes are log-transformed, the three categorical fields are one-hot encoded (121 dimensions in total) and all features are min-max scaled on the training split. The cloud-native variant adds kernel telemetry from eBPF probes on process-execution and namespace-transition system calls and Kubernetes audit events [39], [40]; these are not used here.

4.2 L2 — Detect

Table 2. Detectors as implemented.

Detector	Family	Trained on	Score	Configuration
Isolation Forest [6]	Unsupervised	Benign training records	Negated sklearn score (higher = more anomalous)	200 trees
VAE [7], [8]	Unsupervised	Benign training records	Negative ELBO, 8 Monte-Carlo samples	121–64–32–8 encoder, mirrored decoder, 25 epochs
XGBoost [9]	Supervised	All training records	Attack probability	300 trees, depth 6, $\eta = 0.1$
MLP	Supervised	All training records	Attack probability	121–128–64–1, 15 epochs; also a PGD-trained variant ($\epsilon = 0.05$, 7 steps)
Fusion [10]	Stacked	Validation split	Logistic regression on standardised IF, VAE and XGBoost scores	—

The temporal convolutional detector of the original design was omitted because NSL-KDD records are unordered; the MLP is included as the differentiable model needed for white-box attacks.

4.3 L3 — Calibrate and explain

Thresholds are set either by the empirical benign quantile or by Eq. (2). Explanations are TreeSHAP attributions for the XGBoost model, which satisfy additivity — attributions plus the base value equal the model’s log-odds output — exactly [16].

4.4 L4 — Decide and adapt

Scores map to the tier lattice of Eq. (3). Threshold adaptation is evaluated with four policies (§5, E5): a fixed threshold; a port of the heuristic controller in the companion visualisation, which nudges θ according to cumulative true-positive and false-positive rates computed from *true labels*; a proportional–integral controller that estimates FPR from a 2% random audit sample whose labels arrive one window late; and an alert-budget policy that sets θ to pass at most 5% of the previous window’s traffic.

4.5 Containment layer (specified, not evaluated)

For cloud-native workloads, AERCF specifies a containment playbook keyed to the tier lattice: *step-up* raises logging and requires re-authentication; *isolate* applies, in order, an egress drop on the host via an eBPF program, a quarantine label that triggers an isolating Kubernetes NetworkPolicy, and revocation of the workload’s service-account token. The motivating threat is a container escape through a kernel vulnerability, of which CVE-2022-0492 (cgroups release_agent) and CVE-2022-0847 (“Dirty Pipe”) are documented examples [41], [42]. Each action, its SHAP rationale and the triggering scores are written to an append-only, hash-chained audit record. No part of this layer was deployed for this paper; §9 states the criteria under which it should be validated.

V. EXPERIMENTAL DESIGN

5.1 Data

Table 3. NSL-KDD partitions used [2].

Partition	Records	Benign	Attack	Notes
KDDTrain+	125,973	67,343	58,630	Split 80/20 (stratified by attack type) into training and validation per seed
KDDTest+	22,544	9,711	12,833	3,750 attack records (29.2%) from 17 types absent from training
KDDTest-21	11,850	2,152	9,698	Records that 21 reference classifiers found hardest; 3,740 novel-type records

5.2 Protocol

All models were trained with five seeds (0–4), each with its own train/validation split. Unless stated otherwise, thresholds were set on benign validation records for $\alpha = 1\%$, and results are mean $\pm$ standard deviation over seeds. For seed 0 we report 95% bootstrap intervals (1,000 resamples of KDDTest+). Experiments E3–E7 use the seed-0 models.

Table 4. Experiments.

ID	Measures	Design
E1	Detection under shift	AUROC, AUPRC, detection rate (DR), FPR, precision and F1 on validation, KDDTest+ and KDDTest-21; DR on seen vs novel attack types and by category
E2	EVT calibration	GPD fitted to half the benign validation scores ($u = 90$ th percentile); realised FPR on the other half and on KDDTest+ benign records, for $\alpha \in \{1\%, 0.5\%, 0.1\%\}$; empirical quantile as comparator
E3	Explanation	TreeSHAP on 2,000 test attacks: additivity error; deletion faithfulness (replace top-k features by the benign training median, $k \in \{1, 3, 5, 10\}$, against k random features); stability across four bootstrap-retrained models
E4	Adversarial robustness	L_∞ PGD-20 on 4,000 test attacks, continuous features only, clipped to $[0, 1]$, $\epsilon \in \{0.01, 0.02, 0.05, 0.1\}$; white-box against MLP and PGD-trained MLP; transfer to all detectors and to OR-ensembles; an adaptive attack that also minimises VAE reconstruction error
E5	Threshold adaptation	60 windows $\times$ 1,000 records resampled from KDDTest+ with attack prevalence rising from 5% to 60% and back; four policies (§4.4) on fused scores
E6	Containment operating points	Tier shares for benign and attack records at $\alpha_1 = 1\%$, $\alpha_2 = 0.1\%$, with and without corroboration (Eq. 3)
E7	Cost	Training time and per-record inference latency at batch sizes 1 and 1,024 on 2 vCPUs

VI. RESULTS

6.1 RQ1 — Detection collapses at the distribution boundary, and fusion does not help

On in-distribution validation data every supervised model looked nearly perfect (Table 5): XGBoost reached AUROC 1.0000 and detected 99.98% of attacks at a 1% false-positive rate, and the fusion 99.96%. These are the conditions under which the 98–99% figures common in the literature are obtained.

Table 5. Detection at the validation 1% FPR threshold (mean $\pm$ SD, five seeds).

