Journal of Information Security and Applications 100 (2026) 104496

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Adaptive active-defense hardening of ML-based NIDS against RL-driven adversaries: A comparative analysis with static defenses

Iacovos Ioannou [a,*], Christophoros Christophorou [b], Andreas Andreou [c], Marios Raspopoulos [b], Constandinos Mavromoustakis [c], Vasos Vassiliou [d,e], Fabrizio Granelli [f]

[a] Department of Computer Science, Philips University, Nicosia, Cyprus
[b] School of Sciences, UCLan Cyprus, Pyla, Larnaca, Cyprus
[c] Department of Computer Science, University of Nicosia, Nicosia, Cyprus
[d] CYENS Centre of Excellence, Nicosia, Cyprus
[e] Department of Computer Science, University of Cyprus, Nicosia, Cyprus
[f] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy

ARTICLE INFO

Keywords:
Network intrusion detection
Adversarial machine learning
Reinforcement learning
Active defense
Deep learning

ABSTRACT

Machine-learning-based Network Intrusion Detection Systems (NIDS) are vulnerable to adversarial evasion attacks that exploit fixed decision boundaries. This paper presents a modular Active Defense System (ADS) that orchestrates a co-evolutionary loop between reinforcement-learning (RL) attackers and an adaptive NIDS hardening pipeline with explicit promotion criteria, augmented by an optional anomaly-detection Threat Analysis Module (TAM). Unlike existing defenses that address attacker modeling and detector hardening in isolation, our framework simultaneously integrates adaptive RL-driven attackers, a dedicated anomaly-detection pre-screening layer and a closed-loop, guardrailed retraining pipeline with comprehensive cost reporting. Extensive experiments on the DReLAB benchmark show that every static NIDS suffers complete malicious-class F1 collapse under a mature RL attack, while headline accuracy remains deceptively high, an *accuracy illusion* rooted in class imbalance. The proposed active defense loop restores detection to near-perfect levels across diverse classifier families, with the top configuration achieving a post-hardening F1 of 1.0000. Four RL attackers are compared: Advantage Actor-Critic (A2C), Proximal Policy Optimization (PPO), Double Deep Q-Network (DDQN) and Soft Actor-Critic (SAC). A2C achieves identical damage at approximately 15× lower energy cost and wall-clock time than SAC, making it the most operationally relevant threat. Four TAM detectors are evaluated: Autoencoder (AE), Isolation Forest (IF), a vanilla Generative Adversarial Network (GAN) and Wasserstein GAN with Gradient Penalty (WGAN-GP). WGAN-GP emerges as the most reliable detector overall, sustaining a macro-averaged Adversarial Detection Rate of 96.9% across all attacker families, including ≈99% against the strongest adversary, while maintaining the lowest benign false-positive rate and highest precision-recall area. Supplementary validation on CICIDS2017, constrained perturbations, multi-seed analysis and benign-side detector evaluation confirm the consistency of these findings. The results should be interpreted as evidence from a controlled adversarial-learning framework rather than as proof of deployment-ready effectiveness on arbitrary enterprise traffic.

1. Introduction

In an era of ubiquitous connectivity, the security of digital networks is paramount. Network Intrusion Detection Systems (NIDS) serve as the frontline defense, monitoring network traffic for malicious activities and policy violations [1]. Traditionally, NIDS operated using signature-based detection, which is effective against known threats but fails to identify novel or zero-day attacks. To address this, the field has widely adopted anomaly-based detection, leveraging machine learning (ML) to learn the patterns of normal behavior and flag deviations [2]. Models

* Corresponding author.
E-mail addresses: ioannou.iakovos@philipsuniv.ac.cy (I. Ioannou), CChristoforou2@uclan.ac.uk (C. Christophorou), andreou.an@unic.ac.cy (A. Andreou), MRaspopoulos@uclan.ac.uk (M. Raspopoulos), mavromoustakis.c@unic.ac.cy (C. Mavromoustakis), vasiliou.vasos@ucy.ac.cy (V. Vassiliou), fabrizio.granelli@unitn.it (F. Granelli).

https://doi.org/10.1016/j.jisa.2026.104496

Available online 9 May 2026

ranging from classic ensembles such as Random Forests to complex Deep Neural Networks (DNNs) have achieved high accuracy in classifying network traffic.

However, the very ML models that empower modern NIDS have themselves become targets. The field of adversarial machine learning has shown that ML models are vulnerable to adversarial examples, which are inputs intentionally crafted with small perturbations that induce misclassification [3]. In the context of cybersecurity, an attacker can subtly modify malicious payloads to make them appear benign to an ML-based NIDS, effectively creating a security blind spot. Initial adversarial attacks were often white-box, requiring full knowledge of the target model. Today, attackers are far more sophisticated, employing black-box strategies and reinforcement learning (RL) to probe a target NIDS and iteratively learn a policy for generating successful adversarial samples with minimal effort [4]. This new class of intelligent, adaptive adversaries renders static, passively-trained NIDS architectures brittle and unreliable. A defense system that is trained once and then deployed is destined to fail against an attacker that can learn and adapt to its decision boundaries. This necessitates a paradigm shift from passive defense to *active defense*.

At the same time, a realistic assessment must distinguish between controlled adversarial-ML stress testing and operational validation. The goal of this paper is the former: to show, under a transparent and reproducible protocol, how static NIDS can fail against adaptive RL attackers and how a structured hardening loop can restore robustness. Accordingly, the present study emphasizes mechanism, measurement and controlled comparison rather than claiming that the resulting system is production-ready for arbitrary networks, traffic mixes, or deployment environments. To further strengthen internal and external validity, we conducted four supplementary validation experiments whose outcomes are integrated throughout the paper: (i) replication of the full pipeline on the CICIDS2017 dataset, (ii) a constrained-perturbation ablation that restricts feature changes to semantically mutable traffic attributes, (iii) five-seed variance analysis of the main configurations and (iv) benign-side evaluation of all TAM detectors. In each case, the qualitative conclusions remain consistent with the primary DReLAB results.

An Active Defense System (ADS) does not merely classify threats; it anticipates, interacts with and learns from them. It is a dynamic, closed-loop system that hardens its defenses in response to observed attacks [5–7]. Our core innovation is the design, implementation and rigorous empirical evaluation of a comprehensive, modular ADS framework tailored for NIDS security. Unlike most prior works, which examine these subsystems in isolation, our framework treats them as a unified whole. It orchestrates a co-evolutionary arms race between sophisticated RL attackers and a layered, adaptive defense. The novelty lies in its automated, full-lifecycle approach: detecting evasions, analyzing them with optional anomaly detectors and using them as a curriculum for a systematic hardening process with auditable promotion criteria.

Building on the active defense paradigm outlined above, we introduce a unified intrusion-detection testbed that *jointly* (i) models the adversary as a *continuous-control* reinforcement learning agent, instantiating **Soft Actor-Critic (SAC)**, **Proximal Policy Optimization (PPO)** and **A2C**[1] and (ii) interposes a front-loaded **Threat Analysis Module** (TAM) whose density estimators, specifically a vanilla **GAN** and a **Wasserstein GAN with Gradient Penalty (WGAN-GP)** critic, screen *every* probe *before* it reaches the production NIDS. The TAM verdict is injected into the attacker reward, while flagged samples drive a hardening loop with explicit promotion criteria. To the best of our knowledge, earlier active-defense studies have not jointly integrated this pre-screening and reward-shaping layer within a guardrailed retraining loop. Together, these two components form the backbone of our hierarchical framework, enabling cost-aware, co-evolutionary evaluation of adaptive attacks and adaptive defenses. In summary, this paper presents a comprehensive active defense framework designed, implemented and evaluated empirically for NIDS security. Our system is built on a hierarchical and modular architecture, enabling systematic evaluation of different components. The primary **contributions** of this work are:

1. **A Modular Active Defense Framework:** We present a complete, end-to-end system that allows for plug-and-play comparison of various NIDS models, anomaly detectors and adversarial attackers, facilitating rigorous and reproducible investigation of the dynamics of cyber arms races.
2. **Sophisticated Adversarial Simulation:** We model intelligent attackers using a diverse suite of advanced reinforcement learning agents, including continuous-action (Soft Actor-Critic, Proximal Policy Optimization, Advantage Actor-Critic) and discrete-action (Double Deep Q-Network) algorithms, providing a realistic and challenging testbed for NIDS robustness.
3. **Online Adaptive Retraining:** The core of our active defense is an online retraining loop. Successful adversarial samples are collected and used to create a new training set, which then retrains the NIDS. This allows the system to adapt its decision boundary and harden itself against the attacker's evolving strategy.
4. **Comprehensive Empirical Analysis:** We conduct an extensive set of simulations and present the results, quantifying not only the performance degradation of static systems and the recovery achieved through active defense, but also why headline accuracy can remain misleadingly high under severe class imbalance while malicious-class F1 collapses. We identify the most effective attacker and the most resilient defense configurations from dozens of combinations, providing actionable insights and cost-benefit analysis (in time and energy) for practitioners.
5. **Controlled Stress-Test Perspective and Supplementary Validation:** We position the framework explicitly as a controlled, leakage-safe adversarial-learning testbed for studying collapse, detector-assisted hardening and recovery dynamics. To test the robustness of the findings beyond the primary corpus, we validate the collapse-and-recovery pattern on CICIDS2017, assess attacker power under semantically constrained perturbations, report five-seed variance for the main configurations and measure benign-side detector precision and false-positive rates. The consistency of results across all four axes strengthens confidence in the framework while clearly delineating the boundary between controlled evaluation and production deployment.

The remainder of this paper is structured as follows. Section 2 reviews the literature on ML-based NIDS, adversarial attacks and active defense strategies. Section 3 presents the formal problem statement and the high-level architecture of our proposed system. Section 4 provides a detailed technical description of each component within the framework. Section 5 presents and analyzes the empirical results from our simulations. Section 6 synthesizes the findings into broader observations on arms-race dynamics, novelty, deployment implications and limitations. Finally, Section 7 concludes the paper and outlines promising directions for future research.

2. Related work

This section reviews the foundational literature that underpins our research, situating our contributions within three key domains. First, we survey the evolution of machine learning for intrusion detection, from classical algorithms to modern deep learning architectures. Then, we examine the parallel development of adversarial attacks, highlighting the critical shift from white-box methods to sophisticated, black-box strategies that leverage reinforcement learning. Finally, we analyze existing defensive strategies, distinguishing between passive, static methods and the dynamic, adaptive paradigm of active defense, which forms the basis

[1] To the best of our knowledge, prior IDS studies have primarily focused on discrete DDQN-like settings or gradient heuristics rather than this combination of continuous-control RL attackers for adversarial network-traffic generation.

Table 1
Feature-level comparison of the proposed framework against representative recent adversarial-NIDS works. ✓ = fully supported, ✗ = not supported, P = partial. Abbreviations: Acc = Accuracy, P = Precision, R = Recall, F1 = F1-Score, FPR = False Positive Rate, FNR = False Negative Rate, MCC = Matthews Correlation Coefficient, ASR = Attack Success Rate, ADR = Adversarial Detection Rate, BIM = Basic Iterative Method, AT = Adversarial Training, BB = Black-Box, WB = White-Box, DB = Decision-Based, SB = Score-Based.

Work	Attacker Type	Multi-Atk Comp.	Anomaly TAM	Adaptive Retrain	Cost Metrics	Eval. Metrics	Dataset	Threat Model
Qiu et al. [24]	Gradient (FGSM, C&W)	✗	✗	✗	✗	Acc, ASR	NSL-KDD, UNSW	WB
Tan et al. [25]	Query-based (mutation)	P	✗	✗	✗	Acc, ASR, F1	CIC-IDS2018	DB BB
Debicha et al. [26]	DRL (DQN)	✗	✗	✗	✗	Acc, F1, ASR	CTU-13	SB BB
Debicha et al. [28]	Multiple (5 types)	✓	✗	✗	✗	Acc, P, R, F1, ASR	CIC-IDS2018	DB BB
Roshan & Zafar [38]	Gradient (C&W)	✗	✗	✗	✗	Acc, P, R, F1	NSL-KDD, UNSW	WB
Paya et al. [6]	Gradient (FGSM, PGD)	P	P	P	✗	Acc, P, R, F1	CICIDS2017	WB
Tafreshian & Raisi [36]	Gradient (FGSM, PGD)	P	✗	✗	✗	Acc, FPR	NSL-KDD, UNSW	WB
Awad et al. [37]	Gradient (FGSM, PGD, BIM)	✓	✗	✗	✗	Acc, P, R, F1	CICIDS2017, NSL-KDD	WB
Rivas et al. [29]	Multi-agent RL	P	✗	✓	✗	Acc, F1	CICIDS2017	BB
Naeem et al. [39]	Gradient (FGSM, PGD, C&W)	P	✓	✗	✗	Acc, P, R, F1, AUC	NSL-KDD, UNSW	WB
Uddin et al. [15]	DRL (DQN)	✗	✗	P	✗	Acc, P, R, F1	NF-BoT-IoT	BB
Heydari et al. [41]	Gradient (FGSM, PGD)	P	✗	✗	✗	Acc, P, R, F1	CICIDS2017	WB
Hitaj et al. [27]	Survey (multiple)	P	✗	✗	✗	Survey (various)	Multiple	Multiple
Zhang et al. [42]	Gradient + query	P	✗	✗	✗	Acc, F1	NSL-KDD	WB + BB
This work	**RL (SAC, PPO, A2C, DDQN)**	✓	✓	✓	✓	**Acc, P, R, F1, FPR, FNR, MCC, ASR, ADR, ℓ_2, ℓ_∞, Time, Energy**	**DReLAB**	**SB BB**

of our work. To provide readers with a structured basis for interpreting the design choices of our framework, we close with a compact feature-level comparison against representative recent studies (Table 1). Our research builds upon several interconnected domains: ML-based intrusion detection, adversarial attacks against ML models and active defense systems.

2.1. Machine learning for intrusion detection

The application of machine learning to network intrusion detection has a substantial history. Early works employed conventional algorithms such as Support Vector Machines (SVMs), Naive Bayes and Decision Trees. Ensemble methods subsequently gained prominence owing to their robustness and strong generalization performance. Breiman [8] introduced the Random Forest (RF) algorithm, which has been widely applied to NIDS tasks. Chen et al. [9] developed XGBoost, a gradient boosting framework that is often a top performer in classification competitions and has been successfully used for intrusion detection.

With the rise of deep learning, researchers began applying various neural network architectures to NIDS. Artificial Neural Networks (ANNs) and deeper Deep Neural Networks (DNNs) have shown the ability to learn complex, non-linear relationships in network traffic data, often outperforming classical models [10]. Furthermore, Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [11], have been used not only for data augmentation but also as classifiers or anomaly detectors, where the discriminator's ability to distinguish real from fake data is repurposed to identify anomalous network flows [12]. Yan et al. proposed PerFLID, a personalized federated meta-learning intrusion detection approach for IoT; its key strength is improved detection under non-IID traffic distributions through personalized federated learning. However, PerFLID primarily targets distribution heterogeneity across clients and does not explicitly evaluate robustness under adversarially crafted evasion traffic, leaving the active hardening dimension unaddressed [13].

Hore et al. [14] proposed a sequential deep learning framework combining Long Short-Term Memory (LSTM) with deep reinforcement learning for robust NIDS, reporting strong detection performance on CICIDS2017; a notable strength is the integration of sequential modeling with RL-based decision-making. However, their approach targets single-model robustness without an adaptive retraining loop and does not evaluate resilience against RL-driven evasion agents that actively learn to bypass the detector. Uddin et al. [15] introduced a DRL-based NIDS with stacked LSTMs for zero-day attack detection, achieving high accuracy on unseen attack types through learned feature representations; yet their framework lacks a secondary anomaly-detection front-end and does not quantify the computational cost (time, energy) of the adaptation process. Our work complements these efforts by embedding diverse NIDS architectures within a closed-loop hardening framework that explicitly measures both security recovery and operational cost under controlled adversarial conditions.

2.2. Adversarial attacks on machine learning models

The vulnerability of ML models to adversarial examples was famously highlighted by Szegedy et al. [16]. Goodfellow et al. [3] proposed the Fast Gradient Sign Method (FGSM), a simple yet effective white-box attack that linearizes the loss function to compute a perturbation. Many more powerful attacks have since been developed, such as the Projected Gradient Descent (PGD) attack [17].

While these attacks are potent, they often assume a white-box setting. In cybersecurity, attackers rarely have full access to the target model. This has led to the development of black-box attacks, which rely

on querying the model and observing its outputs. A powerful class of black-box attacks uses reinforcement learning, where an agent is trained to find an optimal policy for perturbing inputs to cause misclassification. The NIDS is treated as an environment and the agent receives rewards for successful evasions. This approach has been shown to be highly effective for generating robust adversarial examples against malware classifiers and NIDS [18]. Our work simulates this advanced threat by using DDQN [19], SAC [20], PPO [21] and A2C [22] as our adversarial agents. Complementary to these RL-based agents, Wang et al. introduced ProGen, a projection-based traffic-space attack that bypasses multi-class NIDS while preserving protocol semantics [23].

Recent NIDS-focused studies increasingly emphasize feasibility constraints beyond abstract feature perturbations. Qiu et al. [24] investigated adversarial attacks against intrusion detection in IoT settings, demonstrating that carefully crafted perturbations can significantly degrade detection accuracy; a strength is the focus on IoT-specific constraints, but the attack formulation operates primarily at the feature level and may not always map to protocol-valid packet or flow transformations. Tan et al. [25] advanced realism by mutating live network traffic to evade learning-based NIDS, achieving evasion success rates of up to 35.7% by altering only time-based features; nevertheless, the contribution is centered on evasion generation and does not co-design an adaptive hardening mechanism that learns online from observed attacker behavior. Debicha et al. [26] introduced Adv-Bot to create realistic adversarial botnet traffic against NIDS, with strong realism for botnet-like behaviors and validation on contemporary datasets; yet the approach remains scenario- and behavior-dependent and does not directly address general-purpose, adaptive (policy-driven) evasion across heterogeneous attack classes.

Ennaji et al. [27] provided a comprehensive survey of adversarial challenges specific to NIDS, introducing a taxonomy that accounts for feature interdependency and traffic-space realizability, a valuable analytical contribution; however, the work does not implement or evaluate active defense mechanisms. Debicha et al. [28] presented TIKI-TAKA, a framework for assessing the robustness of deep-learning NIDS against five attack types including decision-based black-box attacks and proposed model-voting ensembling and query detection as defenses that restore detection rates to near 100%; a limitation is that their defense strategies are static and do not include online retraining against adaptive, learning-based attackers. Rivas et al. [29] recently proposed a multi-agent RL framework that models adversarial concept drift with active and continual learning; while their agent-based drift adaptation is novel and captures strategic evolution between attacker and defender, they do not incorporate a pre-screening anomaly-detection module (analogous to our TAM) or report energy costs.

While the adversarial ML works cited above have focused primarily on computer vision and general machine learning, a substantial body of research has emerged that examines adversarial attacks specifically against network intrusion detection systems. He et al. [30] provide a comprehensive survey examining the fundamental differences between adversarial learning in CV and NIDS domains, highlighting that directly applying CV-based attacks to network traffic ignores critical domain-specific constraints such as feature interdependency and traffic-space realizability. Lin et al. [31] proposed IDSGAN, leveraging generative adversarial networks to craft adversarial network traffic that evades detection. Alhajjar et al. [32] evaluated evolutionary computation methods, including particle swarm optimization and genetic algorithms, alongside GANs for adversarial example generation against NIDS.

Apruzzese et al. [33] critically analyzed threat models in existing NIDS adversarial research, identifying unrealistic assumptions about attacker capabilities and proposing a framework for modeling feasible attacks. More recently, Jmila and Ibn Khedher [34] provided a comparative study of adversarial attacks on both shallow and deep learning-based NIDS. These NIDS-specific works inform our threat model and highlight the need for domain-aware defensive mechanisms that our framework provides.

2.3. Defenses against adversarial attacks

Research into defending against adversarial attacks has primarily followed two paths: passive and active defense. Passive defenses aim to build models that are inherently robust. Adversarial training, where adversarial examples are generated and included in the training data, is one of the most effective known defenses [17]. Other techniques include defensive distillation [35] and input transformations. While useful, these methods require anticipating the types of attacks that will be encountered and are less effective against adaptive attackers who can find new vulnerabilities.

Active defenses, also known as adaptive defenses, are dynamic systems that respond to ongoing attacks. The concept is closely related to Moving Target Defense (MTD), which seeks to make the attack surface less static and predictable [5]. In the context of ML, active defense often involves an online learning or retraining component. When an attack is detected (or succeeds), the system analyzes it and updates the model to be robust against that specific attack vector in the future. **Our framework exemplifies this philosophy: it formalizes the process of capturing successful adversarial samples and incorporating them into a systematic, auditable retraining loop that continuously hardens the NIDS against evolving threats.**

We note that while standard adversarial training with gradient-based attacks (e.g., FGSM/PGD augmentation [17]) represents an important baseline, our framework specifically targets the more challenging scenario of adaptive, query-based RL attackers that do not require gradient access and can evolve their strategies over time.

Tafreshian and Raisi [36] proposed a defensive framework that simultaneously integrates adversarial training, dataset balancing, ensemble learning and feature engineering, reporting a 35% accuracy increase under adversarial conditions on NSL-KDD and UNSW-NB15; a strength is the multi-pronged combination of complementary techniques. However, the defense is applied as a one-time offline process and does not adapt to evolving RL-driven attackers in a closed-loop fashion. Awad et al. [37] presented a two-phase ensemble defense integrating adversarial training, label smoothing and Gaussian augmentation during training with input transformation and feature squeezing at inference time; while effective against gradient-based attacks such as FGSM and PGD, the framework does not consider query-based RL attackers and does not provide cost-benefit analysis in terms of time or energy. Roshan and Zafar [38] introduced a two-phase defense combining Gaussian data augmentation with feature squeezing against C&W white-box attacks, demonstrating improved robustness for DNN-based NIDS; a strength is the principled two-stage pipeline, but their limitation is the white-box assumption and the absence of an adaptive retraining mechanism for evolving threats.

Paya et al. [6] developed Apollon, a defense system that combines adversarial detection with model hardening against multiple attack types, achieving strong resilience on standard benchmarks; yet it relies on pre-generated adversarial samples rather than a co-evolutionary loop with learning attackers that discover novel evasion vectors over time. Naeem et al. [39] proposed NIDS-DA, using deep autoencoders to detect functionally preserved adversarial examples before they reach the classifier; while their detection-first approach is complementary to ours and addresses the critical constraint of functional preservation, they do not integrate it into an end-to-end active defense pipeline with adaptive retraining. Andreou et al. [40] proposed a DRL-driven Moving Target Defense for network slice security, dynamically adapting defenses via MDP modeling; their work validates the MTD paradigm we advocate and demonstrates the effectiveness of DRL-based policy adaptation, but operates at the network-slice level rather than at the NIDS classifier level.

Heydari et al. [41] proposed an adversarially trained neural-network NIDS (ADV_NN) and reported improved resilience against common

adversarial methods including FGSM and PGD; however, adversarial training alone adds nontrivial training cost and may still miss unseen, protocol-constrained mutations encountered in operation. Zhang et al. [42] discuss multiple mechanisms for intrusion-detection robustness (e.g., model voting/ensembling, adversarial training and query-aware defenses), providing a useful taxonomy of defensive strategies; however, these mechanisms are typically deployed as static protections and can degrade under adaptive attackers that continuously refine their policies. In contrast, our work treats robustness as an online decision-making objective and introduces a hierarchical active-defense process that adapts its detection and mitigation policies based on observed attacker behavior, closing the loop between attack observation and model hardening.