Detector	Validation AUROC	KDDTest+ AUROC	KDDTest+ DR	KDDTest+ FPR	KDDTest-21 DR	KDDTest-21 FPR
Isolation Forest	0.982 $\pm$ 0.002	0.948 $\pm$ 0.001	56.2 $\pm$ 0.9%	1.37 $\pm$ 0.37%	42.5 $\pm$ 1.2%	6.2 $\pm$ 1.7%
VAE	0.990 $\pm$ 0.001	0.968 $\pm$ 0.002	59.6 $\pm$ 1.1%	1.37 $\pm$ 0.05%	47.0 $\pm$ 1.5%	6.0 $\pm$ 0.2%
XGBoost	1.000 $\pm$ 0.000	0.967 $\pm$ 0.002	81.3 $\pm$ 0.7%	3.85 $\pm$ 0.14%	75.3 $\pm$ 0.9%	16.4 $\pm$ 0.6%
MLP	1.000 $\pm$ 0.000	0.912 $\pm$ 0.017	76.4 $\pm$ 0.6%	7.20 $\pm$ 1.62%	68.8 $\pm$ 0.8%	31.6 $\pm$ 7.1%
Fusion (stacked)	1.000 $\pm$ 0.000	0.958 $\pm$ 0.019	77.1 $\pm$ 1.7%	3.28 $\pm$ 0.07%	69.7 $\pm$ 2.3%	14.6 $\pm$ 0.4%

On KDDTest+ the picture changed in two ways at once (Figure 2). Detection fell — XGBoost to 81.3%, the unsupervised detectors to 56–60% — and the false-positive rate, which had been set to 1%, rose to 1.37% for the unsupervised detectors and to 3.85% for XGBoost. On KDDTest-21 XGBoost’s false-positive rate was 16.4%. The bootstrap intervals for seed 0 are narrow (XGBoost AUROC 0.966–0.971; fusion DR 75.4–76.9%, FPR 2.8–3.5%), so these are not sampling artefacts.

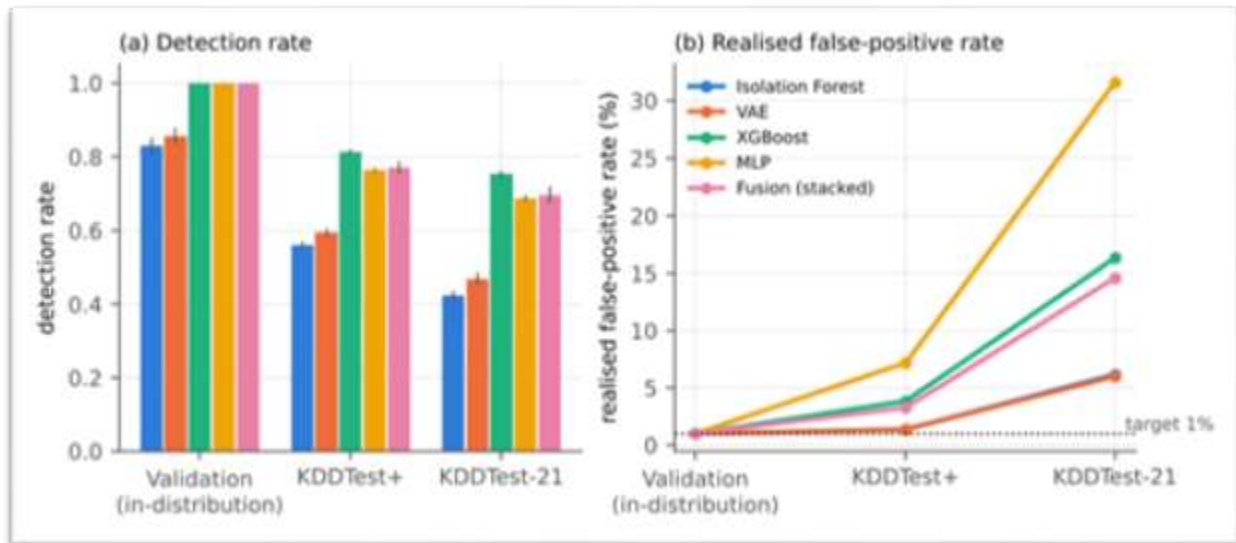


Figure 2. Detection rate (a) and realised false-positive rate (b) at a threshold set for 1% FPR on in-distribution validation data, five seeds (error bars: SD). Colours identify detectors in both panels. The calibration that holds on validation data fails on both held-out test sets, most severely for the supervised models.

Fusion did not help. The stacked fusion was worse than XGBoost alone on every test metric and markedly less stable (AUROC SD 0.019 against 0.002). Its learned weights explain why: fitted on validation data where XGBoost is already perfect, the meta-learner had no signal with which to value the unsupervised detectors, and the Isolation Forest weight ranged from -0.59 to $+0.71$ across seeds. Stacking learns how to combine detectors on the distribution it is fitted to, which is precisely the distribution on which the unsupervised detectors are not needed.

Novel attacks and categories. Figure 3a compares detection of attack types seen in training with the 17 novel types. The VAE lost only 3.5 percentage points ($60.6\% \rightarrow 57.1\%$), whereas XGBoost lost 13.3, the fusion 18.3 and the MLP 27.5. By AUROC, the unsupervised detectors ranked novel attacks as well as seen ones (Isolation Forest 0.952 novel vs 0.947 seen). Figure 3b shows the weakness that every detector shares: remote-to-local (R2L) attacks, which resemble legitimate sessions at the flow level, were detected at 3–4% by the unsupervised detectors and at best 33% by XGBoost.