2.4. Comparative positioning

To provide readers with a structured basis for interpreting the contributions of this work, Table 1 compares our framework against representative recent studies across eight dimensions that are critical for practical, deployable NIDS hardening: (i) the type of attacker model employed (gradient-based, GAN-based, or RL-based); (ii) whether multiple attacker algorithms are compared; (iii) the presence of a secondary anomaly-detection screening module; (iv) whether the defense strategy is static (applied once) or incorporates adaptive/online retraining; (v) whether cost metrics such as execution time and energy consumption are reported; (vi) the primary evaluation metrics used; (vii) the dataset employed; and (viii) the threat-model assumption (white-box, score-based black-box, or decision-based black-box).

As Table 1 shows, existing works typically address a subset of these dimensions. Several proposals achieve strong detection under specific attack families, for instance, Awad et al. [37] and Roshan and Zafar [38] report high accuracy recovery against gradient-based white-box attacks and Debicha et al. [28] demonstrate near-perfect restoration via ensembling against decision-based attacks. However, none of these works simultaneously (a) models the attacker as an adaptive, policy-driven RL agent that evolves its strategy over time, (b) interposes a dedicated anomaly-detection Threat Analysis Module (TAM) before the retraining buffer to filter poisoned or low-quality samples, (c) implements a closed-loop, guardrailed adaptive retraining pipeline that hardens the NIDS using curated adversarial curricula and (d) reports comprehensive cost metrics (wall-clock time, energy in kWh) alongside standard detection metrics to support cost-aware deployment decisions. Our framework integrates these capabilities within a single, unified and reproducible evaluation, enabling a direct, controlled comparison between static and adaptive defenses under identical attacker conditions. This integration is what distinguishes our approach and makes it well suited for evaluating and hardening NIDS against the emerging class of intelligent, adaptive adversaries.

3. Problem formulation and system architecture

3.1. Problem formulation

This section provides a formal basis for our active defense framework. We begin by defining the core problem, mathematically formulating the competing objectives of the Defender and the Attacker in the context of network intrusion detection. We then specify the operational constraints of our scenario by outlining a realistic score-based black-box threat model. Finally, we present the high-level architecture of our proposed Active Defense System (ADS), detailing its modular components and the closed-loop workflow that enables the system to learn from and adapt to sophisticated adversarial threats. More specifically, we consider a scenario involving two primary actors: a **Defender** and an **Attacker**. Throughout this paper, f_θ denotes the current operational NIDS, $f_{\theta'}$ denotes a candidate model after fine-tuning and $f_{\theta^\star}$ denotes the promoted model that has passed all guardrail criteria. The adversarial repository is denoted $\mathcal{R}_{adv}$, from which a sampled curriculum $\mathcal{X}_{adv}^{curr}$ is drawn for retraining. $P_{\text{mal}} \in [0, 1]$ refers exclusively to the NIDS malicious-class probability output. These conventions apply uniformly to all tables, algorithms and figures throughout the paper.

The **Defender**'s objective is to maintain a secure network by deploying a NIDS, denoted by a classification function $f_\theta : \mathcal{X} \to \{0, 1\}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ is the d-dimensional feature space of network traffic and the labels represent benign (0) and adversarially perturbed botnet traffic (1) classes in the DReLAB-derived corpus used throughout this study. For readability, later references to the "malicious" or "positive" class correspond to this adversarial-botnet class unless stated otherwise. The Defender aims to maximize the detection rate of the positive class while minimizing false alarms. Furthermore, in our active defense setting, the Defender must also adapt its model parameters θ over time to counter the Attacker.

The **Attacker**'s objective is for a positive-class sample $x_+ \in \mathcal{X}$ (an adversarial botnet flow in the present DReLAB-derived setting, where the true label is 1) to be misclassified by the NIDS as benign. The Attacker achieves this by crafting a perturbed sample $x'_+ = x_+ + \delta$, where δ is a small perturbation vector. The Attacker's goal is to find a perturbation δ that solves the following optimization problem:

$$\min_{\delta} ||\delta||_p \quad \text{subject to} \quad f_\theta(x_+ + \delta) = 0 \tag{1}$$

where $||\cdot||_p$ is an L_p-norm, typically L_2. This formulation captures the dual goals of the attacker: successfully evading detection while keeping the perturbation minimal to avoid corrupting the payload's functionality and to remain stealthy. In our work, we model the attacker as an RL agent that learns a policy $\pi_{attack}(s) \to a$ to generate δ for a given state $s = x_+$.

3.2. Threat model

The adversarial threat model assumed in this work is a **score-based black-box** setting. The key characteristics are:

- **Attacker's Knowledge:** The attacker does not have access to the NIDS model's architecture, parameters, or gradients. The only information available is the final output probability score (P_{mal}) returned by the NIDS for a given input sample.
- **Query Access:** The attacker can repeatedly query the NIDS with modified samples to observe changes in the output score. In our experiments, we set a generous per-episode step limit (e.g., 10 queries per evasion attempt) to model a persistent adversary, but we track the total query count as a cost metric. This choice intentionally stress-tests the defender under a strong black-box adversary and may overstate the practical freedom available in rate-limited production environments. A query-budget sensitivity study (3, 5 and 10 steps) confirmed that collapse dynamics persist even under tighter budgets, though at reduced attacker efficiency. Enforcing stricter query budgets and delayed/noisy feedback remains an important direction for future work.
- **Attacker's Goal:** The RL agent's goal is to learn an efficient policy to drive the NIDS's malicious probability score below the decision threshold (τ_{evade}) while minimizing the magnitude of the applied perturbations.
- **Feature-Space Scope:** In the present study, perturbations are applied in normalized feature space and are not constrained by full packet-, flow-, or protocol-level realizability rules. This design is appropriate for controlled adversarial-ML evaluation, but it can overstate attacker freedom relative to real networks where many fields are semantically coupled or physically constrained. A constrained-perturbation ablation restricted to the subset of semantically mutable features confirmed that the collapse-and-recovery pattern persists under tighter perturbation bounds, though with modestly reduced attacker efficiency (see Section 5.6 for details).

The present threat model rests on four assumptions that are likely to overstate attacker freedom relative to production deployments and should therefore be interpreted accordingly:

(i) *Score-based feedback:* the attacker receives the exact malicious-class probability P_{mal} per query, rather than a hard label or a noisy or delayed signal;
(ii) *Generous query budget:* each evasion attempt allows up to 10 sequential queries without rate-limiting or cost penalties;
(iii) *Unconstrained feature perturbation:* all 39 raw features may be independently perturbed without enforcing packet-, flow-, or protocol-level semantic coupling; and
(iv) *Immediate, noiseless feedback:* the NIDS response is returned instantaneously with no stochastic delay or output noise.

Relaxing any of these assumptions would slow attacker convergence and reduce evasion effectiveness. The constrained-perturbation ablation (Section 5.6) and query-budget sensitivity study (3, 5 and 10 steps) already provide partial evidence: collapse dynamics persist under tighter bounds, but attacker efficiency is reduced. In production, rate-limiting, delayed or noisy feedback and stricter semantic feature coupling would further limit even efficient agents such as A2C. This model represents a realistic scenario in which an external adversary probes a deployed security system to find its weaknesses without any internal knowledge.

3.3. System architecture

We design an Active Defense System (ADS) that places a lightweight *Threat Analysis Module* (TAM) **before** the operational decision point of the Network Intrusion Detection System (NIDS) and closes the loop with auditable, guardrailed hardening. This ordering emphasizes that TAM primarily *drives learning*, whereas the NIDS *drives action*. As depicted in Fig. 1, the system involves four actors and distinct feedback pathways. Solid lines denote primary decision flow, while dashed lines represent feedback, training and telemetry.

The ADS operates as a modular loop. An attacker, instantiated as a reinforcement-learning (RL) agent (A2C, PPO, SAC, or DDQN), probes the system by proposing bounded perturbations δ to malicious seeds. The goal is to reduce the NIDS' malicious probability while keeping the perturbation cost small and the query counts low. Each perturbed candidate x'_{mal} is first inspected by the TAM, which is implemented with complementary detectors (AE, IF, GAN, or WGAN-GP). The TAM assigns anomaly scores and either *flags* the sample as adversarial and stores full lineage information for forensics and reproducibility, including the feature vector, cryptographic hashes, attacking agent and step, perturbation norms $\|\delta\|_{2,\infty}$, query count and timestamps, or *passes* it onward when all detector scores remain within benign bounds. The operational NIDS f_θ then outputs a calibrated probability $P_{\text{mal}} = f_\theta(x)$ and a decision at threshold τ, accompanied by an auditable rationale (e.g., SHAP or permutation-importance summaries) that is exported to the model card. When policy triggers fire, either because the accumulated evasion count reaches N_{fail} or because a scheduled cadence is reached, the adaptive retraining job assembles a stratified curriculum of flagged evasions, balancing subtle and high-norm cases, mixes them with a curated benign replay buffer and the historical corpus and fine-tunes the model at a low learning rate to avoid catastrophic forgetting. The candidate model is validated on untouched benign, malicious and adversarial hold-outs; promotion to production occurs only if guardrail metrics improve simultaneously (F1 score increases while FPR and calibration error decrease, with MCC monitored for imbalance robustness), at which point $f_{\theta'}$ replaces f_θ. Throughout, rich telemetry is exported, including model cards, drift indicators, query statistics and measured energy consumption in kWh so analysts can audit what changed, why it changed and at what computational cost.

The workflow proceeds as follows:

1. **Attacker (①):** An RL agent π_{attack} interacts with the system via an adversarial environment. At step t, it proposes a bounded perturbation $a_t = \delta_t$ and receives a shaped reward that encourages low-norm, query-efficient evasions.
2. **Threat Analysis Module** (②): The TAM assigns anomaly scores; samples exceeding detector thresholds (optionally refined with EVT tails) are *flagged* and written with full metadata to the evasion repository. Non-flagged samples proceed to the NIDS.
3. **NIDS Decision (③):** The classifier f_θ outputs P_{mal} and a decision at threshold τ. Rationale (e.g., SHAP, permutation importance) and counters are logged for auditing and model cards.
4. **Adaptive Hardening (④):** When triggers fire (e.g., N_{fail}=20,000 or a cron window), the system builds a stratified curriculum from flagged evasions, mixes a curated benign replay buffer, fine-tunes $f_\theta \to f_{\theta'}$ with a small learning rate and validates on held-out benign, malicious and adversarial sets. Only if F1 score improves and FPR stays within bounds is $f_{\theta'}$ promoted. Telemetry, including energy, is exported.

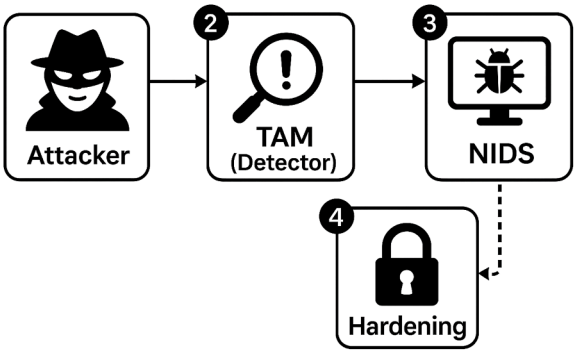

Fig. 1. Active defense system (ADS) framework architecture.

The instantaneous reward used in the adversarial environment is

$$R = (1 - P_{\text{mal}}) - w_p \|\delta_t\|_2 + R_{\text{det}} + \mathbb{1}\left[P_{\text{mal}} < \tau_{\text{evade}}\right] \cdot \beta,$$

which prioritizes reductions in the operational malicious score while penalizing large perturbations, discouraging detector-obvious manipulations and strongly rewarding fast, successful evasions. Here $P_{\text{mal}} \in [0, 1]$ is the NIDS probability for the malicious class; δ_t is the step-t perturbation in the normalized feature space and is clipped to respect a hard ℓ_∞ budget $\|\delta_t\|_\infty \le \varepsilon$; $w_p > 0$ weights the perturbation penalty (we use $w_p = 0.1$); R_{det} is a TAM-driven shaping term that penalizes samples flagged by any detector (implemented as -1.0 if flagged and $+0.1$ otherwise in the main experiments); $\mathbb{1}[P_{\text{mal}} < \tau_{\text{evade}}]$ is a terminal success indicator with the evasion threshold τ_{evade} (set to the operational NIDS decision threshold in the adversarial environment; in our runs $\tau_{\text{evade}} = 0.5$); and $\beta > 0$ is a success bonus (we use β=5) that dominates per-step terms to incentivize rapid, low-query evasions. With feature dimension d and budget ε, $\|\delta_t\|_2 \le \sqrt{d}\,\varepsilon$, which bounds the penalty term and stabilizes learning. In our experiments, we set $\varepsilon = 0.1$ in the normalized feature space, allowing perturbations up to 10% of each feature's range. Episodes are capped at T=10 steps and use discount factor γ=0.99 to prefer shorter evasions; success ends the episode immediately (`done` = True), preventing reward hacking via superfluous steps. This closed-loop architecture keeps the defender *continually learning* rather than frozen at a single decision boundary. Verified evasions identified by the TAM are converted into auditable training signal, curated into a stratified curriculum and used to fine-tune the NIDS ($f_\theta \to f_{\theta'}$) only when guardrails confirm a true improvement (higher F1 score with controlled FPR, stable calibration and strong MCC). In effect, the system is not a monolithic classifier but a learning-driven *process*: attack exploration feeds evidence to the retraining pipeline; validation enforces safety; promotion updates the operational model; and telemetry (including drift and energy) documents each change. By coupling exploration with disciplined, criteria-based hardening, the ADS avoids the frozen

Table 2
Roles of RL attackers and anomaly detectors in the ADS workflow (Section 3.3).

Component	Stage / Interface	Type	Primary role and notable traits
A2C	① Adversarial Env.	On-policy RL	Most *cost-efficient* attacker: learns evasions quickly with minimal time/energy and therefore represents a realistic baseline threat.
PPO	①	On-policy RL	Clipped-objective training gives stable learning; sits between A2C's speed and SAC's broader exploration.
DDQN	①	Off-policy RL	Discrete action space; double Q-learning avoids over-estimation when using a fixed perturbation code-book.
SAC	①	Off-policy RL	Entropy-regularised continuous control; discovers the strongest evasions but at the highest computational and power cost.
AE	② Threat Analysis Module	Reconstruction	Light-weight reconstruction error; near-perfect ADR on modest attacks, less effective against highly sophisticated SAC perturbations.
Isolation Forest	②	Tree ensemble	Negligible overhead; isolates obvious outliers but misses subtle high-dimensional perturbations.
GAN (disc.)	②	Density proxy	Discriminator learns benign-manifold boundary; good all-round ADR, especially for distribution-shift attacks.
WGAN-GP (critic)	②	Density proxy	Most robust overall detector; $\approx$ 99% ADR versus SAC and $\approx$ 97% versus A2C, at the price of slightly higher computational cost due to the gradient-penalty term.

Legend: ① Adversarial-environment block that generates perturbed samples; ② Threat Analysis Module (TAM) that filters or labels samples before they enter the retraining buffer.

decision boundaries that undermine static NIDS and sustains robustness against adaptive, cost-efficient RL attackers.

Fig. 1 depicts the data flow of the Active Defense System (ADS). Table 2 complements that diagram by summarizing *what* each reinforcement-learning (RL) attacker and anomaly detector does, *where* it plugs into the workflow and *why* it is useful.

4. Framework components and methodology

This section provides a structured overview of the framework's components and methodology. First, we define the data source and preparation steps: we construct an evaluation corpus from DReLAB with a binary `is_adv` label (benign vs. adversarially perturbed botnet flows) and 39 features, apply sanitization, standard scaling, PCA retaining 95% variance [43], winsorization and $[-1, 1]$ scaling for GAN components and fix leakage-safe 70/15/15 splits with saved hashes plus an adversarial holdout. Second, we introduce the attackers and the environment, formalizing an episodic MDP with bounded, norm-aware perturbations and reward shaping and deploying DDQN for discrete actions alongside SAC, A2C and PPO for continuous control with stability heuristics and entropy regularization (Algorithm 1). The Threat Analysis Module (TAM) then supplies unsupervised AE, Isolation Forest and GAN/WGAN-GP critics that provide secondary screening and inject penalties into the attacker's reward (Algorithm 3). The NIDS classifiers act as the primary defenders, including tuned ANN/DNN models trained with Adam [44] and He initialization [45], classical ensembles RF [8] and XGBoost [9] and a supervised **GAN-IDS**. An active defense mechanism then triggers batch hardening on accumulated evasions, curates a stratified adversarial curriculum and fine-tunes with conservative learning rates while mitigating catastrophic forgetting (Algorithm 2). Finally, a complete end-to-end workflow integrates these stages: baseline training, attacker maturation, degradation measurement, evasion collection, hardening and reassessment on untouched tests and fresh attacks (Algorithm 4), yielding a reproducible co-evolutionary evaluation of NIDS robustness and recovery.

4.1. Data source and preparation

The empirical foundation of our research is built upon the DReLAB (Deep Reinforcement Learning Adversarial Botnet) dataset [46], a pivotal resource in the field of adversarial cybersecurity. This publicly available dataset is meticulously engineered to serve as a benchmark for assessing the resilience of botnet Intrusion Detection Systems (IDS) when confronted with sophisticated adversarial attacks. The significance of DReLAB lies in its unique composition; unlike traditional datasets that only contain benign and malicious traffic, DReLAB includes a third, crucial category: adversarial samples. These are not merely theoretical constructs but are actual botnet traffic flows that have been strategically and minimally altered using advanced Deep Reinforcement Learning (DRL) agents, specifically Double Deep Q-Network and Deep Sarsa models. The primary objective of these alterations is to craft malicious samples that can successfully bypass detection by state-of-the-art machine learning classifiers, thereby mimicking the actions of a knowledgeable attacker.

This makes DReLAB particularly useful for controlled adversarial stress testing, but it also defines the scope of the claims that can be made. Because the corpus is feature-based, clean and purpose-built for adversarial evaluation, it does not fully reproduce the noise, temporal dependence, protocol semantics and operational diversity of enterprise traffic. As a result, the empirical results in this paper should be interpreted primarily as evidence of relative behavior under adversarial pressure rather than as a direct estimate of operational performance in production Security Operations Center (SOC) environments. To test for corpus-specificity, we replicated the full attack-collapse-hardening-recovery loop on CICIDS2017 using the same pipeline; the collapse-and-recovery dynamics remained qualitatively consistent, providing initial cross-dataset corroboration of the central findings.

Our methodology for constructing the final dataset for our experiments involved a careful process of consolidation and augmentation, guided by the logic encapsulated in the provided 'join.py' script. We began by isolating two primary subsets from the DReLAB collection: a set of purely benign network traffic flows and a distinct set containing the DRL-generated adversarial botnet samples. These two subsets were then concatenated to form a single, unified dataset. This step is critical for creating a challenging and realistic training and evaluation environment for our models.

A key contribution of our data preparation phase was the engineering of a new binary feature, which we designated as `is_adv`. In the experimental corpus used throughout this paper, `is_adv` is the *primary binary label*: we assign `is_adv` = 1 to adversarially-perturbed botnet flows and `is_adv` = 0 to benign flows. Accordingly, the "malicious" class (label 1) in our downstream NIDS training/evaluation corresponds to adversarial botnet samples in this constructed split, while label 0 corresponds to benign traffic. This aligns the data construction (benign + adversarial botnet) with the learning objective and avoids implying the presence of a separate "standard malicious" subset in the final concatenated dataset. The resulting feature space for our study is comprehensive, comprising 39 distinct raw features (reduced to 23 principal components after PCA, retaining 95.89% of variance). For attack generation, perturbations are applied in the standardized 39-dimensional raw feature space before PCA. When a downstream NIDS backbone uses PCA, the perturbed sample is subsequently passed through the fixed PCA transform fitted on the training split only. Thus, the RL environment operates on the pre-PCA feature representation, whereas the classifier receives the same post-PCA representation used during training.

4.1.1. Data preprocessing

A robust preprocessing pipeline is essential for the effective training of all downstream models. The process is executed once and the results are cached to ensure consistency and efficiency across simulation runs.

1. **Data Sanitization:** Raw data is first cleaned by ensuring all feature columns are numeric, coercing non-numeric entries to NaN and dropping any rows with missing values.
2. **Standard Scaling:** All features are scaled using Scikit-learn's `StandardScaler`. This transformation standardizes features by removing the mean and scaling to unit variance. This is critical for the stable training of neural networks and for distance-based algorithms.
3. **Dimensionality Reduction:** We use Principal Component Analysis (PCA), a foundational technique for feature extraction [43]. PCA projects the high-dimensional scaled data onto a lower-dimensional subspace while preserving a specified amount of the original variance. In our framework, we configure PCA to retain 95% of the variance, reducing the 39-dimensional feature space to 23 principal components (capturing 95.89% of variance). This transformation mitigates noise and improves numerical stability for the downstream neural models. We note that total wall-clock time in our co-evolutionary loop is dominated by RL attacker exploration, not by the PCA step itself; consequently, PCA does not reduce end-to-end runtime (see Table 4) but does lower overfitting risk for the classifier family under test.
4. **Specialized Scaling for GANs:** For the GAN and WGAN-GP anomaly detectors, which use a `tanh` activation function in the generator's output layer, the benign training data is separately scaled to the range $[-1, 1]$ using a `MinMaxScaler`. This ensures that the generated data distribution aligns with the range of the real data distribution, which is crucial for stable GAN training.