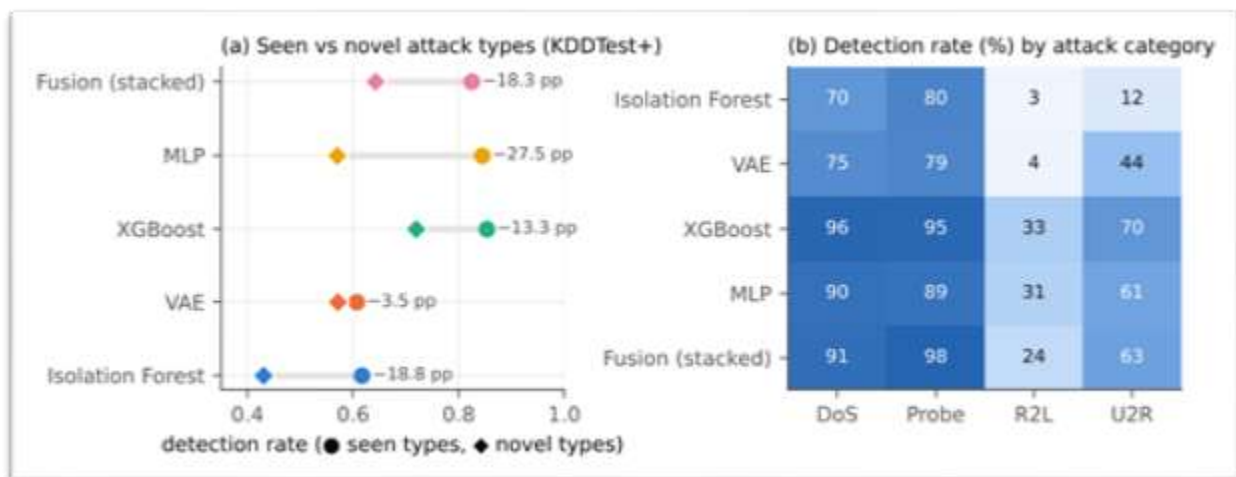


Figure 3. (a) Detection rate on attack types seen in training (circles) and on the 17 novel types (diamonds), KDDTest+, mean of five seeds. (b) Detection rate by attack category. The VAE is the most stable under novelty; no detector handles R2L attacks well.

6.2 RQ2 — EVT thresholds are calibrated in validation, not in deployment

Figure 4 reports the realised calibration ratio ρ for every detector, target and seed. On held-out validation data, EVT thresholds were close to their targets (seed means of ρ between 0.79 and 1.07 for $\alpha \geq 0.5\%$, and 0.86–1.31 at $\alpha = 0.1\%$), and comparable to

empirical quantiles, as theory predicts when the tail is stationary. On KDDTest+ they were not. At $\alpha = 1\%$ the ratio was 1.19 for Isolation Forest, 1.42 for the VAE, 3.27 for the fusion and 3.77 for XGBoost; at $\alpha = 0.1\%$ it reached 29.1 for the fusion and 29.4 for XGBoost. EVT and the empirical quantile failed together (fusion at $\alpha = 0.1\%$: $29.1\times$ and $28.8\times$), because the failure is not in estimating the tail but in assuming that the deployment tail is the validation tail.

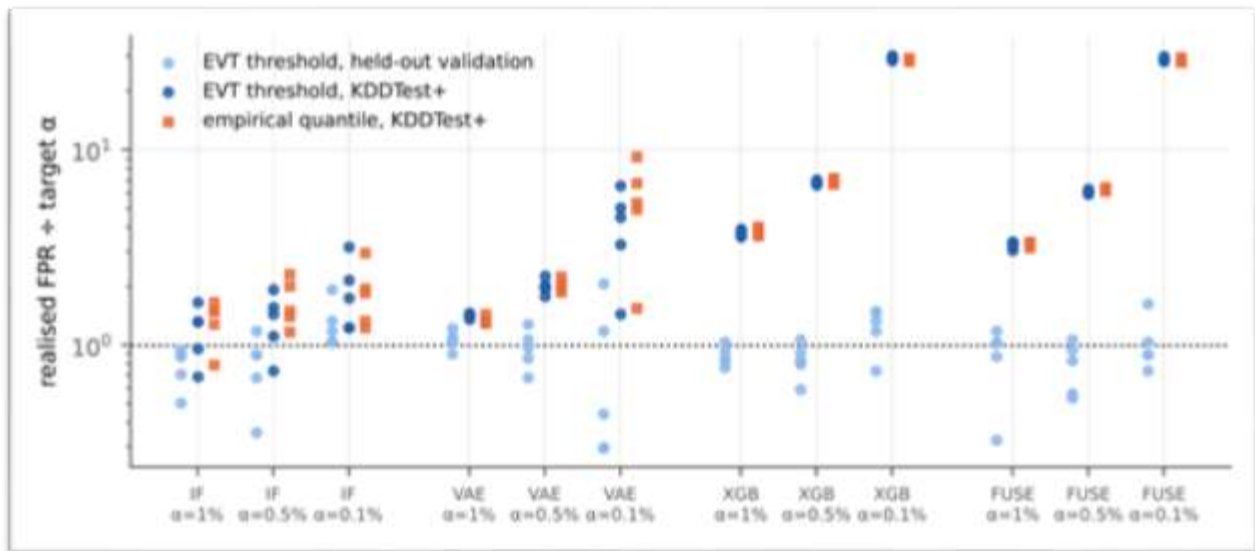


Figure 4. Realised FPR divided by target α (log scale) for EVT thresholds on held-out validation data (light), EVT thresholds on KDDTest+ (dark), and empirical-quantile thresholds on KDDTest+ (orange); one point per seed. The dotted line is perfect calibration. Under shift, supervised detectors miss their target by up to $30\times$, and EVT offers no protection. The fitted shape parameters are informative in their own right: $\xi \approx -0.20$ for Isolation Forest (a bounded tail, as expected for path-length scores), $+0.15$ for the VAE and $+0.36$ for the fusion on the logit scale (heavy tails). The detectors with the lightest tails were also the best calibrated under shift.

6.3 RQ3 — Explanations are faithful and stable

TreeSHAP attributions were exactly additive (maximum absolute error 9.5×10^{-6} in log-odds over 2,000 explained attacks). They were also faithful (Figure 5b): replacing the single most-attributed feature with its benign median flipped 17.5% of detections to benign, and the top three flipped 53.7%, against 0.35% and 1.3% for the same number of random features; the mean drop in attack probability for the top three was 0.661, against 0.008. Faithfulness saturated at five features (80.7% flipped), consistent with a model whose decisions rest on a handful of volume and connection-rate features (Figure 5a).

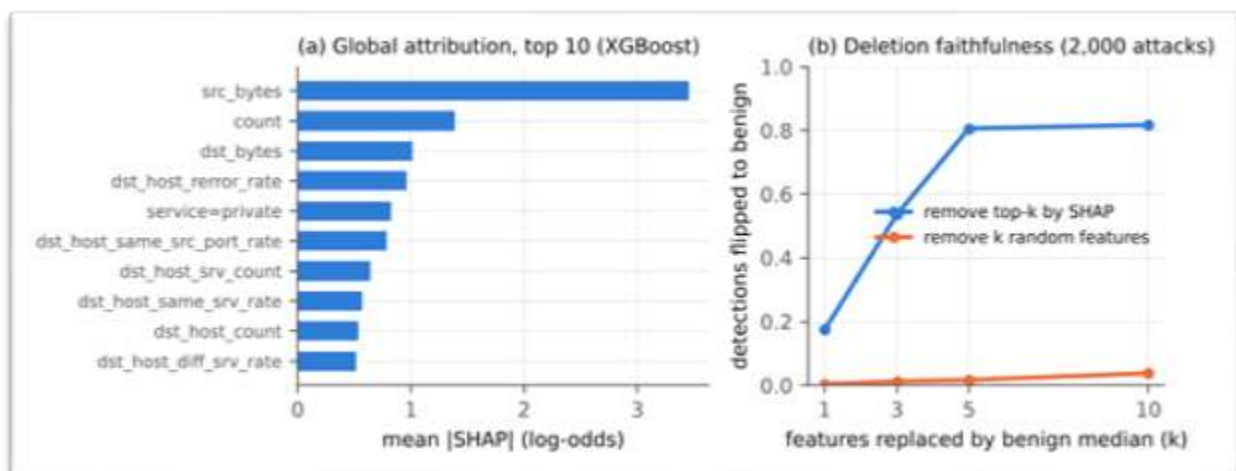


Figure 5. (a) Mean absolute SHAP value of the ten most influential features for XGBoost. (b) Share of detected attacks flipped to benign when the k most-attributed features (blue) or k random features (orange) are replaced by their benign training median.

Across four models retrained on bootstrap resamples, the global feature ranking was stable (Spearman ρ 0.966–0.990; top-10 Jaccard overlap 0.67–1.00) and per-record rankings moderately so (mean Spearman 0.76–0.87). Explanations, in short, were the component of AERCF that held up best. That is not the same as being *useful* to an analyst — an attribution to `src_bytes` and `count` is faithful but generic — which is a question for a user study (§9).

6.4 RQ4 — Adversarial training helps; diversity helps more than fusion

Under white-box PGD-20 (Figure 6a), the standard MLP's detection rate fell from 74.6% to 49.8% at $\epsilon = 0.1$. The PGD-trained MLP started lower (71.5%; clean test AUROC 0.915 vs 0.892) but kept 63.1% at $\epsilon = 0.1$, a gain of 13.3 points at a clean cost of 3.1.

Transferred attacks (Figure 6b) produced a result with design consequences. Perturbations crafted against the MLP cut XGBoost's detection from 81.3% to 61.0% at $\epsilon = 0.1$ and the fusion's from 75.6% to 66.7%, but they *raised* the VAE's detection from 59.9% to 85.0% and Isolation Forest's from 54.0% to 63.7%: pushing a record across a supervised decision boundary made it less like benign traffic. An OR-combination of XGBoost and the VAE, with thresholds jointly set for 1% validation FPR, therefore lost almost nothing (78.9% $\rightarrow$ 79.0%), at a clean cost of 2.2 points (79.5% vs 81.7%) and similar test FPR (3.55% vs 3.69%). The stacked fusion, which had learned to discount the VAE, did not inherit this property.