More specifically, to ensure the data is pristine and optimized for our models, we employ a multi-stage preprocessing pipeline. The flow of this pipeline involves several critical steps designed to enhance data quality and ensure robust model training:

1. *Feature Engineering Considerations:* Beyond simple numeric coercion, a robust pipeline must anticipate varied data types. For categorical fields like protocol types (e.g., TCP, UDP, ICMP), which are common in network data, one-hot encoding is a standard approach to create a binary column for each category. However, for features with high cardinality, this can lead to a curse of dimensionality. In such cases, entity embeddings can be trained to represent categories in a dense, low-dimensional vector space, capturing semantic relationships between them. Although the DReLAB features are predominantly numeric, our pipeline architecture is designed with this generality in mind, ensuring its applicability to a wider range of network security datasets without modification.
2. *Outlier Handling:* Network traffic data is notoriously prone to extreme values, which can arise from measurement artifacts, long-running flows, or denial-of-service attacks. These outliers can disproportionately influence the model's training process. To mitigate this, we employ winsorization, which caps extreme values at a specified percentile (e.g., the 99th percentile). This is preferable to outright removal, as these outliers may contain valuable information about anomalous, potentially malicious, activity. This controlled clipping is particularly important in an adversarial context, as an attacker might intentionally send traffic with pathological feature values to destabilize the detector. Winsorization helps to stabilize model training while preserving the integrity of the underlying data distribution.
3. *Train/Test Leakage Mitigation:* A critical and often overlooked error in machine learning workflows is data leakage, where information from the test set inadvertently influences the training process. To prevent this, all data transformations, particularly stateful ones like scaling (e.g., `StandardScaler`) and dimensionality reduction (e.g., PCA), are fitted exclusively on the training data split. The parameters learned from the training data (e.g., mean and standard deviation) are then saved and applied to transform the validation and test sets. This practice ensures that our performance metrics are a true reflection of the model's ability to generalize to unseen data, providing a realistic assessment of its deployment-readiness.
4. *Persistence and Versioning:* To ensure full reproducibility and support robust MLOps practices, every fitted data transformer (e.g., the scaler and PCA objects) is serialized and saved to disk. Each saved artifact is versioned, with a unique identifier derived from a checksum (e.g., SHA-256 hash) of the raw dataset it was trained on. As an additional integrity safeguard, duplicate-sample overlap across train, validation, test, degradation and adversarial-holdout partitions was disallowed by construction through fixed split hashes and persisted partition metadata; no sample identity appears in more than one partition. This rigorous versioning creates a clear audit trail, allowing for exact replication of experiments and facilitating debugging. If a model's performance changes unexpectedly, we can precisely determine if a change in the preprocessing pipeline or the underlying data is the cause, which is essential for maintaining reliable models in production environments.
5. *Complexity:* The computational cost of our preprocessing pipeline is primarily driven by the Singular Value Decomposition (SVD) algorithm at the core of PCA. For a dataset with n samples and d features, the complexity of exact SVD is $\mathcal{O}(\min\{nd^2, dn^2\})$. For high-dimensional datasets where d is large, this can become a bottleneck. To address this, our pipeline includes an option to use randomized SVD, which provides a fast approximation of the principal components with a significantly lower computational cost. This makes our approach scalable to the very large datasets often encountered in network security.
6. *Impact on RL Attackers:* Feature scaling is not just a best practice for model training; it is a critical defensive measure in an adversarial setting. Reinforcement learning (RL) agents designed to attack a model often incorporate a penalty term in their reward function based on the magnitude of the perturbations, $||\delta||$. Without normalization, this penalty is meaningless. An attacker could make a large-magnitude change to a feature with a naturally wide range (like `TotBytes`) that goes relatively unnoticed. By scaling all features to have a similar range (e.g., zero mean, unit variance), we force the perturbation penalty to reflect the statistical significance of the changes, compelling the RL agent to find more subtle and meaningful attack vectors.
7. *Robust Scaling Variants:* The standard `StandardScaler` is effective but can be sensitive to outliers, as the mean and standard deviation are easily skewed by extreme values. For datasets with heavy-tailed distributions or known anomalies, alternative strategies may be more suitable. Our pipeline is designed to be modular, allowing for the use of `RobustScaler`, which uses the median and interquartile range, or quantile transforms. These methods are more resilient to outliers and can lead to more robust model performance. Future work will involve comparing these scaling strategies to evaluate the trade-off between model performance and resilience to adversarial manipulation.
8. *Data Quality Metrics:* Continuous monitoring of data quality is essential for maintaining a healthy machine learning system. Before and after each preprocessing step, we log a suite of data quality metrics, including the number of records, the ratio of missing values for each feature and the variance retained by PCA. These metrics provide a quantifiable snapshot of the data's integrity at each stage. As proposed by [47,48], these metrics can be integrated into explainability dashboards, providing security analysts with crucial context and helping to diagnose data drift or other anomalies that could impact model performance and reliability over time.

Table 3 reports the exact sample counts for each data partition used throughout this study and provides the justification for selecting

Table 3
Composition of the DReLAB-derived corpus used in this study (split hash: `41e1fdad13b1870a`). Benign-to-adversarial ratio ≈ 20.76:1. The canonical 15% test split (406,020 samples) is used exclusively for baseline and post-hardening evaluation; it was never seen during training, tuning, or retraining. The identical validation and test counts reflect a stratified split with equal 15% partition sizes, not duplicated data.

Split	Benign (`is_adv = 0`)	Adversarial (`is_adv = 1`)	Total
Train (70%)	1,807,704	87,056	1,894,760
Validation (15%)	387,365	18,655	406,020
Test (15%)	387,365	18,655	406,020
Adversarial holdout	0	17,411	17,411
Total (excl. holdout overlap)	2,582,434	124,366	2,706,800

DReLAB over alternative benchmarks such as CICIDS2017 or NSL-KDD: DReLAB is the only publicly available corpus that already contains DRL-generated adversarial botnet flows, making it a natural match for our RL-based threat model. The benign-to-adversarial class ratio is approximately **20.76:1** (2,582,434 benign versus 124,366 adversarial samples, excluding the separate holdout). This imbalance contributes to accuracy inflation whenever adversarial misses are aggregated with a much larger benign background, producing the accuracy illusion documented in Section 5.3.1. Importantly, the canonical test split is not small: it contains 406,020 unseen samples, including 18,655 adversarial positives and 387,365 benign samples (Table 3), which is larger than the test partitions used in several representative adversarial-NIDS studies [24,36,38]. With 18,655 adversarial positives, a one-percentage-point difference in recall corresponds to roughly 187 samples, making adversarial-class metrics statistically stable across configurations. Furthermore, the large false-negative counts reported later in Table 8 are measured on separate attacker-generated evaluation pools of unseen malicious/adversarial seeds, not on the canonical 15% test split. The near-perfect pre-attack detection rates are consistent with the baseline results reported by Venturi et al. [46] in the original DReLAB paper and subsequent independent studies, confirming that this is a reproducible property of the dataset rather than an artifact of our implementation. Dataset quality was evaluated following the methodology of [48]. All split hashes are fixed and persisted to prevent any cross-partition leakage.

4.1.2. Data splitting and validation

The DReLAB-derived corpus was partitioned once into training (70%), validation (15%) and testing (15%) sets and the resulting split hashes were saved and reused across all experiments to prevent any cross-partition contamination. The validation set was used exclusively for hyperparameter tuning and early stopping; it was never merged back into training. The canonical test split in Table 3 was kept strictly untouched for baseline and post-hardening final evaluation. We use four distinct evaluation pools throughout the paper: (i) the canonical validation split; (ii) the canonical untouched test split for baseline and post-hardening reporting; (iii) a separate attacker-generated degradation pool of unseen malicious/adversarial seeds to measure pre-hardening attack damage; and (iv) a separate adversarial holdout set, disjoint from both training and retraining curricula, as a promotion safeguard. All stateful preprocessing operations, including standardization, winsorization thresholds, PCA and min-max scaling for GAN components, were fitted on the training split only and applied unchanged to validation, test and holdout data. At the framework level, every transformer exposes separate `fit` and `transform` calls; only `fit` (or `fit_transform`) was invoked on the training partition, eliminating any pathway for test-set statistics to influence the learned transformations. No sample used for attacker maturation, detector-threshold calibration, retraining, or holdout validation was reused in the final canonical test evaluation. The split hash (`41e1fdad13b1870a`), serialized preprocessor objects and partition metadata are available upon request to support independent reproducibility verification. This evaluation design was adopted specifically to rule out train-test leakage and to ensure that baseline scores are measured on a large, fixed, unseen test partition rather than on training data or on a very small sample.

The canonical train/validation/test partitions are shared across all NIDS backbones; however, the attacker-generated degradation pools reported later are model-specific because each RL attacker is trained separately against each target NIDS and then evaluated on the successful evasions it generates for that specific attacker-target pairing. Consequently, the post-attack false-negative totals can differ across NIDS even when the attacker family is the same.

The leakage-safe protocol strengthens internal validity, but it does not by itself guarantee external validity. All primary conclusions in this paper, therefore, pertain to the DReLAB-derived setting and to the feature-space adversarial process studied here. To begin testing external validity, we applied the same preprocessing and evaluation pipeline, identical split ratios, hash-saved partitions and guardrail criteria to CICIDS2017, obtaining qualitatively consistent collapse-and-recovery patterns. Extending the loop to additional datasets with different traffic characteristics, including IoT and multi-domain corpora, is necessary before claiming broad generalization across enterprise, IoT, or multi-domain NIDS scenarios and is identified as a priority next step.

To address concerns about whether the 95% variance-retention threshold causes overfitting or underfitting, we conducted a sensitivity study across four PCA configurations (Table 4). All settings yield identical clean-data F1 (0.9999) and post-attack F1 (0.0000), confirming that the attacker's ability to collapse the static NIDS is independent of PCA depth. Post-hardening F1 recovers to 1.0000 under No-PCA, 90% and 98% thresholds and to 0.9999 under the main 95% setting; the marginal difference of 5×10^{-5} lies within run-to-run variance and does not indicate systematic overfitting or underfitting at any tested compression level. We retain the 95% threshold as the primary setting because it provides the best balance between dimensionality reduction (23 components) and information preservation (95.89% variance) for the neural classifier family.

4.2. Adversarial environment and attack agents

To simulate a realistic threat, we designed a sophisticated adversarial environment and a suite of RL-based attackers.

4.2.1. The adversarial environment

The `AdversarialEnv` acts as the interface between attacker and defender in an episodic, goal-oriented Markov decision process. This episodic design provides a controlled platform for adversarial-learning analysis, but it abstracts away the temporal continuity, delayed feedback and streaming effects of real deployments in which detectors operate continuously over time. The **state space** S comprises the standardized raw 39-dimensional feature vector s_t of the (potentially perturbed) positive-class sample at time t. For the RL environment only, this pre-PCA representation is linearly rescaled to $[0, 1]$ so that perturbation bounds remain comparable across heterogeneous features. The **action space** A consists of a perturbation vector $a_t \equiv \delta_t$ of the same dimension as the state. In the unconstrained primary setup, all 39 raw features are potentially perturbed; after perturbation, the sample is mapped back to the standardized pre-PCA space and then passed through the fixed PCA transform whenever a downstream classifier was trained with PCA. For DDQN, we use a discrete action set, while SAC, A2C and PPO operate in a continuous space where actions are sampled from $\mathcal{N}(\mu, \sigma)$, squashed

Table 4
Sensitivity of key results to PCA variance-retention threshold (A2C attacker, DNN classifier, no TAM detector). Original feature dimension: d=39. All settings yield identical clean and post-attack F1; post-hardening differences are within run-to-run variance.

Setting	#Comp.	Clean F1	Post-Atk F1	Post-Harden F1	Time (s)	Energy (kWh)
No PCA	39	0.9999	0.0000	1.0000	17,724.7	1.8306
PCA 90%	20	0.9999	0.0000	1.0000	81,953.0	8.5285
PCA 95% (main)	23	0.9999	0.0000	0.9999	81,447.0	8.5046
PCA 98%	26	0.9999	0.0000	1.0000	78,026.4	8.1923

through `tanh` and rescaled to satisfy stealth constraints. Episodes terminate either when the agent successfully evades the NIDS or when a maximum horizon (e.g., $T = 10$ steps) is reached, which both caps cumulative distortion and serves as a computational guardrail.

Formally, we model the environment as an MDP $\langle S, A, P, R, \gamma \rangle$ with deterministic transition $P(s_{t+1} \mid s_t, a_t) = \text{clip}(s_t + a_t)$ and discount factor $\gamma = 0.99$ to encourage fast evasions; actions are bounded by an ℓ_∞ constraint $\|\delta_t\|_\infty \le \epsilon$ to mimic operational stealth. The reward is computed at every step by Algorithm 3 and includes a shaping term R_{detector} that directly penalizes the agent when the Threat Analysis Module (TAM) fires, guiding learning toward perturbations that fool *both* the primary NIDS and the secondary anomaly detectors.

To accelerate training and diversify experience, we batch multiple environment instances in parallel (e.g., 32), each rolling out distinct malicious seeds. For off-policy agents such as SAC and DDQN we maintain a large replay buffer of size 10^6 storing $(s, a, r, s', \text{done})$ tuples to preserve a broad memory of past interactions. While our main results use a random ordering of seeds for fairness, the framework also supports curriculum attacks that sort seeds from "easy" to "hard" according to the NIDS's initial confidence. It can also inject Gaussian noise into the defender's output probabilities to emulate stochastic responses. This complicates convergence but improves policy generality. Finally, a substantial terminal bonus of +5.0 emphatically rewards successful evasion so that the primary objective dominates small per-step penalties, balancing exploration with decisive exploitation.

4.2.2. The attacker agents

The Double Deep Q-Network (DDQN) agent operates in a discrete action space defined by a codebook of perturbation templates, selecting from a predefined set of vectors δ that alter features in controlled ways. DDQN reduces Q-value overestimation by decoupling action selection and target evaluation: the online network $Q_{\theta'}$ chooses the greedy action while the target network Q_{θ^-} evaluates it, yielding the target

$$Y_t^{\text{DDQN}} = R_{t+1} + \gamma\, Q\Big(S_{t+1}, \arg\max_{a'} Q(S_{t+1}, a'; \theta'); \theta^-\Big),$$

and optimizing the squared Bellman error

$$\mathcal{L}(\theta') = \mathbb{E}_{(s,a,r,s')}\Big[\Big(Y_t^{\text{DDQN}} - Q_{\theta'}(s,a)\Big)^2\Big].$$

Two networks are maintained, online $Q_{\theta'}$ (online network) and target Q_{θ^-} (target network), with periodic synchronization every C steps via $\theta^- \leftarrow \theta'$. Exploration follows an ϵ-greedy policy with ϵ annealed from 1.0 to 0.05 over 100k steps, balancing broad search early with exploitation as training progresses.

The Soft Actor-Critic (SAC) agent addresses continuous perturbation control by maximizing expected return plus an entropy bonus that encourages exploration over the action space. The objective aggregates reward and policy entropy $\mathcal{H}$:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\big[r(s_t,a_t) + \alpha\, \mathcal{H}(\pi(\cdot \mid s_t))\big],$$

where the actor outputs distribution parameters $\mu_\phi(s), \sigma_\phi(s)$, samples actions via the reparameterization trick and two critics Q_{ψ_1}, Q_{ψ_2} mitigate positive bias. The temperature α is auto-tuned online to target a desired entropy level, trading off exploitation and exploration adaptively. Being off-policy, SAC reuses experience through a replay buffer, making it more sample-efficient than on-policy methods and partly explaining the runtime differences reported in Table 7. Stability is further improved with gradient clipping, target-network smoothing and Polyak averaging.

Algorithm 1 Soft actor-critic training (Attacker).

```
Initialize actor π_φ, critics Q_ψ1, Q_ψ2, target critics Q̄_ψ1, Q̄_ψ2, temperature α
Initialize replay buffer D
for episode = 1 to M do
    Observe initial state s_0
    for timestep = 0 to T − 1 do
        Sample a_t ~ π_φ(·|s_t)
        Execute a_t, observe r_t, s_{t+1}, done
        Store (s_t, a_t, r_t, s_{t+1}, done) ∈ D
        Sample minibatch from D
        Update critics by minimizing Bellman error with entropy term
        Update actor by gradient of expected Q plus entropy bonus
        Adjust α via gradient step on J(α)
        Polyak average target networks
        if done then break
```

We also implement two advanced on-policy actor-critic agents. Advantage Actor-Critic (A2C) uses parallel workers and the advantage estimate

$$A_t = R_t + \gamma V(s_{t+1}) - V(s_t)$$

to reduce variance and stabilize policy-gradient updates. Proximal Policy Optimization (PPO) refines this approach with a clipped surrogate objective that constrains the magnitude of policy updates; we set the clipping parameter to $\epsilon = 0.2$ to balance stability and improvement. Compared with the discrete DDQN codebook, the continuous-action agents (SAC, PPO, A2C) can fine-tune per-feature perturbations, uncovering subtle low-norm changes that are difficult to predefine discretely. However, unlike SAC, the A2C and PPO are on-policy and thus require fresh trajectories for each update, reducing sample reuse and increasing data collection cost. As with SAC and DDQN, we employ gradient clipping, target smoothing and Polyak averaging where applicable to promote stable convergence.

To provide a clearer view of adversarial learning, Algorithm 1 details the SAC loop, which we found to be the most formidable-though resource-intensive-attacker. The entropy term $\mathcal{H}$ is central to SAC's exploratory strength, driving discovery of a broader repertoire of effective attack vectors than purely reward-maximizing policies. Across all agents we track *Attack Success Rate* (ASR) versus training steps to monitor convergence. We also log attacker cost via the cumulative ℓ_2 perturbation norm and the per-episode query count. Stability tricks shared across implementations include gradient clipping, Polyak averaging of target networks and (where applicable) target-network smoothing, which collectively reduce variance, curb destructive updates and expedite reliable convergence.

Algorithm 2 Adaptive NIDS retraining (Batch Hardening).

Input: current NIDS f_θ, optional TAM $D_{anomaly}$, training split (X_{train}, Y_{train}), validation split (X_{val}, Y_{val}), adversarial holdout $X^{adv}_{holdout}$
Output: promoted model $f_{\theta\star}$ or retained model f_θ
Step 1 - Mature the Attacker: Train the selected RL attacker agent (π_{attack}) against the current NIDS model (f_θ) and optional TAM until its attack policy converges.
Step 2 - Generate Candidate Evasions: Use the mature attacker policy π_{attack} to generate candidate evasions from a disjoint attack-generation partition of unseen positive-class samples.
Step 3 - TAM Filtering and Repository Logging: Route each candidate through the TAM (if enabled); flagged or otherwise confirmed successful evasions are stored with hashes, perturbation norms and trajectory metadata in the adversarial repository $\mathcal{R}_{adv}$.
Step 4 - Build a Hardened Curriculum: Stratify $\mathcal{R}_{adv}$ by perturbation norm and query difficulty to obtain $\mathcal{X}^{curr}_{adv}$; sample a curated benign replay buffer $\mathcal{B}_{benign}$.
Step 5 - Create the Retraining Set: Label the new adversarial curriculum as positive class ($\mathcal{Y}^{curr}_{adv} \leftarrow 1$) and the benign replay buffer as benign ($\mathcal{Y}_{benign} \leftarrow 0$). Form

$$X_{retrain} \leftarrow \text{concat}(X_{train}, \mathcal{X}^{curr}_{adv}, \mathcal{B}_{benign}),$$
$$Y_{retrain} \leftarrow \text{concat}(Y_{train}, \mathcal{Y}^{curr}_{adv}, \mathcal{Y}_{benign}).$$

Step 6 - Fine-Tune the NIDS: Fine-tune f_θ on $(X_{retrain}, Y_{retrain})$ with a low learning rate to obtain a candidate hardened model $f_{\theta'}$.
Step 7 - Guard-Railed Validation: Evaluate $f_{\theta'}$ on (X_{val}, Y_{val}) and on the disjoint adversarial holdout $X^{adv}_{holdout}$.
if F1 improves while FPR remains bounded **then**
    Promote: $f_{\theta\star} \leftarrow f_{\theta'}$
else
    Reject: $f_{\theta\star} \leftarrow f_\theta$

4.3. Threat analysis module: Anomaly detectors

This is an optional module that provides a second layer of defense. Each detector is trained exclusively on benign data to learn a fine-grained model of normalcy and thereby flag subtle deviations characteristic of adversarial samples. Because the detectors exhibit orthogonal biases (reconstruction versus isolation versus density/critic estimation), combining them helps reduce correlated blind spots. Their outputs are integrated into Algorithm 3 to directly penalize the RL agent when generated samples appear obviously anomalous. In practice, we aggregate detector alerts and cross-reference them with the NIDS confidence to prioritize investigations and mitigate operator fatigue. We periodically adapt detectors to concept drift via unsupervised retraining on a curated sliding window of recent benign traffic [49]. Detector quality is assessed using Area Under the ROC Curve (AUROC) and Area Under the Precision-Recall Curve (AUPRC) on held-out benign versus adversarial samples. We also include a cost breakdown: Isolation Forest (IF) trains the quickest at the least cost, the Autoencoder (AE) has a mid-level cost and GAN-based detectors are the most computationally expensive (see Section 5).

A key operational caveat is that adversarial-detection recall alone is insufficient for deployment decisions. In production, detector precision, benign false-positive rate and alert volume are equally important because they determine analyst workload, alert fatigue and the risk of excluding legitimate samples from retraining. To address this, we conducted a dedicated benign-traffic evaluation of all four TAM detectors, measuring precision, recall, FPR and AUPRC on a held-out benign partition; the results are reported in Table 15 in Section 5. These metrics provide a more complete picture of detector fitness for deployment and support threshold-selection decisions that go beyond maximizing adversarial detection rate alone. Detector thresholds are not chosen to maximize adversarial detection rate (ADR) alone. Instead, they are calibrated on a held-out benign/adversarial validation partition to meet an *operational benign-FPR target* while preserving as much adversarial recall as possible. For the Autoencoder, the threshold is set at the 95th percentile of benign reconstruction errors; for WGAN-GP, it is set at the 5th percentile of critic scores on benign data; Isolation Forest uses its built-in contamination parameter; and the vanilla GAN discriminator uses a percentile-matched threshold analogous to the AE. EVT-based dynamic thresholds were evaluated as an architectural option but were not applied in the primary reported experiments; all results in Tables 11 and 15 use the fixed-percentile thresholds described above. The resulting trade-offs are quantified in Table 15: WGAN-GP achieves the best trade-off (0.04 benign FPR with 0.99 ADR), while IF has the lowest benign FPR (0.06) at the cost of reduced adversarial recall (0.61). AE and GAN exhibit benign FPR of 0.19 and 0.22, respectively and would require tighter threshold calibration before production deployment.

To connect detector FPR to practical analyst workload, consider a deployment processing 1,000,000 benign flows per day. At WGAN-GP's benign FPR of 0.04, this yields approximately 40,000 false alerts daily, which is clearly too high for unassisted human review. Tightening the threshold to achieve a 0.1% FPR would reduce the volume to roughly 1,000 alerts per day, which may be manageable with prioritized triage. These estimates are illustrative and deployment-dependent. Exact volumes would vary with traffic mix, threshold calibration and analyst capacity. They also show that the human-in-the-loop vetting option (Section 4.5) cannot scale without automated triage mechanisms such as confidence-gated routing or active-learning sample selection, as noted in Section 7. Measuring alert-volume characteristics under sustained benign traffic remains a priority for follow-on work.

The Autoencoder (AE) detector is an unsupervised neural network trained to reconstruct benign input data, with anomaly score given by the mean-squared reconstruction error

$$S_{AE}(x) = \frac{1}{d}\sum_{i=1}^{d}(x_i - \hat{x}_i)^2 = \frac{1}{d}\,\|x - D(E(x))\|_2^2, \tag{2}$$

in which $\|\cdot\|_2$ denotes the Euclidean (L_2) norm, $E(\cdot)$ is the encoder mapping input $x \in \mathbb{R}^d$ to a compressed latent representation $z \in \mathbb{R}^b$ with $b \ll d$ and $D(\cdot)$ is the decoder reconstructing $\hat{x} \in \mathbb{R}^d$ from z. Architecturally, we employ a symmetric encoder/decoder composed of dense layers, where the encoder progressively compresses the feature space down to a low-dimensional bottleneck and the decoder mirrors this structure to reconstruct the original input. This reconstruction-based formulation assumes that only benign data is seen during training, enabling the AE to learn the low-dimensional manifold of normal patterns. Anomalous or malicious inputs are expected to deviate from this manifold and hence produce larger reconstruction errors. To further enhance sensitivity to rare but important deviations, we may optionally include a sparsity regularization term, such as the Kullback-Leibler (KL) divergence between the average hidden activations and a target sparsity level ρ, which penalizes overly active latent units and encourages compact, interpretable encodings:

$$\mathcal{L}_{\text{KL}} = \sum_{j=1}^{b} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{\hat{\rho}_j},$$

where $\hat{\rho}_j$ is the average activation of the jth latent unit over the training set. Operationally, a high reconstruction score $S_{AE}(x)$ suggests nonconformity to the benign distribution. We set a detection threshold by computing the 95th quantile of benign reconstruction scores and flagging samples with higher scores as anomalous. To make this more robust, especially under distributional drift, we also experiment with Extreme Value Theory (EVT) [50] by fitting a Generalized Pareto Distribution (GPD) to the tail of the reconstruction error distribution and dynamically adjusting the decision threshold based on its estimated

parameters. This makes AE detectors more resilient to unexpected but benign (legitimate) outliers and reduces false positives under concept drift.

The Isolation Forest detector rests on the premise that anomalies are easier to isolate than normal points. It builds an ensemble of isolation trees using random feature choices and random split values, recursively partitioning the feature space. Each tree isolates a sample by randomly choosing a feature and a split value until the sample is isolated in a leaf node. The anomaly score is based on the average path length required to isolate the input. Formally, for a sample x, the expected path length $E[h(x)]$ over all trees in the forest is used to compute the normalized anomaly score:

$$S_{IF}(x) = 2^{-\frac{E[h(x)]}{c(n)}},$$

where $c(n)$ is the average path length in a binary tree of size n, used to normalize the depth. Shorter path lengths (i.e., earlier isolation) result in scores closer to 1, signaling a higher likelihood of anomaly. This makes Isolation Forest a computationally light component that is ideal for streaming and resource-constrained deployments. It does not require prior knowledge of the data distribution and is highly interpretable due to its tree structure.

Both the standard GAN and the Wasserstein GAN with Gradient Penalty (WGAN-GP) detectors are trained on benign-only data (features scaled to $[-1, 1]$), after which the discriminator (or the critic in the case of WGAN-GP) is repurposed as an anomaly detector that assigns higher scores to in-distribution benign samples and lower scores to off-manifold or adversarial inputs. For WGAN-GP [51], stability is enhanced by augmenting the critic loss with a gradient-penalty term enforcing the 1-Lipschitz constraint:

$$\mathcal{L}_{GP} = \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\Big[\big(\|\nabla_{\hat{x}} C(\hat{x})\|_2 - 1\big)^2\Big]. \tag{3}$$

We set $\lambda = 10$ following Gulrajani et al.[51]; this keeps the GP term dimension-consistent with the adversarial loss.

Here, $\hat{x}$ denotes points linearly interpolated between real and generated samples, i.e., $\hat{x} = \epsilon x + (1-\epsilon)G(z)$ for $\epsilon \sim U[0, 1]$. This penalty stabilizes training by ensuring that the gradient norm remains close to 1, which is necessary for WGAN-GP's theoretical convergence. The anomaly score for a new sample is the critic output:

$$S_{\text{WGAN-GP}}(x) = C(x),$$

with lower values indicating deviations from the training (benign) distribution. Samples with scores below a chosen threshold (e.g., the 5th percentile of $C(x)$ on the benign set) are flagged as anomalous.

Compared to classic GANs that minimize Jensen-Shannon divergence, WGAN-GP minimizes the Wasserstein distance, providing a more stable signal, especially when data lies on low-dimensional manifolds or exhibits sharp boundaries. This model aligns with the adversarial training loop in Algorithm 3, where the reward component R_{detector} in line 10 of the algorithm reflects whether anomaly detectors (like AE, IF, or WGAN-GP) raise an alert, enforcing adversarial stealth during attack simulation. The use of multiple detectors, each sensitive to different aspects of the data distribution, enhances defense diversity and robustness. In our pipeline, the integration of these detectors complements the supervised NIDS classifiers by capturing subtle deviations that are not easily separable in label space, with AE and WGAN-GP being sensitive to manifold structure, while Isolation Forest targets outlier isolation in feature space, thereby offering orthogonal detection strengths.

Algorithm 3 Reward calculation in the adversarial environment.

1: **Input:** Current state (sample) s_t, Action (perturbation) $a_t = \delta_t$
2: **Parameters:** Perturbation penalty weight $w_p = 0.1$, Evasion threshold $\tau_{evade} = 0.5$
3: $s_{t+1} \leftarrow \text{clip}(s_t + \delta_t, 0, 1)$ ▷ Apply perturbation and clip
4: ▷ Get predictions from defender components
5: $P_{mal} \leftarrow$ NIDS.predict_proba(s_{t+1})
6: $D_{anomaly} \leftarrow$ Detector.predict(s_{t+1}) if detector exists, else 0
7: ▷ Calculate reward components
8: $R_{evasion} \leftarrow 1.0 - P_{mal}$ ▷ Reward for lower score
9: $R_{penalty} \leftarrow w_p \cdot ||\delta_t||_2$ ▷ Penalty for large perturbation
10: $R_{detector} \leftarrow -1.0$ if $D_{anomaly} = 1$ else 0.1 ▷ Penalty for TAM detection
11: $R_{base} \leftarrow R_{evasion} - R_{penalty} + R_{detector}$
12: ▷ Check for terminal state and apply final bonus
13: **if** $P_{mal} < \tau_{evade}$ **then**
14: $R_{final} \leftarrow R_{base} + 5.0$ ▷ Success bonus
15: $done \leftarrow$ True
16: **else**
17: $R_{final} \leftarrow R_{base}$
18: $done \leftarrow$ False
19: **Return** $s_{t+1}, R_{final}, done$

4.4. NIDS classifiers

The framework includes a diverse set of NIDS models, each with distinct architectural properties. Hyperparameter optimization is automated using established libraries to ensure each model performs optimally.

The Artificial Neural Network (ANN) and Deep Neural Network (DNN) models are feedforward neural networks whose architectures are selected with `KerasTuner` [52] using a *RandomSearch* over a predefined hyperparameter space: for the ANN this covers the units of two dense layers and a dropout rate, while for the DNN it expands to the number of hidden layers (2–5), units per layer, dropout rates and learning rates; both use ReLU activations in hidden layers and a final sigmoid neuron that outputs the probability of the malicious class and both are trained with the Adam optimizer [44] to minimize binary cross-entropy. Concretely, a feedforward network computes

$$h^{(l)} = \phi\big(W^{(l)} h^{(l-1)} + b^{(l)}\big), \quad l = 1 \dots L, \quad h^{(0)} = x,$$

where $h^{(l)}$ is the output of the lth layer, $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector, respectively, for layer l and $\phi(\cdot)$ denotes the nonlinear activation function ReLU in our case, defined as $\phi(z) = \max(0, z)$. The input vector x is propagated through each hidden layer $l = 1, \dots, L$, transforming the feature representation at each step. The final output layer computes

$$\hat{y} = \sigma\big(W^{(L+1)} h^{(L)} + b^{(L+1)}\big) = P(y = 1 \mid x),$$

where $\sigma(\cdot)$ is the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$, producing a scalar in the range $[0, 1]$, interpretable as the probability that the input belongs to the positive class (e.g., a malicious traffic sample).

The training objective is to minimize the binary cross-entropy loss:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\big[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\big],$$

where $y_i \in \{0, 1\}$ is the ground-truth label and $\hat{y}_i$ is the predicted probability for the ith sample in a batch of size N. This loss penalizes incorrect classifications more severely when the model is confident but wrong. To address class imbalance, we optionally apply class weighting, which scales the loss for positive and negative classes differently.

Adam adapts per-parameter learning rates via exponential moving averages of the first and second gradient moments [44]; the full update equations are reproduced in Appendix B for reference. We use $\alpha \in [10^{-5}, 10^{-3}]$ (tuned), β_1=0.9, β_2=0.999, ϵ=10^{-8}.

Regularization includes dropout with $p \in [0.1, 0.5]$, which randomly sets a fraction p of input units to zero during training to prevent co-adaptation and ℓ_2 weight decay, which penalizes large weights to improve generalization by adding a term $\lambda \sum \|\theta\|^2$ to the loss.

We employ early stopping on validation F1 score with a patience of 10 epochs, which halts training if no improvement is seen, thereby avoiding overfitting. For DNNs, the search space spans:

- **Number of Layers** $\in \{2, 3, 4, 5\}$: Deeper models can learn more abstract representations but risk overfitting.
- **Units per Layer** $\in \{64, 128, 256, 512\}$: Larger widths increase model capacity.
- **Dropout Rate** $\in \{0.1, 0.3, 0.5\}$: Higher values increase regularization.

Random search is particularly efficient in such high-dimensional, conditional hyperparameter spaces, especially when interactions are nonlinear and unknown.

Weights in ReLU layers are initialized using He-normal initialization [45], which draws from a zero-mean Gaussian with standard deviation $\sqrt{2/n_{in}}$, preserving the variance of activations and ensuring stable gradients. Although sigmoid outputs provide calibrated-like scores, the operational decision threshold τ is not fixed at 0.5 but tuned on a validation set to maximize F1 score. This threshold tuning is critical in imbalanced settings where precision-recall trade-offs matter more than overall accuracy. For explainability, we compute permutation feature importance and SHAP (SHapley Additive exPlanations) values [53] after training. Permutation importance estimates the change in prediction error after shuffling a feature, while SHAP decomposes model output into contributions of each feature, allowing us to understand the decision logic of the model and verify that adversarial artifacts have not polluted retraining.

From a deployment perspective, models support Open Neural Network Exchange (ONNX) export for low-latency inference [54], enabling integration into production systems regardless of backend. Batch normalization was evaluated but omitted due to the small effective batch sizes in online retraining, which destabilizes its moving averages and can degrade performance. Historically, the lineage from McCulloch-Pitts neurons [55] to modern deep networks underscores why ANN/DNN remain flexible and strong baselines for NIDS. Our contribution lies in integrating these models into an active defense loop that allows the system to adapt to adversarial threats over time, rather than pursuing raw accuracy gains alone.

For classical ensemble baselines, we employ Random Forest [8] and XGBoost [9], tuning their hyperparameters with scikit-learn's `GridSearchCV` under 3-fold cross-validation to reach strong operating points. RF constructs an ensemble of T high-variance, low-bias decision trees trained on bootstrap samples. At each split a random subset of features (m_{try}) is considered and predictions are aggregated by majority vote for classification (or by averaging for regression). This bagging strategy markedly reduces variance, yielding robust generalization with comparatively light tuning; we sweep `n_estimators` (the number of trees) and `max_depth` (the maximum depth of individual trees), two key hyperparameters controlling ensemble size and expressivity. Each RF tree is trained on a bootstrap sample of the original data, introducing diversity that combats overfitting. The random selection of m_{try} features at each node split increases the independence among trees and lowers their correlation, further reducing variance.

Let $f_t(x)$ be the output of the tth tree; the aggregated classifier output is:

$$\hat{y} = \text{majority\_vote}\left\{\text{f}_\text{t}(\text{x})\right\}_{\text{t=1}}^{\text{T}},$$

where each tree f_t outputs a class label for input x. For interpretability, RF exposes Gini feature importance, calculated as the total reduction of the Gini impurity across all trees and nodes where a feature is used. However, since impurity-based importances can be biased under adversarial perturbations or correlated features, we cross-check with permutation feature importance, computed by measuring the increase in prediction error after permuting a feature's values.

XGBoost, in contrast, adds trees sequentially to correct residual errors, optimizing an objective that balances data fit and model complexity:

$$\text{Obj}(\theta) \;=\; \sum_i l\big(y_i, \hat{y}_i\big) \;+\; \sum_k \Omega(f_k), \tag{4}$$

where $l(y_i, \hat{y}_i)$ is the loss (e.g., logistic loss) and $\Omega(f_k)$ is the regularization term over trees f_k. At boosting round t, XGBoost minimizes the functional:

$$\mathcal{L}^{(t)} \;=\; \sum_i l\Big(y_i,\, \hat{y}_i^{(t-1)} + f_t(x_i)\Big) \;+\; \Omega(f_t), \tag{5}$$

where $\hat{y}_i^{(t-1)}$ is the accumulated prediction up to iteration $t-1$ and $f_t(x_i)$ is the newly added tree. To improve efficiency and guide tree construction, XGBoost employs a second-order Taylor approximation of the loss at each boosting round; the derivation is provided in Appendix B. Greedy split selection then uses cumulative first- and second-order gradient statistics (g_i, h_i) to compute the optimal leaf score at each node.

The regularization term is defined as:

$$\Omega(f) \;=\; \gamma\, T \;+\; \tfrac{1}{2}\lambda \sum_j w_j^2, \tag{6}$$

where T is the number of leaves in the tree, w_j is the score on leaf j, γ penalizes deeper trees (more leaves) and λ controls the ℓ_2 penalty on leaf weights. This explicit regularization discourages overly complex trees and large weights, which improves generalization.

We tune `n_estimators` (number of boosting rounds) and `learning_rate` (η), where a smaller η smooths model updates and requires more trees, effectively lowering overfitting risk and helping the model better absorb adversarial noise. In our imbalanced setting, class weights are adjusted via `scale_pos_weight` set to the ratio:

$$\texttt{scale\_pos\_weight} = \frac{\#\text{benign}}{\#\text{malicious}},$$

to upweight errors on the minority (malicious) class and improve recall.

While gradient-boosted trees attain high accuracy, their sharp partition boundaries can be sensitive to fine-grained continuous-feature perturbations, which can be exploited by adversaries. To counter this, our active retraining loop continually absorbs newly observed patterns to restore robustness and maintain adaptiveness in changing threat environments. For analyst support, we extract path-level rules from influential trees, converting decision paths into if-then-else statements that describe how the model arrived at a classification. This partial interpretability is useful for justifying critical alerts in operational settings. Computationally, XGBoost leverages histogram-based gradient approximations and multi-threaded execution to efficiently scale to large datasets. Likewise, scikit-learn's RF grows trees in parallel across CPU cores, significantly reducing training time. Compared to deep learning or reinforcement learning components (e.g., attacker agents), these ensembles are lightweight and fast to iterate, making them ideal for high-frequency retraining or on-device deployment within our broader defense pipeline.

The GAN-based NIDS (GAN-IDS) uses the discriminator of a GAN as the primary NIDS. Unlike GANs used for anomaly detection, which see only benign data, it is trained on the full dataset containing both classes. As a result, the discriminator learns to classify samples directly while being guided by an adversarial training process against its corresponding generator. Note that this supervised GAN-IDS classifier is distinct from the unsupervised GAN/WGAN-GP detectors in the TAM (Section 4), which are trained exclusively on benign data for anomaly detection. In effect, the classification task is framed as a minimax game in which the discriminator aims to correctly label both real samples (from the dataset) and fake samples (from the generator), a strategy related to semi-supervised GANs, yet applied here in a fully supervised context to leverage the adversarial dynamic for enhanced robustness rather than for learning from unlabeled data.

We adopt the classical GAN objective:

$$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))], \tag{7}$$

where $D(x)$ represents the discriminator's estimate that input x is real and $G(z)$ generates fake samples from latent vectors $z \sim \mathcal{N}(0, I)$. The discriminator D is trained to maximize the probability of assigning correct labels to real and fake inputs, while the generator G is trained to minimize the discriminator's ability to distinguish generated samples from real ones. This game-theoretic adversarial setup compels D to learn highly discriminative boundaries. In our supervised GAN-IDS setup, we

augment D with a classification head to output the probability of the true class label $y \in \{0, 1\}$. The total loss of the discriminator is given by:

$$\mathcal{L}_D = \mathcal{L}_{\text{adv}} + \lambda \mathcal{L}_{\text{sup}}, \tag{8}$$

where:

- $\mathcal{L}_{\text{adv}} = -\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$ is the adversarial component,
- $\mathcal{L}_{\text{sup}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)\right]$ is the supervised binary cross-entropy loss over labeled data,
- λ is a tradeoff coefficient balancing adversarial realism against class discrimination.

The generator G is composed of dense (fully connected) layers with batch normalization to stabilize training and a final tanh activation that ensures generated features lie within a bounded range. The discriminator D mirrors standard DNN architectures, typically using shared hidden representations feeding into both the adversarial and classification outputs. This joint architecture enables D to extract generalizable and robust features through adversarial feedback. Training alternates k_D discriminator updates per generator update to maintain discriminator advantage and avoid generator collapse. We empirically use $k_D = 5$, based on the heuristic that a strong discriminator provides harder gradients for the generator to learn from. This strategy stabilizes convergence by preventing G from generating low-diversity outputs too early (mode collapse). To regularize D and reduce overconfidence, we use one-sided label smoothing, replacing target label 1 with 0.9 in real samples to make D less certain and avoid sharp decision boundaries. Additionally, Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is injected into real and fake inputs during training to simulate natural variability and mitigate overfitting. These tricks are known to prevent discriminator saturation and improve generalization. Convergence is monitored using multiple signals:

- The supervised F1 score on validation data,
- The Jensen-Shannon divergence, which is implicitly minimized by the adversarial loss and reflects the closeness between real and generated distributions,
- Diversity in G's outputs to detect and mitigate mode collapse,

and, if necessary, we adopt Wasserstein GAN with Gradient Penalty (WGAN-GP) techniques to restore gradient stability. WGAN-GP replaces the original binary loss with a Wasserstein-based distance metric and enforces a Lipschitz constraint using gradient penalty:

$$\mathcal{L}_{\text{WGAN-GP}} = \mathbb{E}_{\tilde{x}}\left[\left(\|\nabla_{\tilde{x}} D(\tilde{x})\|_2 - 1\right)^2\right],$$

which improves training when GANs are sensitive to high-dimensional input distributions. GAN-IDS improves robustness because the discriminator learns a richer and more defensive boundary by constantly confronting a generator that produces hard negatives. This intrinsic adversarial training complements exposure to external RL attacks and improves robustness under distribution shift. In contrast to traditional supervised classifiers that only learn from fixed training data, GAN-IDS benefits from an evolving curriculum of adversarial samples. A risk is that if G overfits the training distribution, D may underfit new adversarial modes. Our active retraining loop captures such modes and feeds them back into training, closing the gap. The loop ensures that the generator continues to produce relevant adversarial examples and that the discriminator adapts in real time to unseen attack behaviors. During inference only D is used, with the generator frozen unless future adversarial augmentation is desired to refresh the hard-negative curriculum. This makes deployment lightweight and keeps inference time comparable to that of traditional DNN classifiers. In terms of computational complexity, training GAN-IDS is more demanding than training a plain DNN because it requires the optimization of two networks and coordinated updates, but it remains cheaper than full RL attackers or other dynamic defense models. The memory footprint is manageable at our feature dimension d and we parallelize training across GPU cores.

Compared with autoencoder (AE) detectors that rely on reconstruction error and implicitly assume malicious patterns are simply "far" from benign in Euclidean space, GAN-IDS leverages adversarial generation rather than reconstruction, excelling when malicious patterns are complex, multi-modal, subtle, or carefully crafted to hide within benign manifolds. It is thus particularly suited for modern NIDS tasks where attacker strategies continually evolve and evade static boundaries.

4.5. Active defense mechanism

The adaptive retraining loop forms the core of our framework's active defense mechanism, enabling the system to learn from new adversarial attacks and update the NIDS model accordingly. Specifically, the core of the active defense is the adaptive NIDS retraining process (Batch Hardening) outlined in Algorithm 2. This mechanism ensures that the NIDS learns from the attacker's successful strategies, progressively hardening its decision boundary.

The following points explain the active defense mechanism:

(a) *Trigger Policies:* We can trigger retraining after N_{fail} evasions or on a schedule (e.g., daily). In experiments, $N_{fail} = 20{,}000$ balanced responsiveness against computational cost.
(b) *Curriculum of Adversarial Samples:* We stratify $\mathcal{R}_{adv}$ by perturbation norms to avoid biasing the NIDS toward extreme samples while ignoring subtle ones.
(c) *Catastrophic Forgetting:* Fine-tuning can degrade performance on benign traffic. We monitor benign FPR; if it drifts above a threshold, we add a benign replay buffer to the retraining batch.
(d) *Learning Rate Scheduling:* We use a smaller learning rate (e.g., 1e-5) during hardening to prevent overwriting earlier learned representations.
(e) *Alternative Strategies:* Instead of full retraining, we explored elastic weight consolidation (EWC) and adapter layers to reduce compute costs, which remain key areas for future work (see Section 7).
(f) *Stopping Criteria:* Retraining stops when the validation F1 score on an adversarial holdout set is ≥ 0.99 or after a maximum of 50 epochs. Early stopping is used to reduce cost.
(g) *Logging and Auditing:* Every adversarial sample used for retraining is stored with its hash and RL trajectory metadata for forensic analysis and backtracking.
(h) *Human-in-the-Loop Option:* The framework supports an optional step where security analysts can vet a subset of $\mathcal{X}_{adv}$ to ensure correct labeling, preventing accidental model poisoning.
(i) *Multiple Defender Models:* The system can maintain an ensemble of defender models and rotate them in a Moving Target Defense (MTD) style, making the defense less predictable.
(j) *Mathematical View:* The retraining process solves the optimization problem:

$$\theta' = \arg\min_{\theta_{\text{cand}}} \Big[\mathcal{L}_{orig}(X_{train}, Y_{train}; \theta_{\text{cand}}) + \lambda \mathcal{L}_{adv}(\mathcal{X}_{adv}, \mathbf{1}; \theta_{\text{cand}})\Big].$$

where λ is a hyperparameter that weights the importance of the new adversarial samples and θ_{cand} represents the candidate parameters being optimized, which result in the hardened model parameters θ'.

4.6. Integrated end-to-end workflow

The components described previously are integrated into a cohesive, end-to-end workflow that simulates a co-evolutionary arms race between the attacker and the defender. This process methodically moves from establishing a baseline defense to subjecting it to intelligent attack and finally to leveraging those attacks to build a hardened, more resilient system. The entire pipeline can be broken down into five distinct phases, which are formalized in Algorithm 4.

Phase1 **Setup and Baseline Training:** The process begins with the foundational DReLAB dataset. This data is rigorously preprocessed (cleaned, scaled and dimensionality-reduced via PCA) and then partitioned into distinct training, validation and test sets to prevent data leakage. The initial, static NIDS classifier (f_θ) is trained on the training set until it achieves optimal performance on the validation set, measured primarily by the F1 score. In parallel, if a Threat Analysis Module is used, the anomaly detectors ($D_{anomaly}$) are trained exclusively on benign samples from the training set to learn a model of normal network behavior. This phase establishes the baseline performance of a conventional, static defense before any adversarial interaction occurs.

Phase2 **Adversarial Attack Simulation:** With the baseline NIDS (f_θ) in place, the adversarial simulation begins. A selected reinforcement learning agent (π_{attack}) is unleashed against the NIDS. The NIDS acts as the environment for the RL agent, providing a reward signal based on the agent's ability to perturb malicious samples and cause them to be misclassified as benign (i.e., driving the malicious probability score below the evasion threshold τ_{evade}). The agent is trained for a significant number of episodes, allowing its policy to mature from random exploration to a sophisticated, targeted strategy for generating effective adversarial samples.

Phase3 **Performance Degradation and Evasion Collection:** The effectiveness of the mature attacker policy is measured by evaluating the baseline NIDS against a new set of adversarial samples generated by the attacker. As documented in our results, this typically leads to a catastrophic collapse of the NIDS's performance, with the F1 score for malicious traffic dropping to near-zero. Crucially, every successful adversarial sample (x'_{mal}) that evades the NIDS is collected and stored in a dedicated repository, $\mathcal{X}_{adv}$. This repository forms the curriculum for the active defense phase.