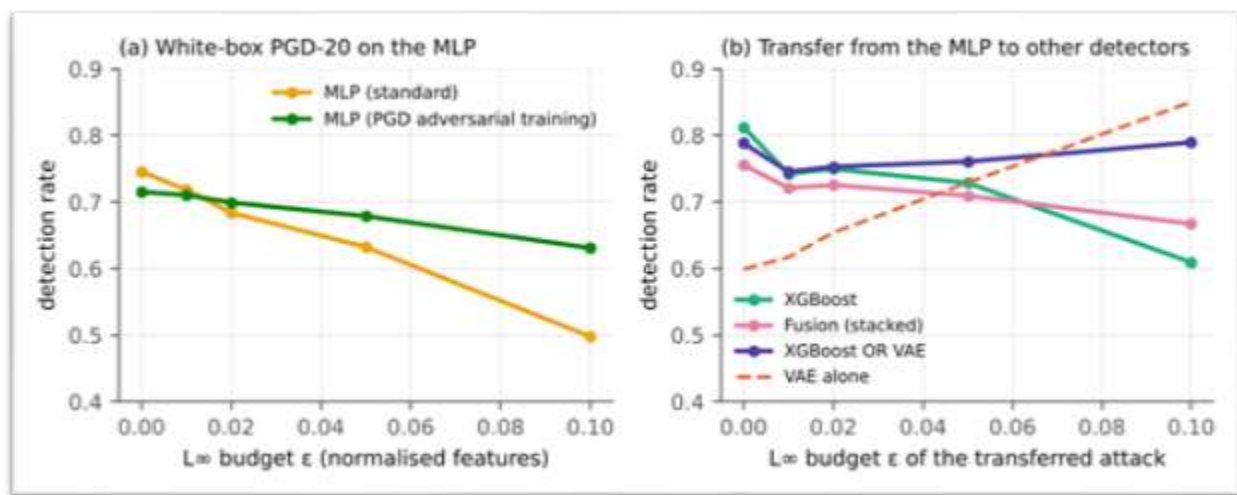


Figure 6. (a) Detection under white-box PGD-20 for a standard and an adversarially trained MLP. (b) Detection under PGD perturbations crafted on the standard MLP and transferred to other detectors; 4,000 KDDTest+ attacks, seed 0. Feature-space attacks are an upper bound on realisable traffic perturbations [26].

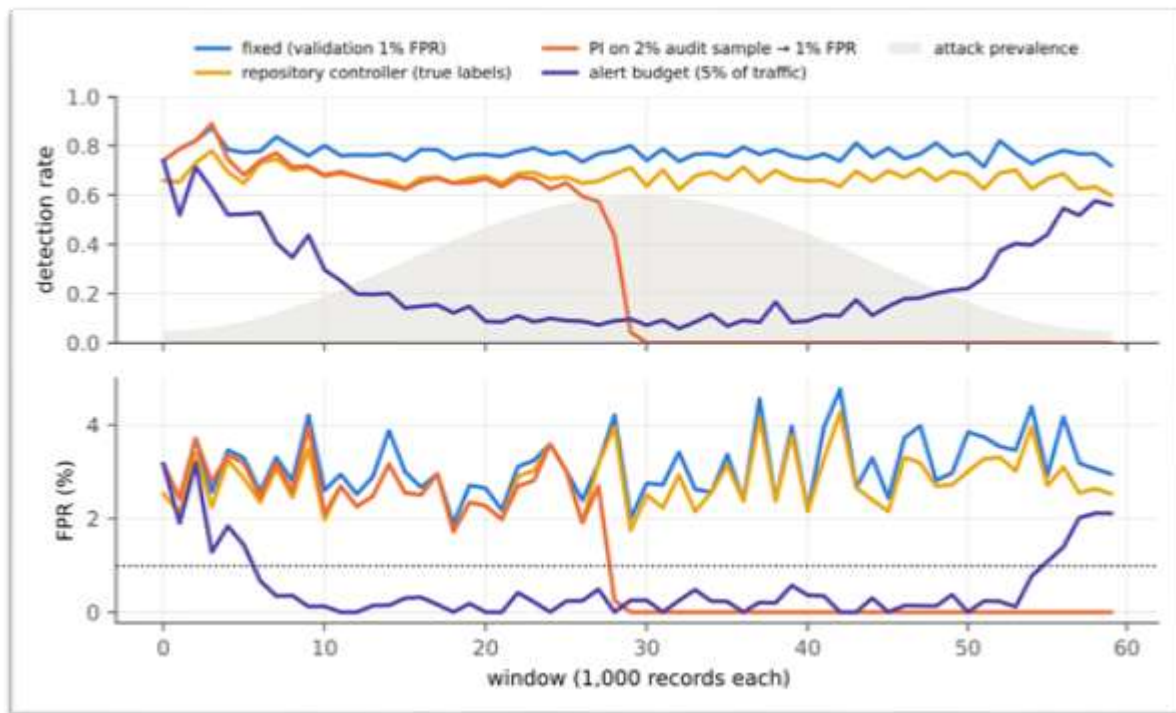
Because transfer results can flatter a defence [25], we also ran an adaptive attack that jointly lowers the MLP's logit and the VAE's reconstruction error (weight $\lambda \in \{0.05, 0.2\}$, $\epsilon = 0.05$). The OR-ensemble kept 75.4–75.8% detection and XGBoost 72.6–73.0%. This is one adaptive formulation, not a robustness guarantee; a stronger attacker optimising directly against the ensemble decision may do better.

6.5 RQ5 — Naive adaptation underperforms a fixed threshold

Figure 7 and Table 6 compare the four threshold policies on a stream whose attack prevalence rises from 5% to 60% and falls back. The **fixed** threshold was the best policy for detection (mean 77.2%) at a stable but miscalibrated false-positive rate (mean 3.13%). The **repository controller**, despite having access to true labels, lowered detection to 67.5% for a small FPR gain, because its update rule responds to cumulative rates rather than to the current window. The **PI controller on a 2% audit sample** tracked the 1% target until prevalence rose, when the audit sample contained too few benign records to estimate FPR, the integrator wound up, and the threshold climbed until detection was zero from window 30 onward. The **alert budget** kept alerts within capacity by suppressing detection exactly when attacks were most frequent (minimum 5.8%) — a policy an adversary can exploit by raising prevalence deliberately.

Table 6. Threshold adaptation over 60 windows (fused scores, seed 0).

Policy	Feedback used	Mean DR	Min DR	Mean FPR	Max FPR
Fixed (validation 1% FPR)	None	77.2%	71.5%	3.13%	4.75%
Repository controller	True labels of every record (oracle)	67.5%	60.0%	2.78%	4.26%
PI on 2% audit sample	20 labelled records per window, one window late	33.0%	0.0%	1.28%	3.96%
Alert budget (5%)	None (alert volume only)	24.8%	5.8%	0.53%	3.18%

**Figure 7.** Per-window detection rate (top) and FPR (bottom) for four threshold policies on a stream resampled from KDDTest+ with attack prevalence (grey) rising from 5% to 60% and back. The dotted line marks the 1% FPR target.