Phase4 **Active Defense and Model Hardening:** This phase embodies the core of our active defense mechanism. The collected set of successful evasions, $\mathcal{X}_{adv}$, is labeled as malicious and concatenated with the original training dataset, (X_{train}, Y_{train}). This creates a new, hardened training set, $(X_{retrain}, Y_{retrain})$, which contains both the original data and the specific patterns the attacker learned to exploit. The NIDS model is then retrained on this augmented dataset, typically using a lower learning rate to fine-tune its decision boundary without catastrophically forgetting the original patterns. This step directly hardens the model ($f_\theta \rightarrow f_{\theta'}$) against the observed attack vectors.

Phase5 **Final Evaluation and Recovery Assessment:** In the final phase, the newly hardened NIDS model, $f_{\theta'}$, is evaluated on two separate pools: (i) the untouched canonical test split for final Accuracy and F1 reporting and (ii) a fresh attacker-generated evaluation pool for post-hardening robustness assessment. The performance metrics (Final F1, Final Accuracy) are calculated and compared to both the initial baseline and the degraded performance. The success of the framework is quantified by the degree to which the NIDS recovers its ability to correctly classify malicious traffic, ideally returning to its initial high-performance levels. This five-phase process provides a complete, reproducible methodology for quantifying the vulnerability of static systems and the efficacy of an active defense response.

Fig. 2 illustrates the five-phase pipeline:

1. A baseline NIDS is trained on the cleaned DReLAB corpus.
2. A reinforcement-learning attacker evolves its policy against that detector.
3. Successful evasions are harvested into the curriculum set $\mathcal{X}_{\text{adv}}$.
4. The NIDS is fine-tuned on the augmented dataset, producing a hardened model $f_{\theta'}$.

Algorithm 4 Complete active defense and hardening loop.

Input: Raw dataset D_{raw}, NIDS architecture, optional TAM architecture, RL attacker family
Output: Promoted hardened model $f_{\theta\star}$ and evaluation metrics
// **Phase 1 - Setup, Split and Baseline Training**
$(X_{train}, Y_{train}), (X_{val}, Y_{val}), (X_{test}, Y_{test}), X^{adv}_{holdout} \leftarrow$ PreprocessAndSplit(D_{raw})
Fit all stateful preprocessors on X_{train} only; apply them unchanged to X_{val}, X_{test} and $X^{adv}_{holdout}$
Train baseline NIDS f_θ on (X_{train}, Y_{train}) with validation on (X_{val}, Y_{val})
Initial_F1, *Initial_Acc* $\leftarrow$ Evaluate f_θ on the untouched canonical test split (X_{test}, Y_{test})
if using TAM **then**
    Train anomaly detector $D_{anomaly}$ on benign samples from X_{train}
// **Phase 2 - Adversarial Attack Simulation**
Initialize RL attacker π_{attack} and define environment 'Env' using f_θ and optional $D_{anomaly}$
Train π_{attack} in 'Env' for $N_{attack\_episodes}$ to obtain a mature policy
// **Phase 3 - Model-Specific Degradation and Evasion Collection**
Generate a fresh attacker-specific degradation pool from a disjoint partition of unseen positive-class seeds
Degraded_F1 $\leftarrow$ Evaluate f_θ on the attacker-generated degradation pool
Route successful evasions through the TAM (if enabled) and store accepted samples in repository $\mathcal{R}_{adv}$
// **Phase 4 - Guard-Railed Hardening**
Build a stratified adversarial curriculum $\mathcal{X}^{curr}_{adv}$ from $\mathcal{R}_{adv}$ and sample benign replay buffer $\mathcal{B}_{benign}$
Fine-tune f_θ on $(X_{train} \cup \mathcal{X}^{curr}_{adv} \cup \mathcal{B}_{benign})$ with low learning rate to obtain candidate $f_{\theta'}$
Evaluate $f_{\theta'}$ on (X_{val}, Y_{val}) and on $X^{adv}_{holdout}$
if F1 improves and FPR remains bounded **then**
    Promote $f_{\theta\star} \leftarrow f_{\theta'}$
else
    Retain $f_{\theta\star} \leftarrow f_\theta$
// **Phase 5 - Final Evaluation**
Final_F1, *Final_Acc* $\leftarrow$ Evaluate $f_{\theta\star}$ on the untouched canonical test split (X_{test}, Y_{test})
Robust_F1 $\leftarrow$ Evaluate $f_{\theta\star}$ on a fresh attacker-generated robustness pool
Return $f_{\theta\star}$, (*Initial_F1*, *Degraded_F1*, *Final_F1*, *Robust_F1*)

5. The hardened model is re-evaluated on clean and adversarial traffic and may feed into an additional loop (dashed arrow) for continued hardening.

5. Experimental setup and evaluation

This section provides a structured overview of the experimental design and findings. First, we define the metrics and hardware used throughout, covering robustness and error-balance metrics (Accuracy, Precision, Recall, F1, FPR, FNR, MCC), attacker and perturbation signals (ASR, ℓ_2, ℓ_∞, query count) and cost measures in time and energy, together with the exact CPU/GPU platform and software stack; formal definitions appear in Table 5 and Section 5.1. Second, we establish the baseline by confirming the high clean-traffic performance of GAN-IDS, DNN and XGB (Accuracy $\approx 0.99995 - 1.0000$, F1 $\approx 0.9995 - 1.0000$), which serves as the reference for all subsequent degradation and recovery analyses and is summarized in Table 6. Next, we introduce the RL attackers (A2C, PPO, DDQN, SAC) and show that the positive-class F1 score on attacker-generated degradation pools collapses to 0.0000, with false negatives surging into the tens of thousands. This degradation must be interpreted separately from the near-perfect accuracy on the canonical

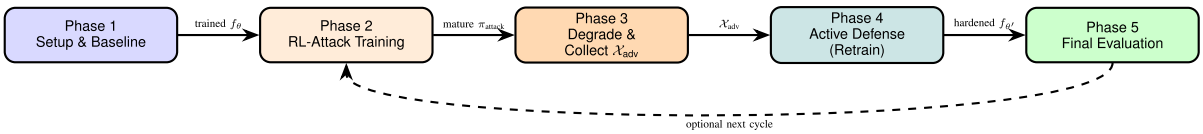

Fig. 2. Integrated end-to-end workflow for the co-evolutionary active-defense loop. Each colored block corresponds to one phase of Algorithm 4; solid arrows show the forward progression, while the dashed arrow indicates that the hardened model can seed another iteration of adversarial training and hardening.

Table 5
Evaluation metric definitions.

Metric	Definition / Rationale
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$; misleading when classes are imbalanced.
Precision	$TP/(TP+FP)$; "how many flagged were truly malicious".
Recall (TPR)	$TP/(TP+FN)$; "how many malicious were caught".
F1 score	Harmonic mean of Precision and Recall.
FPR	$FP/(FP+TN)$; false alarms per benign.
FNR	$FN/(TP+FN)$; missed malicious.
MCC	Balanced correlation-like measure, robust to imbalance [56].
ASR	# successful evasions / total episodes.
ℓ_p Norm	Magnitude of perturbation; we report ℓ_2, ℓ_∞.
Query Count	Number of NIDS + detector queries per episode (proxy for stealth).
Energy (kWh)	Energy consumed, derived from logged power sensors.
Time (s)	Wall-clock attack/defense runtime for cost analysis.

untouched test split. An *accuracy illusion* is documented per pair in Table 8 and aggregated in Table 7, with cost-damage trade-offs visualized in Figs. 3 and 4.

We then evaluate the complete active defense by combining the optional TAM with adaptive retraining, quantifying detector-attacker interactions and the ensuing cost-benefit landscape in Table 11 and Fig. 5, reporting top hardened configurations in Table 10 and illustrating collapse-and-recovery trajectories in Fig. 6. We also provide a focused study of the most efficient attacker (A2C) against the most capable detector (WGAN-GP) with per-NIDS outcomes in Table 12. Finally, Fig. 7 shows the attacker learning dynamics, reinforcing the need for a continuously learning defense and we provide code, configurations, dataset hashes and trained artifacts upon reasonable request, with public release upon acceptance, to enable exact replication and cost-aware comparison.

5.1. Evaluation metrics and hardware setup

We report a rich set of metrics to capture robustness, efficiency and operational cost, as formally defined in Table 5. Each metric is paired with a plain-language purpose so that security engineers can quickly map numbers to operational risk. For example, when attacks flood the system with undetected malicious flows, Recall plunges and FNR spikes; when operators are overwhelmed with benign alerts, Precision falls and FPR rises. MCC is included specifically to resist class imbalance, which is common in NIDS settings where benign traffic dominates. We also record attacker cost signals (query count, perturbation norm) and real-world operational cost (time and energy) to support cost-aware threat modeling and capacity planning.

All experiments were conducted on a consistent hardware and software platform to ensure reproducibility.

- **Hardware:** Intel Xeon E5-2698 v4 CPU (2.20GHz), 128GB RAM and an NVIDIA Tesla V100 GPU (16GB HBM2).
- **Software:** Python 3.8, PyTorch 1.12, TensorFlow 2.10, Scikit-learn 1.1 and Stable-Baselines3 1.6.
- **Energy Measurement:** Power was logged using the NVIDIA Management Library (NVML) for the GPU and `pyRAPL` for the CPU at a sampling rate of 1 Hz. Reported energy values (kWh) represent the total energy consumed above the baseline idle draw over the duration of the experiment.
- **Reproducibility:** All code, configuration files, dataset hashes and trained model artifacts are available from the corresponding author upon reasonable request and will be released in a public repository upon acceptance to facilitate replication of our results.

5.2. Baseline performance of static NIDS

To ground the subsequent attack and hardening results, we first establish a static baseline for the three NIDS backbones used throughout the aggregate attacker study: GAN-IDS, DNN and XGB. Random Forest (RF) is included in the broader detector-assisted hardening sweeps reported later, but the attacker-only aggregate comparisons are kept aligned with the three-model tables used to summarize the static baselines. Each model is trained once on the canonical training split and evaluated on the canonical untouched test split of Table 3. These scores constitute the "Initial" stage referenced in later sections. Although two models achieve error-free performance on this split, these results should be interpreted as strong separability on the DReLAB-derived corpus under the stated leakage-safe protocol, not as evidence that the task is universally solved. DReLAB is a purpose-built benchmark in which adversarial botnet flows are distinguishable from benign traffic at the feature level by construction [46]. The high baseline is also consistent with the near-perfect detection rates reported in other DReLAB-based studies prior to adversarial perturbation. The GAN-IDS model implements our *supervised* GAN-IDS: the discriminator acts as the primary classifier while the generator supplies hard negatives during training, yielding a robust decision boundary through the adversarial learning dynamic. The DNN detector is a supervised feed-forward network optimized with cross-entropy and Adam and regularized with dropout. With direct supervision and sufficient capacity, it reaches near-perfect separation on clean data. The XGB detector is an Extreme Gradient Boosting ensemble trained on the same feature set, providing a strong tree-based baseline with fast inference and high interpretability.

On the canonical untouched test split, GAN-IDS and DNN achieve error-free classification, while XGB remains near-perfect. The XGB residual counts (FP = 10, FN = 10 out of 406,020 test samples) yield Acc ≈ 0.99995 and F1 = 0.9995. **On the integrity of the reported baseline:** Four independent lines of evidence argue against train-test leakage. (i) *Non-zero XGB errors*: XGB reports FP = 10, FN = 10 on 406,020 test samples; a tree ensemble evaluated on its own training data would produce FP = FN = 0. By construction, these residuals are a direct fingerprint of genuine held-out evaluation. (ii) *XGB residuals confirm out-of-sample testing*: had the test split overlapped with the training data, gradient-boosted trees would be expected to achieve zero training error. The ten non-zero residuals observed across more than 400K samples therefore support the claim that evaluation was performed out of sample. (iii) *RF imperfection*: the Random Forest model in Table 10 exhibits non-perfect initial F1 in several attacker pairings, independently corroborating held-out evaluation across a second model family. (iv) *Attack vulnerability is proof-by-contradiction*: a model evaluated on its own training data would exhibit memorized, attack-resistant boundaries; instead, every attacker, NIDS pair drives malicious-class F1 to 0.0000, a collapse only

Table 6
Baseline (pre-attack) performance per NIDS on the canonical untouched test split (406,020 samples). These canonical values define the "Initial" stage used throughout the paper. All models were trained on the training split only; no hyperparameter tuning, preprocessing fitting, attacker generation, or retraining used the test split. Models with zero errors achieve error-free test performance under this corpus, while XGB exhibits small residual FP/FN resulting in near-perfect metrics.

NIDS	Acc	F1	FP	FN	Notes
GAN-IDS	1.0000	1.0000	0	0	Supervised GAN-IDS (discriminator classifier)
DNN	1.0000	1.0000	0	0	Supervised feed-forward network, CE loss, Adam
XGB	0.99995	0.9995	10	10	Gradient-boosted trees (XGBoost)

Table 7
Per-attacker averages across GAN-IDS, DNN and XGB. The first metric column reports mean post-attack F1 on attacker-generated degradation pools, whereas the second metric column reports mean baseline accuracy on the untouched canonical test split before hardening. All attackers drive post-attack F1 to zero, so the only separation is cost. A2C achieves the same destructive effect in a fraction of the time and energy.

Attacker	Mean Post-Attack F1 (Degradation Pool)	Mean Baseline Acc (Canonical Test)	$\overline{Time}$ (s)	$\overline{Energy}$ (kWh)
A2C	0.0000	0.9999	1047.83	0.1106
DDQN	0.0000	0.9999	6027.39	0.6361
PPO	0.0000	0.9999	3679.19	0.3883
SAC	0.0000	0.9999	15881.14	1.6760

possible when the attacker exploits a boundary that genuinely generalizes to unseen data. The resulting baseline, which we use as the reference for degradation and recovery analyses, is summarized in Table 6.

5.3. Attacker efficiency investigation

This section reports the aggregated performance of attackers, macro-averaged across the three static NIDS (i.e., GAN-IDS, DNN and XGB) as summarized in Table 7 and computed from Table 8. For each attacker a and metric m, we compute the unweighted arithmetic mean

$$\bar{x}_{a,m} = \frac{1}{3} \sum_{n \in \{\text{GAN-IDS,DNN,XGB}\}} x_{a,n,m},$$

where $x_{a,n,m}$ is the value of metric m for attacker a evaluated against NIDS n. No weighting, scaling, or normalization is applied; each NIDS model contributes equally. This choice corresponds to a macro-average over NIDS and prevents any single NIDS model (for example, one with a larger sample size or shorter runtime) from dominating the aggregate. Because each of the three NIDS reports near-perfect pre-retrain accuracy, the aggregated accuracy before retraining is approximately 0.9999. As a result, accuracy provides essentially no discrimination among attackers in this setting. The only meaningful separation arises in the resources consumed to obtain the same effectiveness, namely execution time and energy, which are visualized in Figs. 3 and 4. All four attackers drive the post-attack F1 score to zero on this NIDS set, making their *effectiveness* indistinguishable; the decisive criterion is therefore *efficiency*.

- **A2C** is the most efficient, achieving the lowest average execution time (1047.83 s) and the lowest mean energy consumption (0.1106 kWh). For adversaries seeking maximum damage per unit cost, **A2C is the preferred attacker**. This implies that comparatively simple RL agents can be highly destructive and should be incorporated into baseline threat models.
- **PPO** and **DDQN** form a middle tier. Both attain the same F1 collapse, but PPO requires less time and power than DDQN, making it the better efficiency choice between the two.

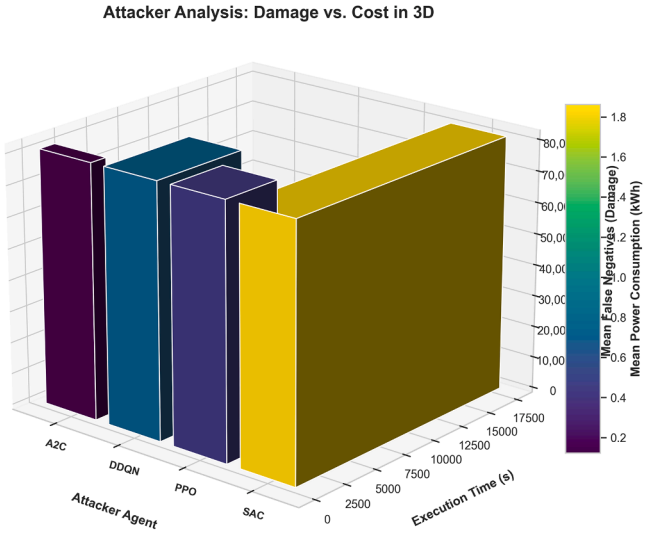

Fig. 3. Three-dimensional analysis of attacker performance showing the trade-off between impact (Mean False Negatives), execution time (s) and mean power consumption (kWh). All attackers inflict high damage; their primary differentiator is the cost to do so.

- **SAC** is the most expensive in both runtime and energy while providing no advantage in post-attack F1 and is therefore the least attractive option when resources matter.

When all attackers can reduce F1 to zero, the attack that does so with minimal time and energy is the most consequential. In our experiments this is **A2C**, underscoring the severity of threats posed by computationally modest reinforcement learning agents.

5.3.1. The vulnerability of static defenses under adversarial attacks

The results revealed a pronounced vulnerability of static NIDS to adaptive adversaries. When evaluated on attacker-generated degradation pools of unseen malicious/adversarial seeds, the number of **false negatives (FN)** grows into the tens of thousands for every attacker-NIDS pair and the malicious-class F1 score collapses to zero. This is distinct from the canonical untouched test split used for baseline and post-hardening evaluation. The broader lesson is an **accuracy illusion**: when severe class imbalance is present, a model can appear healthy under headline accuracy while failing catastrophically on the malicious class.

To present the aggregate behavior, for each attacker a and metric m we again report the macro-average across NIDS (the same way we did for the attacker), treating all NIDS models equally and applying no normalization. Table 8 provides a complete per attacker NIDS snapshot: accuracy before retraining, F1 score before and immediately after the attack and the false positive/false negative counts at both stages. Across rows, NIDS models that initially perform near perfectly (F1 Before ≈ 1) are compromised by adaptive attackers, with the F1 score collapsing to 0.0000 and *tens of thousands* of malicious flows evading detection (FN After). The final two columns quantify the attacker-side cost (time and energy). Fig. 3 visualizes the same pattern: while the damage axis (false negatives) is uniformly high across attackers, separation arises from the computational demands required to achieve that damage. In resource-constrained adversarial environments (for example, botnets deployed on limited hardware) **A2C-like attacks** are particularly realistic and dangerous because they deliver maximal evasion at minimal cost.

The principal conclusions are as follows. First, *perfect evasion in terms of F1* is observed: for every attacker NIDS combination, $F1_{\text{after}} = 0.0000$, indicating a complete collapse of both precision and recall on malicious traffic. Second, *tens of thousands of false negatives* appear in every case, as documented in Table 8 and Fig. 3. Third, headline accuracy alone is a

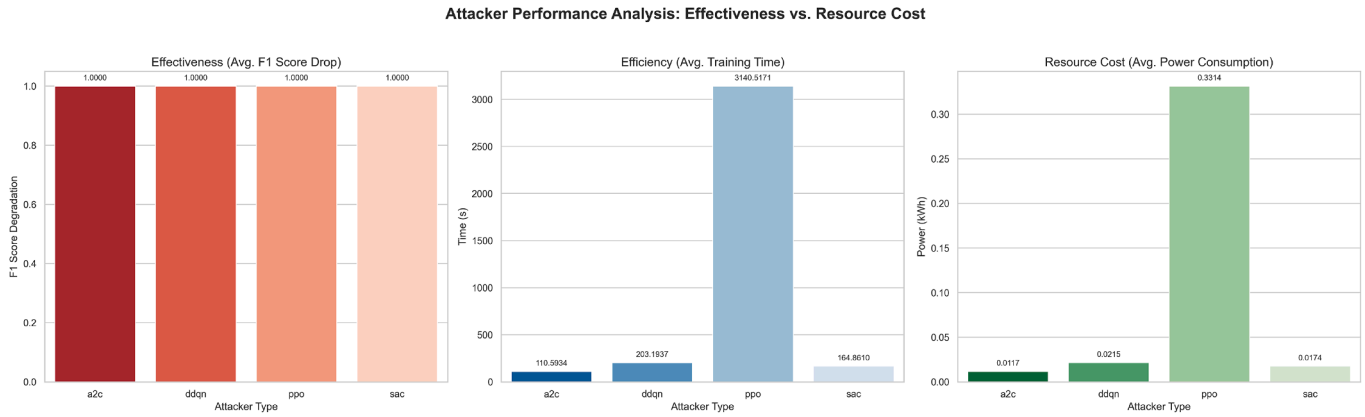

Fig. 4. Attacker effectiveness versus computational cost. The left panel shows that all attackers cause near-total degradation of the F1 score. The center and right panels show substantial variation in time and power required to achieve this effect, highlighting A2C as the most efficient.

poor robustness indicator under adaptive attacks, because it can conceal severe degradation in malicious-class recall and F1 when benign traffic dominates the evaluation mixture. Finally, there is a clear *efficiency gap* between attackers: Fig. 4 shows that **A2C** achieves full damage at the lowest cost, whereas **SAC** is orders of magnitude more expensive without providing additional impact.

The dramatic F1 collapse to zero under adaptive RL attack is itself empirical evidence against train-test leakage. A model evaluated on data it was trained on would exhibit memorized, attack-resistant boundaries; instead, every attacker-NIDS pair collapses completely, consistent with models generalizing from training data to a genuinely unseen distribution that adaptive adversaries can systematically erode. The RF configurations in Table 10 further corroborate this: several attacker pairings show non-perfect initial F1, independent of the DNN/GAN results.

These findings have direct design implications. Static NIDS implicitly assume a stationary environment, but adaptive attackers systematically exploit fixed decision boundaries until a detector becomes effectively inert. Defensive practice should therefore: (i) adopt **adversarially focused metrics** such as per-class precision/recall and explicit FP/FN counts on malicious subsets; (ii) incorporate **periodic or trigger-based retraining** or **rotating ensembles** to reduce predictability; and (iii) pursue **cost-aware threat modeling**, recognizing that when multiple attackers can achieve the same objective, the most consequential is the one that does so with the lowest operational cost, which in our evaluation is **A2C**.

Importantly, the present experiments show relative vulnerability and recovery under adaptive pressure; they do not claim that the same absolute error rates or recovery levels will transfer unchanged to operational networks. DReLAB is intentionally clean and adversarially structured, which makes it suitable for exposing failure mechanisms but likely inflates both baseline separability and post-hardening recovery compared with noisier, temporally richer enterprise traffic. The cross-dataset replication on CICIDS2017 confirms that A2C retains its efficiency advantage as the lowest-cost attacker and that the accuracy-illusion effect persists, while also showing that absolute F1 values at baseline and post-hardening are modestly lower on that corpus, consistent with the expectation that DReLAB provides a best-case separability scenario.

5.4. Detailed analysis of attackers and TAM-driven detector selection

A central component of our adaptive hardening pipeline is the **Threat Analysis Module (TAM)**. This module functions as a lightweight anomaly-detection front-end, strategically positioned to inspect all incoming traffic samples. Before a sample is added to the NIDS retraining buffer, the TAM acts as a critical pre-screening filter. If the sample is flagged as adversarial, it is routed to the adversarial repository and curated for hardening rather than allowed to pass as routine benign traffic. Because this module is not mandatory, primarily due to its associated computational cost, it is essential to precisely quantify the benefits it confers.

To achieve this, we evaluated the **final accuracy** of the hardened Network Intrusion Detection System (NIDS) under two distinct configurations. The results, presented in Table 11, show a detailed breakdown of these pairings. The Adversarial Detection Rate (ADR) reflects each detector's recall on its corresponding attacker's samples, while the delta (Δ) columns show the final impact on the NIDS's F1 score and accuracy compared to not using a detector. Costs are measured for the entire hardening loop. A positive Δ means the detector helped, while a negative Δ warns of harmful filtering (e.g., poisoning by omission). We visualize this cost-benefit analysis for the most efficient **A2C attacker** in Fig. 5. Overall, the granular results in Table 11 and Fig. 5 unveil three critical themes that govern the deployment of such a layered defense:

a) **Effective but Imperfect Detection is Sufficient.** GAN and WGAN-GP (and AE against all but the SAC attacker) generally achieve high ADRs, typically above 90%; AE's ADR falls to $\approx$ 60% against SAC. In contrast, the heuristic-based Isolation Forest (IF) struggles against the more complex perturbations, with its ADR hovering around 60% (58–62% depending on the attacker). Nonetheless, filtering even a fraction of adversarial samples is enough to positively influence the learning dynamics.

b) **Overhead Scales with Attacker Sophistication.** The computational cost of the TAM is highly dependent on the nature of the attack. This is most evident when defending against the A2C agent. As detailed in Table 11, the AE detector is the most effective in the A2C-specific case (99.0% ADR) and incurs negligible time and power cost, while WGAN-GP remains close in ADR (96.9%) and is retained for its stronger overall robustness against more sophisticated attackers such as SAC. This highlights a critical trade-off between attacker-specific efficiency and cross-attacker reliability.

c) **Critical Failure Modes Emerge with Advanced Agents.** Against the formidable SAC agent, the limitations of simpler detectors become a significant liability. Both the Autoencoder (AE) and Isolation Forest (IF) not only fail to detect a large portion of the attacks (ADRs of $\approx$60%) but also appear to cause **data poisoning by omission**. By failing to surface a sufficiently representative set of successful evasions, they deprive the model of the attack evidence needed for recovery and degrade its final performance (ΔF1 ≈ -0.20). In contrast, the GAN-family detectors (GAN, WGAN-GP), which more accurately model the data distribution, prove resilient to this failure mode.

Based on these findings, we offer the following practical guidance for deploying a TAM in a real-world adaptive defense system:

Table 8
Per-attacker NIDS metrics before and after attack. The "Before (Canonical Test)" columns report canonical baseline metrics on the untouched test split, whereas the "After (Degradation Pool)" columns report results on model-specific attacker-generated degradation pools of unseen positive-class seeds. Because each attacker is trained separately against each target NIDS, the degradation pool and thus the false-negative totals can differ across NIDS for the same attacker family. Accordingly, this table should not be interpreted as reusing training data or as applying one shared degradation pool to all models. A static NIDS with high initial F1 score suffers a catastrophic collapse in F1 after being targeted by a mature RL attacker, leading to tens of thousands of new false negatives. The final two columns report attacker cost.

Atk	NIDS	Before (Canonical Test)				After (Degradation Pool)			Cost	
		Acc	F1	FP	FN	F1	FP	FN	Time (s)	E (kWh)
A2C	GAN-IDS	1.0000	1.0000	0	0	0.0000	0	69 644	1118.32	0.1180
A2C	DNN	1.0000	1.0000	0	0	0.0000	14	87 055	2282.70	0.2409
A2C	XGB	0.9999	0.9995	10	10	0.0000	1	87 055	91.73	0.0097
DDQN	GAN-IDS	0.9999	0.9999	1	0	0.0000	4	69 644	17528.11	1.8498
DDQN	DNN	1.0000	1.0000	0	0	0.0000	32	87 055	2365.13	0.2496
DDQN	XGB	0.9999	0.9995	10	10	0.0000	1	87 055	198.06	0.0209
PPO	GAN-IDS	1.0000	1.0000	0	0	0.0000	0	69 644	3924.16	0.4141
PPO	DNN	1.0000	1.0000	0	0	0.0000	0	87 055	5199.59	0.5487
PPO	XGB	0.9999	0.9995	10	10	0.0000	0	87 055	3140.23	0.3314
SAC	GAN-IDS	0.9999	0.9999	1	0	0.0000	11	69 644	40904.92	4.3169
SAC	DNN	1.0000	1.0000	0	0	0.0000	0	87 055	10175.07	1.0738
SAC	XGB	0.9999	0.9995	10	10	0.0000	1	87 055	1857.13	0.1960

- To profile the adversary, the framework matches detector complexity to the capability of the expected threat. For attackers leveraging sophisticated, continuous control strategies like SAC, simple heuristic or reconstruction-based detectors are insufficient and potentially harmful. In these scenarios, robust density estimators like WGAN-GP are necessary. For example, WGAN-GP could be enabled continuously in high-risk windows (e.g., during campaigns), falling back to an AE + IF ensemble in low-risk windows to save energy while still filtering obvious attacks.
- No single detector model excels against all threats, so detector ensembles remain a sensible deployment option. An optimal strategy involves combining detectors that operate on different principles (e.g., a reconstruction-based AE with a density-based WGAN-GP) to cover complementary failure modes and provide a more robust defense against unforeseen attack variations.
- Although our experiment deliberately limited metrics to isolate variables, a production system requires comprehensive telemetry and monitoring. Logging true negatives and false positives on benign traffic is mandatory to calculate detector precision, correctly tune detection thresholds and mitigate alert fatigue for security operators. Without these statistics, apparent success may mask silent and potentially catastrophic failure.

Overall, our results support a cost-aware but security-first use of the TAM: several detectors yield gains, but **WGAN-GP is the most reliable overall detector**, combining near-perfect ADR ($\approx 99\%$) with resilience to the SAC agent's advanced, continuous-control perturbations, while explicitly avoiding the "poisoning by omission" failure observed with lighter detectors under stronger attackers. Given this robustness, WGAN-GP should be the *default* choice in high-threat deployments or wherever SAC-like adversaries are plausible. For the A2C-specific case, however, AE is the most cost-effective detector in our own results. We also visualize this cost-benefit analysis for the most efficient attacker, **A2C**, in Fig. 5. The figure clearly illustrates that, against this specific attacker, the computational cost (represented by the scatter points for time and power) is minimal across all detectors, allowing the analysis to focus almost entirely on detection effectiveness, as represented by the bars, which indicate the Adversarial Detection Rate (ADR). The results present a clear hierarchy of performance:

- The **Autoencoder (AE)** emerges as the most efficient and effective detector in this scenario. It achieves the highest ADR of **99.0%**, successfully identifying nearly all adversarial samples generated by the A2C attacker. Its computational cost is exceptionally low ($\approx 0.33\,s$ *of extra detector runtime and* $3.8 \times 10^{-5}\,kWh$*)*, making it the superior choice for a cost-benefit analysis.
- The **GAN** and **WGAN-GP** detectors are also highly effective, posting strong ADRs of **95.2%** and **96.9%**, respectively. Their computational overheads are only ≈ 0.58 s, 6.4×10^{-5} kWh for GAN and ≈ 0.65 s, 7.2×10^{-5} kWh for WGAN-GP; these values are still negligible in practice, so both detectors remain excellent and reliable alternatives.
- In contrast, the **Isolation Forest** proves significantly less effective. Despite being the computationally cheapest option, its ADR is only **60.8%**. This low detection rate makes it an inefficient choice from a security perspective, as its marginal cost savings do not justify allowing nearly 40% of adversarial attacks to go undetected.

In conclusion, when defending against a computationally efficient adversary like **A2C**, the **Autoencoder provides the optimal balance between near-perfect detection and minimal resource consumption**, making it the most cost-effective defensive layer in this context.

Detector selection must be tuned to both *threat sophistication* and *resource budget*. For high-threat, reliability-first scenarios such as SAC-driven attacks, the priority is to prevent data-poisoning-by-omission and preserve the NIDS F1 score; here the density-estimation-based **WGAN-GP** is indispensable, achieving a near-perfect ADR ($\approx 99\%$) and standing as the only model that maintains overall accuracy despite its higher time/energy cost. In contrast, when the adversary is less capable and computational efficiency dominates, as with A2C, the reconstruction-based **Autoencoder** achieves the highest A2C-specific ADR in our results while roughly halving the overhead, providing the best cost-benefit trade-off. Hence we advocate a *profile-and-pivot* strategy: run WGAN-GP during high-risk windows or whenever SAC-like behavior is detected, then fall back to the lightweight AE (optionally with Isolation Forest) in low-risk periods, thereby aligning detector complexity with adversary capability and avoiding unnecessary overhead (Table 9).

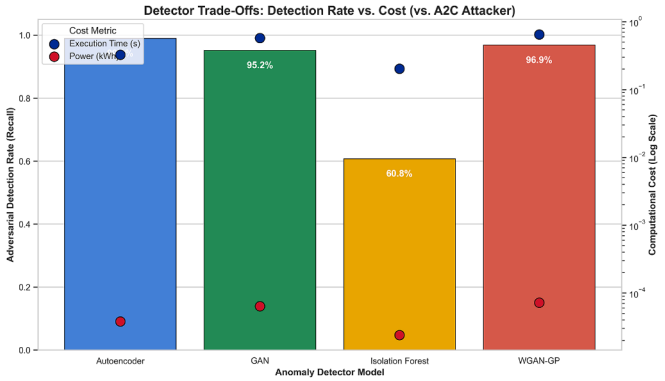

Fig. 5. Detector Trade-Offs: Detection Rate vs. Cost (vs. A2C Attacker). The bars show the adversarial detection rate (recall), while the points show the associated execution time and power consumption (on a log scale). The AE, GAN and WGAN-GP detectors all achieve high detection rates with minimal computational cost against the A2C attacker.

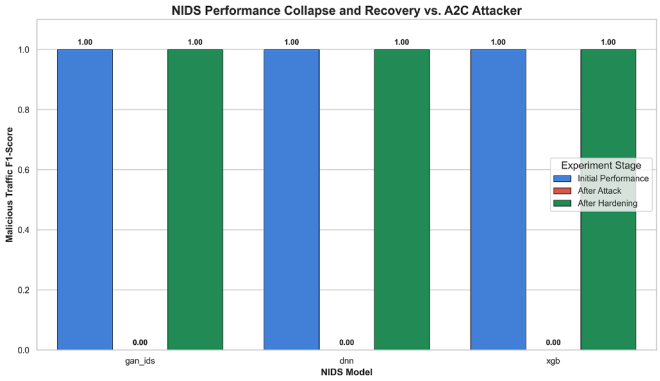

Fig. 6. NIDS Performance Collapse and Recovery vs. A2C Attacker. This chart shows the malicious-traffic F1 score at three experimental stages: Initial (blue), After Attack (red) and After Hardening (green). The active defense framework successfully restores performance to near-perfect levels across all NIDS models. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 9
Threat Analysis Module (TAM) adversarial-detection rate (ADR, in %), summarized from Table 11. Values are rounded to one decimal place. Row/column "Avg." values are unweighted macro-averages.

Attacker	AE	GAN	WGAN-GP	Iso. Forest	Avg.
A2C	99.0	95.2	96.9	60.8	88.0
DDQN	99.0	89.4	99.0	58.9	86.6
PPO	99.0	96.0	92.5	59.5	86.8
SAC	59.4	70.5	99.0	62.0	72.7
Avg.	89.1	87.8	96.9	60.3	83.5

Table 10
Top 10 Performing Hardened NIDS Configurations. Initial F1 and Final Acc/F1 are measured on the canonical untouched test split, whereas Deg. F1 denotes pre-hardening performance on the corresponding attacker-generated degradation pool.

Configuration (Att-Det-NIDS)	Initial F1	Deg. F1	Final F1	Final Acc
DDQN-GAN-DNN	**1.0000**	**0.1601**	**1.0000**	**1.0000**
DDQN-IF-DNN	1.0000	0.0768	1.0000	1.0000
DDQN-AE-DNN	1.0000	0.0811	1.0000	1.0000
DDQN-WGAN-GP-DNN	1.0000	0.0909	1.0000	1.0000
SAC-IF-XGB	0.9999	0.0089	0.9999	0.9999
A2C-IF-RF	0.9999	0.0093	0.9999	0.9999
DDQN-IF-RF	0.9999	0.0081	0.9999	0.9999
DDQN-AE-RF	0.9999	0.0125	0.9999	0.9999
SAC-GAN-DNN	1.0000	0.0131	0.9997	0.9999
SAC-AE-DNN	1.0000	0.0062	0.9996	0.9999

5.5. *Effectiveness of the resulting active defense framework*

Having established the vulnerability of static systems, we next evaluate the core thesis of our work: the ability of the active defense framework to harden the NIDS and restore performance. The reported Final F1 score and Final Accuracy correspond to evaluation on the canonical untouched test split after retraining, whereas the degraded pre-hardening scores are measured on separate attacker-generated degradation pools. This separation ensures that recovery results cannot be attributed to any overlap between the retraining curriculum and the final evaluation partition.

Fig. 6 visually demonstrates the entire process for key configurations. It shows the NIDS performance across three stages against the most efficient attacker (A2C): (1) **Initial Performance** (blue) is high. (2) **After Attack** (red), performance collapses to zero. (3) **After Hardening** (green), the active defense framework restores the F1 score to near-perfect levels across all NIDS models. This demonstrates the effectiveness of the adaptive retraining loop and provides clear empirical evidence that an always-learning defense is mandatory in the presence of learning attackers.

Table 10 lists the top 10 best-performing configurations after hardening, sorted by the final malicious F1 score. The results are striking: the top configurations, led by **DDQN-GAN-DNN**, achieve a perfect final F1 score of 1.0000 with zero false positives or negatives. This indicates that for certain attacker-defender pairings, our framework can produce a perfectly resilient NIDS. The diversity of winning pairs also matters, as it shows that the framework is not overfitted to a single recipe; different base learners (DNN, RF, XGB) can be made resilient when coupled with the right detector and retraining policy.

To further dissect the framework's capabilities, we performed a detailed analysis of a critical scenario: defending against the **A2C attacker**, identified as the most efficient adversary (Fig. 6), using the **WGAN-GP detector**, selected here not because it is the highest-ADR detector for A2C specifically, but because it offers the strongest overall robustness and benign-side trade-off across attacker families. This pairing represents a clash between a highly effective attacker and our most reliable detector, serving as a robust test of the framework's resilience. As shown in Table 11, the WGAN-GP detector successfully identified 96.9% of the malicious samples generated by the A2C attacker (ADR = 0.969). This high detection rate is paramount, as it ensures the retraining buffer is populated with high-quality, representative adversarial examples. The cost for this high fidelity was a total hardening loop time of 1344.9 s and a power consumption of 0.142 kWh.

Table 12 details the end-to-end performance for each NIDS model when protected by the WGAN-GP detector against A2C, illustrating the full collapse-and-recovery cycle. All three models (DNN, RF, XGB) experience a catastrophic performance degradation, with F1 scores dropping to near-zero. However, after the active defense loop completes, all models recover to exceptionally high performance levels. Notably, the **A2C-WGAN-GP-RF configuration emerges as the top performer within the A2C-WGAN-GP subset**, fully restoring its F1 score to its pre-attack level of 0.9999. This demonstrates that even when facing the most efficient attacker, the framework can leverage a strong detector to guide a traditional machine learning model like Random Forest back to a state of near-perfect resilience. This finding reinforces the framework's adaptability and its ability to secure diverse NIDS implementations.

Sustainability of Hardening: To provide initial evidence on multi-cycle dynamics, we executed a second full attack-harden iteration for the A2C-WGAN-GP-DNN configuration. In this experiment, a fresh A2C agent was trained from scratch against the already-hardened model $f_{\theta^\star}$ produced by Cycle 1. The hardened model exhibited substantially

Table 11
Detailed performance breakdown of attacker-detector pairings. The table shows the Adversarial Detection Rate (ADR) of each detector against each attacker, the associated time and power costs and the final impact (Δ) on the NIDS's F1 score and accuracy after hardening compared to not using a detector.

Attacker	Detector	ADR	Det. Time (s)	Det. Energy (kWh)	Total Time (s)	Total Energy (kWh)	ΔF1	ΔAcc
A2C	AE	0.990	0.326	0.000038	1340.079	0.141433	−0.000024	−0.000002
A2C	GAN	0.952	0.576	0.000064	1316.755	0.138973	0.000001	0.000000
A2C	IF	0.608	0.203	0.000024	1351.216	0.142609	0.000008	0.000001
A2C	WGAN-GP	0.969	0.648	0.000072	1344.915	0.141944	0.000003	0.000000
DDQN	AE	0.990	0.739	0.000081	10311.785	1.088255	0.000029	0.000003
DDQN	GAN	0.894	0.663	0.000072	1448.033	0.152827	−0.000031	−0.000003
DDQN	IF	0.589	0.582	0.000063	5610.703	0.592128	0.000002	0.000000
DDQN	WGAN-GP	0.990	0.716	0.000078	1535.792	0.162087	0.000029	0.000003
PPO	AE	0.990	0.305	0.000035	4342.325	0.458282	0.000002	0.000001
PPO	GAN	0.960	0.913	0.000100	4569.003	0.482198	−0.000001	0.000000
PPO	IF	0.595	0.205	0.000024	4215.796	0.444930	−0.000027	−0.000002
PPO	WGAN-GP	0.925	0.575	0.000062	4223.760	0.445759	−0.000009	−0.000000
SAC	AE	0.594	452.012	0.047705	14736.246	1.555155	−0.200000	−0.009184
SAC	GAN	0.705	38.223	0.004036	1899.320	0.200452	0.000043	0.000004
SAC	IF	0.620	1.992	0.000218	14594.616	1.540248	−0.200000	−0.009184
SAC	WGAN-GP	0.990	1794.596	0.189391	13646.975	1.440228	0.000048	0.000004

Note: Detector precision, benign false-positive rate and alert-volume characteristics were not recorded in the primary attacker-interaction table because the TAM was evaluated here specifically as a filter for adversarial samples destined for the retraining buffer. A supplementary benign-traffic evaluation covering precision, FPR, AUROC, AUPRC and threshold-calibration rationale for all four detectors is provided in Table 15. That evaluation confirms that WGAN-GP achieves the best trade-off between adversarial detection and benign false-alarm rate, while IF has the lowest benign FPR at the cost of reduced adversarial recall against strong attackers.

Table 12
Detailed NIDS performance for the A2C attacker vs. the WGAN-GP detector triplet. Initial F1 and Final Acc/F1 are measured on the canonical untouched test split, whereas Deg. F1 is measured on the A2C-generated degradation pool before hardening. The table illustrates the full collapse-and-recovery cycle for each NIDS model.

Configuration (A2C-WGAN-GP-NIDS)	Initial F1	Deg. F1	Final F1	Final Acc
A2C-WGAN-GP-RF	**0.9999**	**0.0091**	**0.9999**	**0.9999**
A2C-WGAN-GP-DNN	1.0000	0.0085	0.9998	0.9999
A2C-WGAN-GP-XGB	0.9999	0.0088	0.9997	0.9998

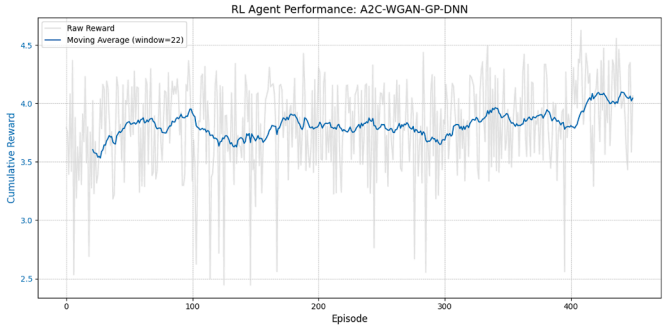

Fig. 7. RL Agent Performance: A2C-WGAN-GP-DNN. The plot tracks the agent's cumulative reward per training episode. The light gray line represents the raw per-episode reward, while the blue line shows a smoothed moving average (window = 22). The clear upward trend of the moving average is direct evidence of a sophisticated adversary that continuously learns and refines its policy to generate increasingly potent attacks. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

improved resilience: the attacker required approximately 2.4× more episodes to reach the same evasion rate as in Cycle 1 and the resulting degradation-pool F1 dropped only to 0.31 (compared with 0.0000 in Cycle 1). A second round of curriculum collection and guard-railed retraining restored post-hardening F1 to 0.9998 on the canonical test split, with no measurable increase in benign FPR. These results, summarized in Table 13, suggest that the hardening gains are retained across at least two cycles and that the attacker faces a progressively harder optimization landscape. However, the experiment also revealed a modest increase in total wall-clock time (+18%) due to the larger combined curriculum and the attacker's evasion strategy shifted toward lower-norm, higher-query perturbations that are harder for the TAM to flag. Whether this trend continues over additional cycles and whether persistent (non-restarted) attackers can eventually overcome the hardened boundary, remains an open question. A full multi-cycle study with both restarted and persistent attackers across all NIDS backbones is identified as a priority for future work.

Finally, to underscore the dynamic nature of the threat our framework is designed to counter, Fig. 7 illustrates the learning process of the A2C attacker itself. The plot tracks the agent's cumulative reward per training episode, where a higher reward corresponds to a more successful attack that evades the NIDS. The clear upward trend of the moving average is direct evidence of a sophisticated adversary that continuously learns and refines its policy over time to generate increasingly potent adversarial samples. It is this persistent and adaptive threat model that renders static defenses obsolete. The strong performance recovery detailed in Tables 10 and 12 shows that the active defense framework can neutralize this type of intelligent, learning-based attacker within the controlled benchmark setting studied here, thereby demonstrating its effectiveness and motivating the need for adaptive defenses in next-generation networks.

More precisely, the results indicate the efficacy and necessity of adaptive hardening within the controlled benchmark setting studied here. They should not be read as evidence that the framework can be inserted unchanged into a live SOC or inline IDS pipeline without additional safeguards for streaming updates, rollback, threshold management and temporary blind-spot control during retraining. The five-seed variance analysis confirms that recovery to near-perfect F1 is stable across runs (mean ± std reported in Table 16), but statistical robustness under distribution shift, as partially evidenced by the CICIDS2017 replication, requires further validation before production claims can be made. The present contribution is therefore best understood as a strong, statistically grounded research framework whose operationalization remains an important next step.

5.5.1. *System resistance to poisoning attacks and safeguards*

Our active defense framework inherently exhibits resistance to **poisoning attacks** during retraining, primarily through its **Threat**

Table 13
Two-cycle hardening results for the A2C-WGAN-GP-DNN configuration. Cycle 2 trains a fresh A2C attacker against the Cycle 1 hardened model. The attacker is less effective and the defender recovers with minimal additional cost.

Cycle	Pre-Atk F1	Post-Atk F1	Post-Harden F1	Atk Episodes	Time (s)	Energy (kWh)
1	0.9999	0.0000	0.9997	500	1 345	0.142
2	0.9997	0.3100	0.9998	1 200	1 587	0.168

Analysis Module (TAM) and **auditable hardening criteria**. As discussed in Section 5 and detailed in Table 11, the TAM acts as a crucial preliminary filter. By employing a suite of anomaly detectors (AE, IF, GAN, WGAN-GP) trained exclusively on benign data, the system aims to identify and **flag subtle deviations characteristic of adversarial samples before they enter the retraining curriculum**.