The lesson is not that adaptation is futile but that an adaptive policy optimises exactly the signal it is given. Under shift, reaching a nominal FPR target requires a threshold high enough to discard most attacks; a controller that pursues the target faithfully will discard them. A sound controller needs anti-windup, a floor on detection or a ceiling on threshold movement, prevalence-aware FPR estimation (stratified audit sampling), and an alarm when the estimated FPR and the observed alert rate disagree.

6.6 RQ6 — Containment must be tiered and corroborated

With the tier thresholds of Eq. (3) set for 1% and 0.1% validation FPR, 66.7% of KDDTest+ attacks reached *isolate*, 9.5% *step-up* and 23.8% remained at *monitor* (Figure 8). But 2.78% of benign records also reached *isolate* — 28× the 0.1% target. Requiring corroboration by Isolation Forest or the VAE halved benign isolation to 1.35% while still isolating 54.1% of attacks. By category, isolation reached 87.6% of DoS and 74.3% of probe records but only 7.2% of R2L and 20.9% of U2R records — the low-volume attacks that most often precede a container escape or credential theft.

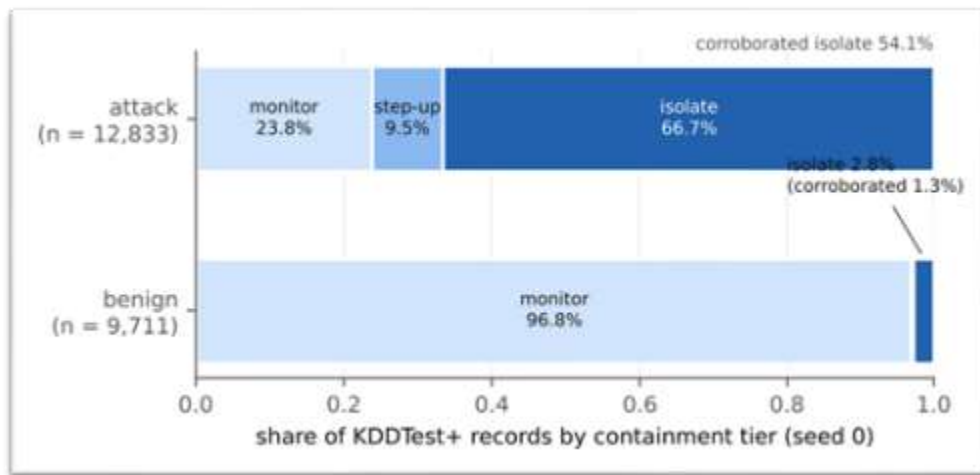


Figure 8. Share of KDDTest+ benign and attack records assigned to each containment tier by the fused score (seed 0), with the effect of the corroboration rule on the isolate tier.

In an environment that evaluates a million benign events a day, a 1.35% isolation rate would mean thousands of false quarantines. The measured numbers therefore argue against automatic isolation driven by a single fused score, and for a design in which *isolate* requires agreement between detector families plus either a high-confidence deterministic signal (for example, a namespace-escape system call from a web container) or human approval. This is a projection from flow-level detectors, not a measurement of the containment layer.

6.7 Cost

Table 7. Training time and inference latency on 2 vCPUs (seed 0).

Component	Training (s)	Latency, batch 1 (ms/record)	Latency, batch 1,024 (ms/record)
Isolation Forest	0.3	11.37	0.018
VAE	11.6	0.58	0.009
XGBoost	3.0	0.32	0.002
MLP / PGD-trained MLP	6.5 / 27.8	0.05	0.0003
Full fusion (IF + VAE + XGBoost + stacker)	—	14.11	0.030 ($\approx$ 33,000 records/s)
TreeSHAP explanation (XGBoost)	—	0.90	0.148

Per-record latency at batch size 1 is dominated by fixed call overhead in the Isolation Forest implementation; in batches the full pipeline scored about 33,000 records per second and explained about 6,700 per second on two vCPUs, so explaining every alert is affordable.

VII. DISCUSSION

7.1 Treat post-deployment FPR as unknown

The most consequential finding is that no threshold method — empirical or extreme-value — carried its false-positive rate across the distribution boundary, and that the miscalibration was largest for the most accurate models. A deployed detector should therefore report an *estimated* FPR with uncertainty from continuing audit samples, raise an alarm when it drifts, and treat validation calibration as an initial condition. Conformal methods [32] give finite-sample guarantees only under exchangeability, which shift breaks for the same reason it breaks EVT; their value here is as drift detectors, in the spirit of rejecting predictions when non-conformity rises [30], [31].

7.2 Prefer diversity to learned fusion

Stacking optimised on in-distribution data learned to ignore the detectors that mattered under novelty and perturbation. A simple OR over families, with jointly calibrated thresholds, preserved both properties at a small clean cost. The general principle is that a combiner should be fitted on data that represents the conditions it must survive — or should not be fitted at all.

7.3 Adaptation needs control-theoretic safeguards

The repository controller's use of true labels for every record is an oracle unavailable in practice; the realistic controllers either wound up or were exploitable. Any learned policy — including the DQN/POMDP formulation in the original design [33], [34] — inherits these hazards and should be evaluated against delayed, sparse, prevalence-biased feedback and an adversary who controls prevalence, with a shield that bounds threshold movement and prevents detection from falling below a floor [36].