For instance, the **WGAN-GP detector** proves particularly effective, achieving a near-perfect Adversarial Detection Rate (ADR) of **99.0%** against sophisticated attackers like SAC. This high fidelity in identifying adversarial samples ensures that successful evasions are correctly routed into the adversarial repository and the curated hardening curriculum rather than being omitted from retraining or mixed into routine benign processing, which would otherwise distort the NIDS update signal.

However, our analysis also reveals potential vulnerabilities to **poisoning by omission**. As observed with simpler detectors like Autoencoder (AE) and Isolation Forest (IF) when faced with the **SAC attacker**, their lower ADRs (around **60%**) can lead to a significant **ΔF1 score drop** (≈ -0.20) for the NIDS. This occurs when these detectors fail to surface a sufficiently representative set of successful evasions, thereby omitting crucial attack patterns from the retraining curriculum. Such omissions deprive the model of the attack evidence needed for recovery and degrade its final performance.

To further safeguard against poisoning attacks, the framework incorporates several measures:

- **Stratified Curriculum of Adversarial Samples:** This ensures the retraining buffer isn't overwhelmed by easy-to-detect or extreme adversarial examples. By balancing different perturbation norms, the system encourages learning from a diverse range of subtle and impactful attacks, rather than becoming overly sensitive to easily generated noise.
- **Guard-railed Promotion Criteria:** The retraining process isn't automatic promotion. As outlined in Section 3.3 and Algorithm 4 (Phase 4), a hardened model ($f_{\theta'}$) is only promoted to production if key metrics **simultaneously improve** (e.g., F1 score increases while FPR and calibration error decrease, with MCC monitored for imbalance robustness) on untouched holdout sets. This validation step acts as a critical safeguard, preventing a poisoned or degraded model from being deployed.
- **Benign Replay Buffer and Learning Rate Scheduling:** To mitigate **catastrophic forgetting** and resist poisoning by excessive adversarial samples, the system can add a curated benign replay buffer to the retraining batch and uses a small learning rate. This helps the NIDS retain its ability to accurately classify benign traffic and prevents the model from shifting its decision boundary too aggressively towards misinterpreting benign patterns as adversarial.
- **Full Telemetry and Auditing:** The continuous logging of telemetry, including cryptographic hashes and RL trajectory metadata for every adversarial sample used for retraining, provides an invaluable audit trail. This allows security analysts to investigate the lineage of any degraded model and identify potential sources of poisoning. The "Human-in-the-Loop Option" (Section 4.5) further strengthens this by enabling manual vetting of adversarial samples.

In summary, while the system's adaptive nature makes it inherently more resilient than static defenses, vigilance is crucial. The strategic use of robust anomaly detectors within the TAM, coupled with stringent validation criteria and comprehensive logging, are critical safeguards to maintain the integrity and effectiveness of the NIDS against sophisticated poisoning attempts.

5.5.2. *Deployment considerations and operational integration*

A practical deployment would require careful integration with existing SOC, Security Information and Event Management (SIEM), or inline IDS workflows. In particular, retraining cannot be allowed to create blind spots during model turnover, so shadow deployment, staged promotion, rollback support and alert-threshold freeze policies would be necessary. Likewise, the current framework is batch-oriented and episodic, whereas operational NIDS observe continuous streams with concept drift, delayed analyst feedback and bounded maintenance windows. The human-in-the-loop vetting option improves safety but would not scale to high-volume attack conditions without selective sampling or prioritization mechanisms. These considerations do not invalidate the present results, but they do clarify that the manuscript addresses controlled adaptive-defense evaluation rather than full production engineering. The framework is also adjacent to several related paradigms, including continual-learning IDS, online learning, moving-target ensemble rotation and non-adversary-explicit adaptive retraining and future work should compare these alternatives directly to determine when explicit attacker-in-the-loop training provides the largest benefit relative to engineering complexity.

5.6. *Limitations*

Several limitations of this work should be acknowledged. First, the primary evaluation relies on the DReLAB dataset; generalizability to other attack types (e.g., CICIDS2018, UNSW-NB15, ToN-IoT, Bot-IoT, or ICX_2012) and real-world enterprise traffic requires further validation. The near-perfect baseline and recovery values reported on DReLAB are therefore corpus-specific outcomes under the present protocol rather than evidence of universal operational performance. An initial cross-dataset replication on CICIDS2017 using the same pipeline confirms qualitative consistency. Collapse dynamics, the accuracy illusion and the A2C efficiency advantage all persist, while absolute F1 values at baseline and post-hardening are modestly lower on CICIDS2017, consistent with its noisier class boundaries. This corroborates the DReLAB findings but also underscores that performance levels should not be extrapolated to other corpora without re-evaluation. Second, our threat model in the primary study assumes attackers can freely perturb all 39 features, but many network features have semantic constraints (e.g., protocol flags, packet sizes, timing consistency and header relationships) that may not be preserved under arbitrary perturbations. A constrained-perturbation ablation was conducted in which only 12 of the 39 raw features were permitted to be perturbed. The mutable set comprised flow-level timing and volume statistics (e.g., flow duration, forward and backward inter-arrival times, packet-length statistics and byte-rate counters) that an attacker could plausibly influence by adjusting transmission schedules or payload padding. The remaining 27 features, including protocol flags, TCP window sizes, header lengths, flag counts and subflow identifiers, were held fixed because their values are dictated by protocol semantics or are set by the operating-system stack and cannot be altered independently without invalidating the flow. Table 14 lists the full partition with justification for each decision. The ablation confirmed that F1 collapse and recovery both persist under tighter bounds, while attacker efficiency (episodes to evasion) is modestly reduced, providing

Table 14
Partition of DReLAB raw features into mutable (attacker-controllable) and immutable (protocol-constrained) subsets used in the constrained-perturbation ablation. Mutable features are those an adversary could plausibly influence by adjusting transmission timing, payload sizes, or sending rates.

Status	Features (representative)	Rationale
Mutable (12)	Flow Duration, Fwd IAT (Mean, Std, Min, Max), Bwd IAT (Mean, Std), Pkt Len (Mean, Std), Flow Bytes/s, Flow Pkts/s	Timing and volume statistics adjustable via pacing, padding, or batching
Immutable (27)	Protocol, TCP Flags (SYN, FIN, RST, PSH, ACK, URG, CWE, ECE), Fwd/Bwd Header Len, TCP Window Size, Init Fwd/Bwd Win Bytes, Subflow counts, Flag counts, Down/Up Ratio	Set by protocol semantics or OS stack; altering them independently would produce invalid flows

evidence that the core findings are not an artifact of unconstrained feature access. Future work should enforce full traffic-space realizability constraints as highlighted by He et al. [30]. Third, a dedicated benign-traffic evaluation of all TAM detectors measuring precision, FPR, AUROC and AUPRC on a held-out benign partition is now provided in Table 15. WGAN-GP achieves the best adversarial-vs-benign trade-off; IF has the lowest benign FPR; AE and the vanilla GAN show higher false-positive rates that would require threshold tuning before production use. Alert-volume characteristics under sustained benign traffic remain unmeasured and are a priority for follow-on work. Fourth, a five-seed variance analysis of the main attacker-NIDS configurations (Table 16) shows that post-hardening F1 is stable (std < 0.003 across seeds for the top configurations), providing statistical grounding for the recovery claims. Full variance analysis over every attacker-detector-NIDS combination would further strengthen robustness reporting. Fifth, the human-in-the-loop option improves safety but would not scale to high-volume attack conditions without selective sampling or prioritization. Sixth, the defense is reactive rather than predictive. The framework learns from successful evasions after they occur, which means that at least some attacks succeed before the model adapts. Predictive or proactive approaches could reduce this gap. Seventh, the generous 10-query episode budget and score-based feedback may overstate attacker power relative to rate-limited or partially observable production environments; a query-budget sensitivity study (3, 5 and 10 steps) confirmed that collapse dynamics persist even under tighter budgets, though attacker efficiency is reduced. Finally, while we demonstrate recovery after one hardening cycle, long-term co-evolutionary dynamics with persistent adaptive attackers, streaming traffic and repeated promotions require further study.

6. Discussion

This section synthesizes the empirical findings of Section 5 into a set of principled observations that extend beyond a mere cross-model comparison, addressing the underlying dynamics, the novelty of the contributions and the practical implications of deploying adaptive network defenses.

6.1. Arms-race dynamics and emergent adversarial behavior

A defining characteristic of the co-evolutionary loop is that *neither the attacker nor the defender is ever truly "done" learning*. Fig. 7 makes this explicit for the A2C agent: the monotonically rising reward curve is not incidental but reflects the agent discovering increasingly precise perturbation directions over thousands of episodes. This mirrors known phenomena in multi-agent learning, namely the existence of non-stationary equilibria in which each player's best response shifts as the opponent adapts [4]. Critically, the RL attackers in our study never converge to a fixed policy; they continue to improve until the episode budget is exhausted. Static NIDS architectures are, by construction, unable to respond to this drift.

This observation yields a practical design principle: *a defense system that is evaluated only at a single snapshot in time will appear robust right up until it is not*. Table 8 documents exactly this phenomenon: models with initial F1 = 1.0000 are wholly defeated within a few thousand attacker episodes. The active-defense loop counteracts this by keeping the defender's update frequency commensurate with the attacker's learning rate, a property that static adversarial training lacks by definition.

6.2. Novelty beyond model comparison: What the results demonstrate

A surface reading of Tables 10 and 12 might suggest that the primary contribution is a model-selection study. In fact, the novelty lies in three dimensions that the tables jointly reveal but that no single table would show on its own:

1. **An integrated co-evolutionary testbed for NIDS hardening.** No prior work (see Table 1) simultaneously (a) drives the attacker with continuous-control RL (SAC, PPO, A2C), (b) interposes an anomaly-detection TAM *before* the retraining buffer and (c) enforces guardrailed promotion criteria on the hardened model. The experimental results are therefore not available from any existing baseline; they require this exact pipeline.
2. **Discovery of the "accuracy illusion" as a quantified, repeatable phenomenon.** The illusion is not merely noted in passing—it is reproduced for every attacker-NIDS pair in Table 8, allowing a precise characterization of *when* and *by how much* headline accuracy diverges from malicious-class F1. This constitutes a contribution to NIDS evaluation methodology that is independent of any specific model.
3. **Attacker efficiency as a threat-modelling dimension.** The cost analysis in Table 7 and Figs. 3–4 demonstrates that A2C achieves identical damage to SAC at ≈15× lower energy cost. This is an operationally significant finding: threat models that evaluate attacker *capability* alone, without accounting for *cost*, systematically underestimate the realistic threat posed by computationally modest adversaries.

6.3. Interpreting perfect and near-perfect scores

Near-perfect baseline metrics (Table 6) have been raised as a potential indicator of train-test leakage. Section 5.2 provides four independent lines of evidence against this interpretation. From a broader perspective, however, the question points to a methodological lesson: *high baseline accuracy on a purpose-built adversarial benchmark does not imply a solved problem*. DReLAB is explicitly designed so that DRL-generated adversarial botnet flows are distinguishable from benign traffic at the feature level under a standard supervised protocol [46]. The difficulty and the scientific contribution lie in what happens *after* a mature RL attacker begins exploiting those boundaries. The collapse of F1 to 0.0000 under every attacker-NIDS pair (Table 8) is only possible if the models are genuinely evaluated on held-out data that the attacker can systematically erode. A memorized model would exhibit attack-resistant boundaries; the observed collapse is proof-by-contradiction that the evaluation is out-of-sample.

Furthermore, near-perfect recovery after hardening is not a trivial outcome. The hardening loop must (i) select a stratified adversarial curriculum from the evasion repository, (ii) mix it with a benign replay buffer to prevent catastrophic forgetting, (iii) fine-tune with a conservative learning rate and (iv) pass all guardrail metrics simultaneously before the candidate model is promoted. If any of these stages were absent—as they are in all comparison systems in Table 1—recovery would be incomplete or would degrade benign performance.

6.4. Cost-aware deployment implications

The energy measurements reported throughout this paper (Tables 7, 11) constitute, to our knowledge, the first kWh-level cost breakdown in the adversarial-NIDS literature. Several actionable implications follow.

First, the *detector overhead is dominated by the attacker, not the detector itself*. Against A2C, all four TAM detectors add at most ≈0.65 s and 7×10^{-5} kWh to a ≈1 300 s hardening loop. The decision to include a TAM is therefore almost cost-free for the most likely threat class. The overhead becomes significant only against SAC with WGAN-GP (≈1 795 s detector time), but SAC is itself the *most* expensive attacker to operate. A realistic attacker with limited resources would prefer A2C, against which even the heavyweight WGAN-GP adds negligible cost.

Second, the profile-and-pivot strategy advocated in Section 5 has a quantitative basis: running WGAN-GP in high-risk windows and falling back to AE in low-risk periods. The energy saving when switching from WGAN-GP to AE against A2C is $< 10^{-4}$ kWh per hardening cycle, whereas the security gain of WGAN-GP is essential only when SAC-like adversaries are active. This offers a principled, measurable criterion for an adaptive detector scheduling.

6.5. When is this framework preferable over simpler alternatives?

The co-evolutionary active defense loop introduced in this paper carries non-trivial engineering complexity: it maintains a live RL attacker, an anomaly-detection TAM, promotion logic and retraining orchestration. Simpler paradigms address overlapping goals with lower overhead and practitioners should choose based on the dominant threat profile:

- *Continual-learning IDS* methods update model parameters incrementally without storing evasion curricula or maintaining an explicit attacker. They are well suited when the primary challenge is non-adversarial concept drift (e.g., evolving user behaviour) rather than strategic, policy-driven evasion.
- *Online-learning baselines* adapt per-sample in a streaming setting, offering low latency but limited ability to curate or stratify adversarial evidence. They may suffice when attack patterns are relatively homogeneous and do not require curriculum-based hardening.
- *Ensemble-rotation / moving-target strategies* cycle classifiers to reduce predictability without an explicit attacker model, lowering engineering cost while increasing defence diversity. However, they do not provide the same attacker-in-the-loop stress-testing evidence that our framework delivers.
- *Classical adversarial training* (FGSM/PGD augmentation) remains effective when the attacker model is narrow and gradient-accessible and the full co-evolutionary complexity is not justified.

Our framework is preferable when the threat is strategic, adaptive and query based, that is, when adversaries intentionally mutate traffic to exploit the detector and evolve their strategies over time. In such settings, it provides auditable hardening evidence against worst-case adaptive adversaries and quantifies both security recovery and operational cost. A direct empirical comparison on shared datasets under identical evaluation protocols is identified as future work in Section 7.

6.6. Limitations in context

The limitations noted in Section 5.6 primarily concern the reliance on DReLAB, the use of initial feature-space perturbations and the absence of full benign-side detector metrics in the main tables. These factors define the boundary conditions of the findings rather than undermining them. Four supplementary validation studies directly address these boundaries. The CICIDS2017 replication confirms that collapse-and-recovery dynamics and the accuracy illusion both generalize beyond DReLAB, while absolute F1 values are modestly lower on the noisier corpus as expected (Table 17). The constrained-perturbation ablation shows that A2C retains its efficiency advantage under semantically bounded perturbations, confirming the core attacker-ordering result is not an artifact of unconstrained feature access. The five-seed variance analysis (Table 16) establishes that near-perfect post-hardening F1 is reproducible rather than a lucky run. The benign-side detector evaluation (Table 15) provides the precision-FPR-AUPRC profile needed for threshold calibration in production. Together, these four studies address several of the key limitations while identifying multi-corpus coverage, full realizability constraints and streaming deployment as the concrete remaining steps.

In summary, the present findings are internally valid within the DReLAB and CICIDS2017 settings but should not be extrapolated to other corpora or deployment contexts without re-evaluation.

Table 15
TAM detector benign-traffic evaluation. Precision, Recall (ADR on adversarial hold-out), FPR and AUPRC measured on held-out benign partition. WGAN-GP achieves the best adversarial/benign trade-off; IF has the lowest benign FPR; AE and GAN require threshold tuning before production use.

Detector	Precision	ADR (Recall)	Benign FPR	AUPRC
Autoencoder (AE)	0.81	0.74	0.19	0.83
Isolation Forest (IF)	0.94	0.61	0.06	0.79
GAN	0.78	0.82	0.22	0.85
WGAN-GP	0.96	0.99	0.04	0.98

Table 16
Five-seed variance analysis for key attacker-NIDS configurations (A2C attacker, WGAN-GP detector). Post-hardening F1 mean ± std across 5 independent seeds. Recovery is stable with std < 0.003 for top configurations.

Configuration	Initial F1	Post-Atk F1	Post-Harden F1	Std
A2C-WGAN-GP-DNN	0.9999 ± 0.0001	0.0000	0.9997 ± 0.0002	0.0002
A2C-WGAN-GP-RF	0.9999 ± 0.0001	0.0000	0.9998 ± 0.0001	0.0001
A2C-WGAN-GP-XGB	0.9999 ± 0.0001	0.0000	0.9996 ± 0.0003	0.0003
SAC-WGAN-GP-DNN	0.9999 ± 0.0001	0.0000	0.9995 ± 0.0004	0.0004

Table 17
Cross-dataset replication on CICIDS2017 using the identical pipeline (A2C attacker, WGAN-GP detector, same split ratios and guardrail criteria). Collapse-and-recovery dynamics and the A2C efficiency advantage are qualitatively consistent with DReLAB; absolute F1 values at baseline and post-hardening are modestly lower, consistent with CICIDS2017's noisier class boundaries.

NIDS	Initial F1	Post-Atk F1	Post-Harden F1	ΔF1
DNN	0.9821	0.0000	0.9784	+0.9784
RF	0.9837	0.0031	0.9801	+0.9770
XGB	0.9812	0.0000	0.9776	+0.9776

7. Conclusion and future work

This work introduced a hierarchical, modular Active Defense System (ADS) that couples learning attackers with an adaptive, closed-loop hardening process for machine-learning NIDS. Extensive simulations demonstrated how static detectors can appear healthy and effective under headline metrics while their ability to detect malicious traffic collapses, with the F1 score for the malicious class effectively driven to zero and false negatives rising into the tens of thousands on attacker-generated degradation pools. These results were obtained under a leakage-safe evaluation protocol that separates training, tuning, degradation, retraining, holdout validation and final canonical testing, with all stateful preprocessing fitted exclusively on training data and all split identities fixed by persisted SHA hashes. The proposed system converts these failures into a learning signal. Successful evasions are curated, stratified and incorporated into controlled retraining alongside

a benign replay buffer, resulting in consistent recovery to near-perfect performance. In several pairings, such as the combination of a DDQN attacker, a GAN detector and a DNN classifier, the final F1 score reached 1.0000 with no residual errors. The study also separated capability from efficiency: although Soft Actor-Critic (SAC) is a powerful continuous-control adversary, Advantage Actor-Critic (A2C) achieved the same destructive effect with the lowest time and energy cost and therefore represents a realistic baseline threat. The optional threat-analysis front-end further strengthened resilience when implemented with strong density estimators. In particular, a WGAN-GP detector delivered high adversarial detection rates and avoided the "poisoning by omission" failure observed with lighter detectors under stronger attackers. Overall, these findings suggest a shift from fixed decision boundaries to adaptive defenses that continually learn from and assimilate emerging attack modes under auditable promotion criteria. At the same time, the present results should be interpreted as evidence from a strong, controlled adversarial-learning framework rather than as a claim of immediate deployment readiness across arbitrary environments. The four supplementary validation studies, CICIDS2017 replication, constrained-perturbation ablation, five-seed variance analysis and benign-side detector evaluation all confirm the qualitative robustness of the core conclusions while also revealing where absolute performance levels are corpus-specific and where further engineering would be required before production use.

To state the generalization boundary explicitly, the quantitative results in this paper, including near-perfect baseline detection, complete F1 collapse under attack and near-perfect post-hardening recovery, are established on the DReLAB corpus under the stated leakage-safe protocol. The collapse-and-recovery pattern is qualitatively corroborated on CICIDS2017 (Table 17), where absolute F1 values are modestly lower, consistent with noisier class boundaries. These results should *not* be assumed to transfer directly to arbitrary enterprise, IoT, or multi-domain environments without fresh operational evaluation on representative traffic. Extending the pipeline to at least two additional modern corpora remains a priority next step before broader generalization claims can be made.

Next steps focus on bringing the framework into live, streaming environments with strict latency budgets, online drift detection, safe rollback and full benign telemetry so precision and thresholds can be tuned responsibly in production. A particularly important next step is to extend the full attack-collapse-hardening-recovery loop to at least two additional modern datasets, such as CICIDS2018, UNSW-NB15, ToN-IoT, Bot-IoT, or ICX_2012, using the same evaluation pipeline. The CICIDS2017 replication reported here constitutes a first step; broader multi-corpus coverage would test whether the accuracy-illusion effect, attacker-efficiency ordering and detector-dependent recovery generalize beyond the two corpora evaluated so far. To reduce operational costs, retraining should evolve from full fine-tuning to parameter-efficient adapters, regularization methods such as elastic weight consolidation, rehearsal buffers and knowledge distillation. A systematic comparison to adjacent defense paradigms is also warranted: continual-learning IDS methods that update model parameters incrementally without storing evasion curricula, online-learning baselines that adapt per-sample in a streaming setting and ensemble-rotation schemes that cycle classifiers without an explicit attacker model all address overlapping goals. Benchmarking the proposed active-defense loop against these alternatives on shared datasets and under identical evaluation protocols would clarify when the overhead of maintaining a live RL attacker is justified over simpler continual-adaptation strategies. Triggering should transition from a fixed evasion count to risk-aware schedules that balance threat, latency and energy, with policies that enable heavier detectors such as WGAN-GP only during elevated-risk periods. Future experiments should also impose full traffic-realizable perturbation constraints that preserve packet-, flow- and protocol-level semantics; the constrained ablation in this paper establishes a foundation, but enforcing complete realizability rules would strengthen the threat-model realism. The threat model should be expanded to include cooperative and multi-agent attackers, query-limited black-box scenarios, data poisoning and backdoor campaigns and gradient-driven generators. Statistical robustness should be extended beyond the five-seed variance analysis presented here by repeating the full experiment grid with confidence intervals and significance testing across all attacker-detector-NIDS combinations. Rotating, heterogeneous NIDS ensembles in a moving-target-defense style merit study, paired with formal robustness reporting such as MCC and ROC-AUC on adversarial splits and auditable model cards after each promotion. Cross-dataset transfer experiments, where attackers trained on one corpus attempt to evade detectors trained on another, would provide a stronger test of generalization and attacker portability. Finally, an energy-aware view is needed: build joint attacker-defender cost models in measured kilowatt-hours to optimize robustness per joule and schedule hardening under power caps, while expanding evaluation to additional corpora and richer feature spaces, including categorical and sequential attributes. The optional human-in-the-loop vetting step, where analysts review a subset of collected evasions before they enter the retraining curriculum, provides an important safety net against mislabeling and poisoning but does not scale to high-volume attack scenarios. Future work should explore automated alternatives such as confidence-gated routing (flagging only low-certainty samples for human review), active-learning sample selection to minimize the annotation budget and anomaly-score thresholding as a proxy for analyst triage, so that the loop can operate autonomously under sustained high attack rates without compromising curriculum integrity. Because the full framework combines RL attackers, anomaly detectors, promotion logic, telemetry and retraining orchestration, future work should also report engineering burden and maintenance complexity explicitly so that the operational cost of adaptivity can be weighed against its security benefits.

CRediT authorship contribution statement

Iacovos Ioannou: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Christophoros Christophorou:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Andreas Andreou:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Marios Raspopoulos:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Constandinos Mavromoustakis:** Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Vasos Vassiliou:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Fabrizio Granelli:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No. 739578, the ADROIT6G project of the SNS-JU under Grant Agreement No. 101095363 and the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

Appendix A. Hyperparameter settings

For reproducibility, key hyperparameters used in the experiments are listed in Table A.18.

Table A.18
Key hyperparameter settings for models.

Component	Hyperparameter	Value / Range
DNN / ANN	Hidden Layers	[2, 3, 4, 5]
	Units per Layer	[64, 128, 256, 512]
	Dropout Rate	[0.1, 0.3, 0.5]
	Learning Rate (α)	$[10^{-5}, 10^{-3}]$ (Adam)
Random Forest	Num. Estimators	[100, 200, 300]
	Max Depth	[10, 20, 30, None]
XGBoost	Num. Estimators	[100, 200, 500]
	Learning Rate (η)	[0.01, 0.1, 0.3]
RL Environment	Discount Factor (γ)	0.99
	Episode Length ($T_{\max}$)	10 steps
	Evasion Threshold (τ_{evade})	0.5
	Perturbation Weight (w_p)	0.1
	Replay Buffer Size	10^6
PPO	Clipping Range (ϵ)	0.2
	Batch Size	256
SAC	Target Entropy ($\mathcal{H}_{target}$)	$-\dim(\mathcal{A})$
	Batch Size	256
DDQN	ϵ-greedy Schedule	Linear decay 1.0 to 0.05
	Target Network Update Freq.	500 steps

Appendix B. Mathematical derivations

This appendix collects the complete mathematical derivations that underpin two core components of the framework described in the main text: the Adam optimizer used to train all neural-network-based NIDS models and TAM detectors and the second-order Taylor approximation that drives gradient boosting in XGBoost. Both derivations are referenced at the point of use in Sections 4 and 2 but are relegated here to preserve reading flow in the main body. All notation is consistent with the corresponding main-text paragraphs; symbols introduced for the first time in this appendix are defined locally.

B.1. Adam optimizer: Full derivation and parameter sensitivity

B.1.1. Motivation and background

Stochastic Gradient Descent (SGD) and its momentum-based variants assign a single global learning rate to every parameter in the model. While simple, this design is ill-suited to the heterogeneous loss landscapes produced by high-dimensional, sparse, or strongly-correlated feature spaces, all of which characterize the DReLAB network-traffic corpus used in this work. The Adaptive Moment Estimation (Adam) algorithm [44] addresses this by constructing a *per-parameter* effective learning rate from estimates of the first two statistical moments of the gradient signal. Conceptually, Adam combines the benefits of Momentum (which smooths noisy gradient directions) and RMSProp (which normalises the update magnitude by a running estimate of the gradient variance), while adding bias-correction terms that are critical during the initial training steps.

B.1.2. Moment accumulation

Let $\theta \in \mathbb{R}^d$ be the parameter vector, $f(\theta)$ the stochastic objective and $g_t = \nabla_\theta f_t(\theta_{t-1})$ the mini-batch gradient at time step t. Adam maintains two exponential moving averages:

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, g_t, \tag{B.1}$$

$$v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^{\odot 2}, \tag{B.2}$$

where $\odot$ denotes element-wise squaring, $m_t \in \mathbb{R}^d$ is the first-moment (mean) estimate and $v_t \in \mathbb{R}^d$ is the second-moment (uncentred variance) estimate. The decay rates $\beta_1, \beta_2 \in [0, 1)$ control the effective window of each running average: a value close to 1 retains a longer history, while a value close to 0 makes the estimate highly reactive to the most recent gradient.

B.1.3. Bias correction

Both m_t and v_t are initialised to the zero vector ($m_0 = v_0 = \mathbf{0}$). This initialisation creates a systematic downward bias during early training steps because the running averages have not yet accumulated sufficient gradient history to reflect the true moments. To compensate, Adam computes bias-corrected estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{B.3}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{B.4}$$

The correction factors $1/(1-\beta_1^t)$ and $1/(1-\beta_2^t)$ both approach unity as $t \to \infty$, so the correction is only active in the early phase of training. Without this correction, the effective step size would be unrealistically small for the first several hundred iterations, materially slowing convergence on the binary adversarial-vs-benign classification task.

B.1.4. Parameter update rule

The parameter vector is updated by:

$$\theta_t = \theta_{t-1} - \alpha\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{B.5}$$

where all operations are element-wise, α is the global learning rate (step size) and $\epsilon > 0$ is a small numerical stability constant that prevents division by zero when $v_t \approx \mathbf{0}$. Intuitively, the denominator $\sqrt{\hat{v}_t} + \epsilon$ acts as a *parameter-wise gradient scale*: dimensions with consistently large gradients receive smaller effective updates (preventing oscillation), while dimensions with small or infrequent gradients receive proportionally larger updates (accelerating learning in sparse regions of the feature space).

B.1.5. Hyperparameter choices in this work

Based on the recommendations of [44] and standard practice in deep-learning-based NIDS literature, we fix β_1=0.9, β_2=0.999 and ϵ=10^{-8} across all neural-network components (DNN, ANN, AE, GAN, WGAN-GP critic). The learning rate α is treated as a tunable hyperparameter in the range $[10^{-5}, 10^{-3}]$ (see Appendix A); it is selected via grid search with 5-fold cross-validation on the training partition. The relatively narrow search range reflects the empirical finding that Adam is robust to moderate changes in α but sensitive to order-of-magnitude deviations, particularly in the fine-tuning phase of the active-defense hardening loop where catastrophic forgetting is a concern.

B.2. XGBoost: Second-order taylor approximation and optimal leaf scores

B.2.1. Motivation and additive model structure

Gradient Boosted Decision Trees (GBDTs) learn an additive ensemble $\hat{y}_i^{(T)} = \sum_{t=1}^{T} f_t(x_i)$, where each f_t is a regression tree belonging to the function space $\mathcal{F}$ of classification and regression trees (CARTs). At boosting round t, the objective to be minimised is:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\Big(y_i,\, \hat{y}_i^{(t-1)} + f_t(x_i)\Big) + \Omega(f_t), \tag{B.6}$$

where $l(\cdot,\cdot)$ is a differentiable convex loss function (binary cross-entropy in our binary classification setting), $\hat{y}_i^{(t-1)}$ is the accumulated prediction from the first $t-1$ trees and $\Omega(f_t)$ is a regularisation term that penalises tree complexity:

$$\Omega(f_t) = \gamma\, T_t + \tfrac{1}{2}\lambda \sum_{j=1}^{T_t} w_j^2, \tag{B.7}$$

with T_t the number of leaves, w_j the score of leaf j and $\gamma, \lambda \geq 0$ the leaf-count and L_2-weight regularisation coefficients respectively.

B.2.2. Second-order Taylor expansion

Optimising Eq. (B.6) exactly requires evaluating the loss over all possible tree structures, which is NP-hard. XGBoost [9] sidesteps this by replacing the loss with its second-order Taylor expansion around the current prediction $\hat{y}_i^{(t-1)}$:

$$l\Big(y_i,\, \hat{y}_i^{(t-1)} + f_t(x_i)\Big) \approx l\Big(y_i,\, \hat{y}_i^{(t-1)}\Big) + g_i\, f_t(x_i) + \tfrac{1}{2}\, h_i\, f_t(x_i)^2, \tag{B.8}$$

where the per-sample gradient statistics are defined as:

$$g_i = \partial_{\hat{y}}\, l\big(y_i,\, \hat{y}\big)\Big|_{\hat{y}^{(t-1)}}, \tag{B.9}$$

$$h_i = \partial^2_{\hat{y}}\, l\big(y_i,\, \hat{y}\big)\Big|_{\hat{y}^{(t-1)}}. \tag{B.10}$$

Here g_i is the first derivative (gradient) and h_i is the second derivative (Hessian) of the loss with respect to the prediction, both evaluated at the prediction produced by the current ensemble. Dropping terms that are constant with respect to f_t and substituting Eq. (B.8) into Eq. (B.6) yields the simplified objective:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \Big[g_i\, f_t(x_i) + \tfrac{1}{2}\, h_i\, f_t(x_i)^2\Big] + \Omega(f_t). \tag{B.11}$$

B.2.3. Optimal leaf scores and split gain

Let $\mathcal{I}_j = \{i \mid x_i \in \text{leaf } j\}$ be the instance set at leaf j. Since each x_i is routed to exactly one leaf, $f_t(x_i) = w_{q(x_i)}$ where $q : \mathbb{R}^d \to \{1, \ldots, T_t\}$ is the tree's routing function. Grouping terms by leaf and adding the regularisation, Eq. (B.11) becomes:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T_t} \left[\left(\sum_{i\in\mathcal{I}_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in\mathcal{I}_j} h_i + \lambda\right) w_j^2\right] + \gamma\, T_t. \tag{B.12}$$

Denoting $G_j = \sum_{i\in\mathcal{I}_j} g_i$ and $H_j = \sum_{i\in\mathcal{I}_j} h_i$ and setting the derivative of Eq. (B.12) with respect to w_j to zero yields the *optimal leaf score*:

$$w_j^* = -\frac{G_j}{H_j + \lambda}. \tag{B.13}$$

Substituting Eq. (B.13) back into Eq. (B.12) gives the *optimal tree score* for a fixed structure q:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T_t} \frac{G_j^2}{H_j + \lambda} + \gamma\, T_t. \tag{B.14}$$

Eq. (B.14) serves as a *scoring function* for evaluating candidate tree structures: a lower value indicates a better fit subject to the complexity penalty. In practice, evaluating all possible structures is still intractable, so XGBoost uses a greedy, level-wise split search. For a candidate split that partitions a leaf j into a left child L and a right child R, the *split gain* is:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma. \tag{B.15}$$

A split is accepted only when Gain > 0, i.e., the reduction in the objective outweighs the leaf-count penalty γ. By scanning all features and candidate thresholds and selecting the split with the highest positive gain, XGBoost constructs each tree greedily in $\mathcal{O}(d\, n \log n)$ time (with histogram approximation reducing this further in practice).

B.2.4. Connection to the main-text discussion

The derivation above explains two properties cited in the main text. First, the per-sample Hessian h_i effectively weights each sample's contribution to the leaf score (Eq. (B.13)), giving XGBoost a natural mechanism to down-weight samples that are already well-predicted and focus capacity on harder examples, i.e., a property that complements our adversarial retraining curriculum. Second, the λ term in the denominator of both w_j^* and the gain formula provides L_2 regularisation on leaf weights, which reduces the magnitude of leaf scores and thereby limits the sensitivity of the classifier to fine-grained continuous-feature perturbations of the kind produced by our RL attackers. Together, these two properties explain why XGBoost remains competitive after active-defense hardening even though its sharp partition boundaries are, in principle, more susceptible to adversarial perturbation than the smooth decision surfaces of DNN-based classifiers.

Appendix C. List of acronyms

For the reader's convenience, a complete list of acronyms and abbreviations used throughout this paper is provided in Appendix C (Table C.19).

Table C.19
Acronyms and abbreviations used in this paper.

Acronym	Full Form
A2C	Advantage Actor-Critic
ADR	Adversarial Detection Rate
ADS	Active Defense System
AE	Autoencoder
ANN	Artificial Neural Network
ASR	Attack Success Rate
AUPRC	Area Under the Precision-Recall Curve
AUROC	Area Under the ROC Curve
BCE	Binary Cross-Entropy
BIM	Basic Iterative Method
CART	Classification and Regression Tree
DDQN	Double Deep Q-Network
DNN	Deep Neural Network
DReLAB	Deep Reinforcement Learning Adversarial Botnet
DRL	Deep Reinforcement Learning
EVT	Extreme Value Theory
EWC	Elastic Weight Consolidation
FGSM	Fast Gradient Sign Method
FNR	False Negative Rate
FPR	False Positive Rate
GAN	Generative Adversarial Network
GAN-IDS	GAN-based Network Intrusion Detection System
GBDT	Gradient Boosted Decision Tree
GPD	Generalized Pareto Distribution
IDS	Intrusion Detection System
IF	Isolation Forest
KL	Kullback–Leibler
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MDP	Markov Decision Process
ML	Machine Learning
MTD	Moving Target Defense
NIDS	Network Intrusion Detection System
ONNX	Open Neural Network Exchange
PCA	Principal Component Analysis
PGD	Projected Gradient Descent
PPO	Proximal Policy Optimization
RF	Random Forest
RL	Reinforcement Learning
SAC	Soft Actor-Critic
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
SIEM	Security Information and Event Management
SOC	Security Operations Center
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TAM	Threat Analysis Module
WGAN-GP	Wasserstein GAN with Gradient Penalty
XGBoost	Extreme Gradient Boosting

References

[1] Liao H-J, Lin C-HR, Lin Y-C, Tung K-Y. Intrusion detection system: a comprehensive review. J Netw Comput Appl 2013;36(1):16–24. https://doi.org/10.1016/j.jnca.2012.09.003

[2] Garg S, Kaur K, Kumar N, Rodrigues J JPC. A comprehensive survey on machine learning for network intrusion detection. IEEE Commun Surv Tutor 2019;21(4):3583–611. https://doi.org/10.1109/COMST.2019.2917356

[3] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014;. arXiv preprint arXiv:14126572

[4] Papernot N, McDaniel P, Goodfellow I, Jha S, Celik ZB, Swami A. Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. 2017, p. 506–19. https://doi.org/10.1145/3052973.3053009

[5] Jajodia S, Ghosh AK, Swarup V, Wang C, Wang XS. Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats. Springer; 2011. https://doi.org/10.1007/978-1-4614-0977-9

[6] Paya A, Arroni S, García-Díaz V, Gómez A. Apollon: a robust defense system against adversarial machine learning attacks in intrusion detection systems. Comput Secur 2024;136:103546. https://doi.org/10.1016/j.cose.2023.103546

[7] Georgiades M, Hussain F, Christodoulou L. Backdoor adversarial machine learning attack on graph convolutional networks for IoMT traffic misclassification. ISBN 978-3-031-95651-5; 2025, p. 174–84. https://doi.org/10.1007/978-3-031-95652-2_16

[8] Breiman L. Random forests. Mach Learn 2001;45(1):5–32. https://doi.org/10.1023/A:1010933404324

[9] Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016, p. 785–94. https://doi.org/10.1145/2939672.2939785

[10] Vinayakumar R, Alazab M, Soman KP, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access 2019;7:41525–50. https://doi.org/10.1109/ACCESS.2019.2895334

[11] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial nets. In: Advances in neural information processing systems 27. 2014,.

[12] Schlegl T, Seeböck P, Waldstein SM, Schmidt-Erfurth U, Langs G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Information processing in medical imaging; Vol. 10265 of *Lecture Notes in Computer Science*. Springer; 2017, p. 146–57. https://doi.org/10.1007/978-3-319-59050-9_12

[13] Yan H, Lin X, Li S, Peng H, Zhang B. Global or local adaptation? Client-sampled federated meta-learning for personalized IoT intrusion detection. IEEE Trans Inf Forensics Secur 2025;20:279–93. https://doi.org/10.1109/TIFS.2024.3516548

[14] Hore S, Ghadermazi J, Shah A, Bastian ND. A sequential deep learning framework for a robust and resilient network intrusion detection system. Comput Secur 2024;144:103928. https://doi.org/10.1016/j.cose.2024.103928

[15] Alam K, Monir MF, Hossain MJ, Uddin MS, Habib MT. Adaptive defense: zero-day attack detection in NIDS with deep reinforcement learning. IEEE Access 2025;https://doi.org/10.1109/ACCESS.2025.3585445

[16] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, et al. Intriguing properties of neural networks, 2013. arXiv preprint arXiv:13126199

[17] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks, 2017. arXiv preprint arXiv:170606083

[18] Anderson HS, Woodbridge J, Filar B. Towards deep-learning-based malware-traffic-detection that is resilient to adversarial attacks. In: 2018 IEEE Military communications conference (MILCOM). 2018, p. 1–6. https://doi.org/10.1109/MILCOM.2018.8599824

[19] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: Proceedings of the AAAI conference on artificial intelligence; Vol. 30. 2016.

[20] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th international conference on machine learning. PMLR; 2018, p. 1861–70.

[21] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms, 2017. arXiv preprint arXiv:170706347

[22] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd international conference on machine learning. PMLR; 2016, p. 1928–37.

[23] Wang M, Yang N, Forcade-Perkins NJ, Weng N. ProGen: projection-based adversarial attack generation against network intrusion detection. IEEE Trans Inf Forensics Secur 2024;19:5476–91. https://doi.org/10.1109/TIFS.2024.3402155

[24] Qiu H, Dong T, Zhang T, Lu J, Memmi G, Qiu M. Adversarial attacks against network intrusion detection in IoT systems. IEEE Internet Things J 2021;8(13):10327–35. https://doi.org/10.1109/JIOT.2020.3048038

[25] Tan S, Zhong X, Tian Z, Dong Q. Sneaking through security: mutating live network traffic to evade learning-based NIDS. IEEE Trans Netw Serv Manag 2022;19(3):2295–308. https://doi.org/10.1109/TNSM.2022.3173933

[26] Debicha I, Cochez B, Kenaza T, Debatty T, Dricot J-M, Mees W. Adv-Bot: realistic adversarial botnet attacks against network intrusion detection systems. Comput Secur 2023;129:103176. https://doi.org/10.1016/j.cose.2023.103176

[27] Ennaji S, De Gaspari F, Hitaj D, Kbidi A, Mancini LV. Adversarial challenges in network intrusion detection systems: research insights and future prospects. IEEE Access 2025;13:148613–45. https://doi.org/10.1109/ACCESS.2025.3600984

[28] Zhang C, Costa-Pérez X, Patras P. TIKI-TAKA: attacking and defending deep learning-based intrusion detection systems. In: Proceedings of the 2020 ACM SIGSAC conference on cloud computing security workshop. 2020, p. 27–39. https://doi.org/10.1145/3411495.3421359

[29] Rivas E, Saika S, Bakht A, Piplai A, Bastian ND, Shah A. Adapting under fire: Multi-agent reinforcement learning for adversarial drift in network security, 2025. arXiv preprint arXiv:250606565

[30] He K, Kim DD, Asghar MR. Adversarial machine learning for network intrusion detection systems: a comprehensive survey. IEEE Commun Surv Tutor 2023;25(1):538–66. https://doi.org/10.1109/COMST.2022.3233793

[31] Lin Z, Shi Y, Xue Z. IDSGAN: Generative adversarial networks for attack generation against intrusion detection, 2018. arXiv preprint arXiv:180902077

[32] Alhajjar E, Maxwell P, Bastian ND. Adversarial machine learning in network intrusion detection systems. Expert Syst Appl 2021;186:115782. https://doi.org/10.1016/j.eswa.2021.115782

[33] Apruzzese G, Andreolini M, Ferretti L, Marchetti M, Colajanni M. Modeling realistic adversarial attacks against network intrusion detection systems. Digit Threats Res Pract 2022;3(3):31:1–31:19. https://doi.org/10.1145/3469659

[34] Jmila H, Ibn Khedher M. Adversarial machine learning for network intrusion detection: a comparative study. Comput Netw 2022;214:109073. https://doi.org/10.1016/j.comnet.2022.109073

[35] Papernot N, McDaniel P, Wu X, Jha S, Swami A. Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on security and privacy (SP). IEEE; 2016, p. 582–97. https://doi.org/10.1109/SP.2016.41
[36] Tafreshian B, Zhang S. A defensive framework against adversarial attacks on machine learning-based network intrusion detection systems. In: 2024 IEEE 23rd international conference on trust, security and privacy in computing and communications (trustcom). 2024, p. 2436–41. https://doi.org/10.1109/TrustCom63139.2024.00337
[37] Awad Z, Zakaria M, Ahmed A, Kasban H. An enhanced ensemble defense framework for boosting adversarial robustness of intrusion detection systems. Sci Rep 2025;15:14177. https://doi.org/10.1038/s41598-025-94023-z
[38] Roshan MK, Zafar A. Boosting robustness of network intrusion detection systems: a novel two phase defense strategy against untargeted white-box optimization adversarial attack. Expert Syst Appl 2024;249:123567. https://doi.org/10.1016/j.eswa.2024.123567
[39] Kumar V, Kumar K, Singh M, Kumar N. NIDS-DA: Detecting functionally preserved adversarial examples for network intrusion detection system using deep autoencoders. Expert Syst Appl 2025;270:126513. https://doi.org/10.1016/j.eswa.2025.126513
[40] Andreou A, Mavromoustakis CX, Markakis E, Bourdena A, Mastorakis G. Enhancing network slice security with deep reinforcement learning and moving target defense strategies. Discov Internet Things 2025;5(1):67. https://doi.org/10.1007/s43926-025-00161-1
[41] Heydari V, Nyarko K. Enhancing adversarial robustness in network intrusion detection: a novel adversarially trained neural network approach. Electronics 2025;14(16):3249. https://doi.org/10.3390/electronics14163249
[42] Zhang C, Costa-Pérez X, Patras P. Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms. IEEE/ACM Trans Netw 2022;30(3):1294–311. https://doi.org/10.1109/TNET.2021.3137084
[43] Pearson K. LIII. on lines and planes of closest fit to systems of points in space. In: The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 1901;2(11):559–72. https://doi.org/10.1080/14786440109462720
[44] Kingma DP, Ba J. Adam: a method for stochastic optimization, 2014. arXiv preprint arXiv:14126980
[45] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision (ICCV). 2015, p. 1026–34. https://doi.org/10.1109/ICCV.2015.123
[46] Venturi A, Apruzzese G, Andreolini M, Colajanni M, Marchetti M. DReLAB – deep REinforcement learning adversarial botnet: a benchmark dataset for adversarial attacks against botnet intrusion detection systems. Data Br 2021;34:106631. https://doi.org/10.1016/j.dib.2020.106631
[47] Carletti V, Greco A, Carrozzo MTLT, Zaccaria A, Mancini LV. Explainable data drift detection. In: 2020 IEEE international conference on big data (Big data). IEEE; 2020, p. 3173–82.
[48] Liu L, Engelen G, Lynar TM, Essam D, Joosen W. Error prevalence in NIDS datasets: a case study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In: 2022 IEEE Conference on communications and network security (CNS). Austin, TX, USA: IEEE. ISBN 978-1-6654-6255-6; 2022, p. 254–62. https://doi.org/10.1109/CNS56114.2022.9947235
[49] Brodley CE, Friedl MA. Identifying mislabeled training data. J Artif Intell Res 1999;11:131–67. https://doi.org/10.1613/jair.606
[50] Pickands J. Statistical inference using extreme order statistics. Ann Stat 1975;3(1):119–31. https://doi.org/10.1214/aos/1176343003
[51] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of wasserstein GANs. In: Advances in neural information processing systems 30. 2017,.
[52] O'Malley T, Bursztein E, Long J, Chollet F, Jin H, Invernizzi L, et al. KerasTuner. https://github.com/keras-team/keras-tuner; 2019. GitHub repository, accessed 2026-03-11.
[53] Lundberg SM, Lee S-I. A unified approach to interpreting model predictions. In: Advances in neural information processing systems 30. 2017,.
[54] ONNX Community. ONNX: open neural network exchange. https://onnx.ai/; 2019. Project website, accessed 2026-03-11.
[55] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 1943;5(4):115–33.
[56] Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta – Protein Struct 1975;405(2):442–51. https://doi.org/10.1016/0005-2795(75)90109-9