7.4 Explanations are necessary, not sufficient

SHAP explanations were exact, faithful and stable, which justifies using them as the rationale attached to every containment action. They explain the model, not the attack: under shift, a faithful explanation of a false positive is still a false positive. Their operational value — whether they help analysts reject false alarms faster — remains to be measured.

VIII. THREATS TO VALIDITY ANND LIMITATIONS

- **Dataset.** NSL-KDD derives from 1998–99 DARPA traffic and is known to differ from modern networks [2], [44]. We chose it because its official test sets contain novel attack types and a hardness-stratified subset, which suits a shift-first evaluation. Results should be replicated on modern flow corpora such as UNSW-NB15 and CIC-IDS2017, with the latter's known labelling corrections applied [5].
- **Constructed stream.** The adaptation stream resamples real records under a chosen prevalence schedule; real streams have temporal structure, bursts and label delays that it does not capture.
- **Feature-space attacks.** PGD perturbations respect only box and categorical constraints, not protocol semantics, and may not be realisable as traffic [26], [27]. They bound attacker capability from above.
- **Single configuration per model.** Hyper-parameters were fixed a priori and not tuned on test data; different architectures could shift absolute numbers. The temporal detector of the original design was not evaluated.
- **Containment not built.** The eBPF and Kubernetes layer was specified only. No claim is made about containment latency, overhead or effectiveness.
- **Author background.** The author has seven years of operational IT and security engineering experience and an MSc in Data Science (University of East London, 2025); this work has not yet been peer-reviewed. Operational experience shaped the threat scenarios and the emphasis on false-positive cost, but independent replication with the released code is the appropriate check.

IX. ROADMAP AND FALSIFIABLE ACCEPTANCE CRITERIA

Table 8 lists the criteria the next stages of AERCF should meet. Values are **design targets**, not results.

Table 8. Acceptance criteria for future validation (design targets; not yet measured).

Stage	Criterion	Target	Refuted if
Modern data	Replicate E1–E6 on UNSW-NB15 and corrected CIC-IDS2017, cross-dataset (train on one, test on another)	Same qualitative ordering of findings	Fusion beats best member under shift, or EVT holds $p \leq 1.5$
Calibration monitoring	Online FPR estimate from stratified audit sampling with interval	Alarm within 2 windows when $p > 2$	Undetected $p > 2$ for > 5 windows
Adaptation	Controller with anti-windup, detection floor and prevalence-aware estimation	Mean DR $\geq$ fixed threshold; no window with DR $< 50\%$ of fixed	Any collapse to DR ≈ 0
Robustness	Adaptive attack against the OR-ensemble decision itself; problem-space constrained attacks	DR drop ≤ 10 pp at the realisable budget	Drop > 10 pp
Containment (testbed)	eBPF/Kubernetes layer on a test cluster with scripted container-escape scenarios	Benign quarantine $\leq 0.1\%$ of workloads per day; every action audited	Any unaudited action; quarantine rate above target
Analyst study	SHAP rationale vs none for alert triage	Faster correct dismissal of false positives	No difference or slower

X. CONCLUSION

Measured where it can fail, AERCF's promises divide cleanly. Explanation and perturbation robustness held: SHAP attributions were exact, faithful and stable, adversarial training bought real robustness, and pairing a supervised detector with an unsupervised one resisted transferred attacks. Calibration, learned fusion and naive adaptation did not: thresholds tuned for a 1% false-positive rate delivered 3–16% on new data, the fusion was worse than its best member, and the adaptive policies either cost detection or collapsed it. For a framework whose final layer quarantines workloads, those failures are decisive. The design that follows from the evidence is one in which post-deployment calibration is monitored rather than assumed, detector families are combined for diversity rather than fitted for accuracy, adaptation is shielded, and disruptive containment requires corroboration.

DECLARATIONS

Data and code availability. NSL-KDD is publicly available [2]. Supplement S1 contains the implementation (`aercf_core.py`), the experiments (`experiments.py`, `e4b_ensemble.py`), the simulator audit (`sim_audit.py`), the figure script and the raw JSON outputs from which every number in this paper is taken.

Funding. None. **Competing interests.** None. **Ethics.** Public, de-identified benchmark data only.

Use of AI tools. An AI assistant (Claude, Anthropic) was used to help implement the experiments, draft the manuscript and prepare the figures. The author designed the architecture and experiments, reviewed all code and output, and takes full responsibility for the content.

Author contributions (CRediT). O. A. Agunloye: conceptualisation, methodology, software, investigation, formal analysis, visualisation, writing — original draft, writing — review and editing.

REFERENCES

- [1]. H. M. Melaku, "Context-based and adaptive cybersecurity risk management framework," *Risks*, vol. 11, no. 6, art. 101, 2023.
- [2]. M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, pp. 1–6.
- [3]. R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Security and Privacy*, 2010, pp. 305–316.
- [4]. D. Arp et al., "Dos and don'ts of machine learning in computer security," in *Proc. 31st USENIX Security Symp.*, 2022.
- [5]. G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: The CICIDS2017 case study," in *Proc. IEEE Security and Privacy Workshops (SPW)*, 2021.
- [6]. F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, 2008, pp. 413–422.
- [7]. D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2014.
- [8]. J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *SNU Data Mining Center, Tech. Rep.*, 2015.
- [9]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [10]. D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.
- [11]. S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.
- [12]. J. Pickands III, "Statistical inference using extreme order statistics," *Ann. Statist.*, vol. 3, no. 1, pp. 119–131, 1975.
- [13]. S. Coles, *An Introduction to Statistical Modeling of Extreme Values*. London: Springer, 2001.
- [14]. A. Siffer, P.-A. Fouque, A. Termier, and C. Largouët, "Anomaly detection in streams with extreme value theory," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2017, pp. 1067–1075.
- [15]. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems* 30, 2017.
- [16]. S. M. Lundberg et al., "From local explanations to global understanding with explainable AI for trees," *Nature Machine Intelligence*, vol. 2, pp. 56–67, 2020.
- [17]. M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [18]. S. Bach et al., "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, e0130140, 2015.

- [19] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, “A benchmark for interpretability methods in deep neural networks,” in *Advances in Neural Information Processing Systems* 32, 2019.
- [20]. A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, “Evaluating explanation methods for deep learning in security,” in *Proc. IEEE European Symp. Security and Privacy (EuroS&P)*, 2020, pp. 158–174.
- [21]. I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proc. ICLR*, 2015.
- [22] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *Proc. ICLR*, 2018.
- [23]. N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *Proc. IEEE Symp. Security and Privacy*, 2017, pp. 39–57.
- [24]. A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *Proc. 35th Int. Conf. Machine Learning (ICML)*, 2018, pp. 274–283.
- [25]. F. Tramèr, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” in *Advances in Neural Information Processing Systems* 33, 2020.
- [26] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, “Intriguing properties of adversarial ML attacks in the problemspace,” in *Proc. IEEE Symp. Security and Privacy*, 2020, pp. 1332–1349.
- [27]. G. Apruzzese et al., ““Real attackers don’t compute gradients’: Bridging the gap between adversarial ML research and practice,” in *Proc. IEEE Conf. Secure and Trustworthy Machine Learning (SaTML)*, 2023.
- [28]. J. Cohen, E. Rosenfeld, and J. Z. Kolter, “Certified adversarial robustness via randomized smoothing,” in *Proc. 36th ICML*, 2019, pp. 1310–1320.
- [29]. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, art. 44, 2014.
- [30]. R. Jordaney et al., “Transcend: Detecting concept drift in malware classification models,” in *Proc. 26th USENIX Security Symp.*, 2017, pp. 625–642.
- [31]. S. Rabanser, S. Günnemann, and Z. C. Lipton, “Failing loudly: An empirical study of methods for detecting dataset shift,” in *Advances in Neural Information Processing Systems* 32, 2019.
- [32]. A. N. Angelopoulos and S. Bates, “Conformal prediction: A gentle introduction,” *Foundations and Trends in Machine Learning*, vol. 16, no. 4, pp. 494–591, 2023.
- [33]. V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 2015.
- [34]. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.
- [35]. K. J. Åström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ: Princeton Univ. Press, 2008.
- [36]. M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in *Proc. AAAI Conf. Artificial Intelligence*, 2018, pp. 2669–2678.
- [37]. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero trust architecture,” *NIST SP 800-207*, 2020, doi: 10.6028/NIST.SP.800-207.
- [38]. National Institute of Standards and Technology, “The NIST Cybersecurity Framework (CSF) 2.0,” *NIST CSWP 29*, 2024, doi: 10.6028/NIST.CSWP.29.
- [39]. B. Gregg, *BPF Performance Tools: Linux System and Application Observability*. Boston, MA: Addison-Wesley, 2019.
- [40]. The Kubernetes Authors, “Network policies” and “Pod security standards,” *Kubernetes documentation*. [Online]. Available: <https://kubernetes.io/docs/>
- [41]. NIST National Vulnerability Database, “CVE-2022-0492,” 2022. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2022-0492>
- [42]. NIST National Vulnerability Database, “CVE-2022-0847,” 2022. [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2022-0847>
- [43]. Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An ensemble of autoencoders for online network intrusion detection,” in *Proc. Network and Distributed System Security Symp. (NDSS)*, 2018.
- [44]. M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” *Computers & Security*, vol. 86, pp. 147–167, 2019.

APPENDIX A. REPRODUCTION

```

pip install numpy pandas scikit-learn xgboost torch shap scipy matplotlib
git clone https://github.com/defcom17/NSL_KDD nslkdd # or the official NSL-KDD release
python aercf_core.py nslkdd out # 5 seeds, ~1 min each on 2 vCPUs
python experiments.py out/ # E1-E7 -> out/results.json
python e4b_ensemble.py out/ # ensembles, adaptive attack -> out/e4b.json
python sim_audit.py out/sim_audit.json # Appendix B
python figures.py # Figures 1-8

```

Appendix B. What the companion visualisation shows

The browser visualisation in the AERCF repository illustrates the architecture interactively. To prevent its displays being read as results, we ported its flow generator, classifier and threshold controller to Python and characterised them (sim_audit.py). Each flow's detector scores, hybrid score and SHAP values are drawn from fixed uniform distributions conditioned on the generated label; no detector, EVT fit, SHAP computation, PGD attack, certified radius or reinforcement-learning update is computed. Its displayed metrics are therefore fixed by the generator (Table 9).

Table 9. Behaviour of the companion visualisation (Python port; 3,000 ticks per setting).

Setting	Displayed detection rate	Displayed FPR	Reason
Threshold controller off ($\theta = 0.52$)	87.4–88.9% at every threat level	0.00%	Benign scores never exceed 0.28; adversarial flows fall in $[0.33, 0.73]$, so $DR = 0.75 + 0.25 \times 0.525 = 88.1\%$ analytically
Threshold controller on	98.3–99.2%	0.00%	Because FPR is always 0, the controller lowers θ to its floor of 0.35 and stays there; $DR \rightarrow 0.75 + 0.25 \times 0.95 = 98.75\%$
“SHAP waterfall”	—	—	ϕ values are random draws; their sum differs from the displayed score by 0.197 on average, and correlates with it at $r = 0.004$ within the attack class

The visualisation remains a useful teaching aid for the architecture's data flow. It should be described as such in the repository, and figures from it should not be cited as performance evidence.