Cross Dataset Evaluation of Robustness in Machine Learning Based Intrusion Detection Under Structured Label Poisoning

Oğuz Kağan Hitit Damla Görgülü Eren Yavuz Hakan Çapuk Rana Ataseven

Abstract

Machine learning-based Network Intrusion Detection Systems (NIDS) are increasingly deployed to protect critical infrastructure, yet their vulnerability to training-time attacks remains poorly understood. This work presents the first comprehensive cross-dataset evaluation of label poisoning attacks against ML-based NIDS, systematically studying how adversaries who corrupt training labels can blind intrusion detectors while maintaining deceptively high overall accuracy, a phenomenon we term the *accuracy illusion*. We evaluate five model architectures (CNN, RNN, MLP, Logistic Regression, Random Forest) across four benchmark datasets (CIC-IDS2017, UNSW-NB15, CUPID, CIDDS-001) under five poisoning strategies at varying intensities (5%, 10%, 20%). Our experiments reveal alarming vulnerabilities: on CIC-IDS2017, a simple class hiding attack at 20% poisoning reduced attack recall from 98% to 0% while accuracy remained above 80%. We find that dataset class imbalance critically determines vulnerability; balanced datasets resist poisoning while imbalanced datasets enable complete detection failure. Surprisingly, simple random label flipping outperforms sophisticated targeted attacks. We evaluate removal and reweighting defenses, documenting their architecture-dependent effectiveness and ultimate failure at high poisoning rates. We also report a significant baseline failure on CIDDS-001, where extreme class imbalance (367:1) renders standard models non-functional as intrusion detectors even on clean data, a cautionary finding for practitioners. Our results establish that accuracy alone cannot assess NIDS health, and we provide concrete recommendations for robust deployment.

1 Introduction

In today's world, edge devices are interconnected over sophisticated networks all around the globe. These networks move trillions of packets every day with various intentions, such as web browsing, video and audio streaming, file transfers, remote administration, financial transactions, and sensor data collection, among many others.

However, because any device connected to the World Wide Web can be addressed, anybody, especially intruders, can send malicious packets or execute harmful activities, for example jamming normal traffic by launching denial-of-service attacks. Such incidents can have serious consequences, ranging from service outages and data breaches to damage that can affect critical infrastructure and even national-level organizations.

Thus, in order to preserve the safety of both communicating parties, packets must be investigated thoroughly before being accepted as valuable information bits. For this investigation, network intrusion detection systems (NIDS) have been created. NIDS inspect network traffic and attempt to classify behaviors into benign and malicious, providing protection against potential threats. However, the success of these systems is not trivial. Building an effective NIDS requires detailed experiments, wise design choices, and a deep understanding of everyday network activities carried out by security professionals.

In practice, NIDS performance is often framed in terms of two types of errors. A false negative (where the system fails to detect an ongoing attack) can allow intruders to compromise systems, steal data, or disrupt services with little or no response. A false positive (where the system flags legitimate traffic as malicious) can overwhelm analysts with alerts, leading to alert fatigue and potentially causing real attacks to be missed in the noise. Balancing these errors is a central challenge in NIDS design and deployment.

With the increased availability of large datasets and advances in machine learning techniques, efforts on NIDS have intensified. Researchers commonly evaluate their approaches on benchmark datasets such as KDD Cup 99 [1], NSL-KDD [2], UNSW-NB15 [3], and CIC-IDS2017 [4], which provide labeled examples of normal and malicious traffic. Machine learning models, including logistic regression, random forests, and neural networks, are trained on these labeled datasets to distinguish benign from malicious traffic. However, this reliance on labeled training data introduces a critical vulnerability: if an adversary can corrupt the labels during data collection or annotation, the resulting model may learn to misclassify traffic in ways that benefit the attacker.

This threat, known as *label poisoning* or *label flipping*, is a training-time attack where an adversary manipulates a fraction of the training labels without altering the underlying features. Unlike adversarial examples that perturb inputs at inference time, label poisoning corrupts the model before deployment. The attack can be *untargeted* (randomly flipping labels to degrade overall accuracy) or *targeted*, where specific attack classes are relabeled as benign to create blind spots in the detector. Targeted label poisoning is particularly dangerous because it allows adversaries to selectively suppress detection of high-value intrusions while maintaining acceptable overall accuracy, making the attack difficult to notice.

Despite the operational relevance of such structured label-poisoning threats, existing evaluations are often limited to single datasets or narrow model classes, leaving cross-dataset vulnerability poorly characterized. Prior work has demonstrated label poisoning in image classification and spam filtering, but systematic studies on NIDS across diverse network traffic distributions remain scarce.

In this work, we address this gap through a comprehensive cross-dataset study of targeted label-poisoning attacks and practical defenses for NIDS. We evaluate four complementary benchmark datasets (UNSW-NB15, CIC-IDS2017, CUPID, and CIDDS-001), selected to represent diverse network environments, attack distributions, and class imbalance characteristics. We employ a representative model suite spanning traditional machine learning (Logistic Regression, Random Forest) and neural architectures (MLP, CNN, and RNN) to understand how model complexity affects vulnerability. Under bounded label-flip budgets (5%, 10%, 20%), we implement five structured poisoning strategies: class hiding (random label flipping), feature-targeted poisoning, influence-aware poisoning, disagreement-based poisoning, and temporal window poisoning (applied exclusively to CUPID due to its temporal structure). We further evaluate removal and reweighting defenses to characterize their effectiveness across attack intensities.

Our experiments reveal alarming vulnerabilities. On CIC-IDS2017, a class hiding attack with only 10% label poisoning reduced attack recall from 98% to 16%, and at 20% poisoning, the model achieved 0% attack detection, a complete failure to identify any malicious traffic. Critically, overall accuracy remained above 80% in these scenarios, creating an *accuracy illusion* where the model appears functional but is effectively blind to attacks. Defense mechanisms showed limited effectiveness: removal defense provided some protection at low poisoning rates but collapsed at higher rates, while reweighting defenses were largely ineffective across all scenarios. We also observed significant dataset-specific vulnerability patterns, with CIC-IDS2017 being most susceptible and UNSW-NB15 showing greater resilience.

The contributions of this work are:

- **Systematic cross-dataset evaluation**: We provide the first comprehensive study of label poisoning across four diverse NIDS benchmark datasets with consistent experimental methodology.
- **Attack strategy comparison**: We implement and compare five structured poisoning strategies, identifying class hiding as the most effective attack and characterizing the conditions under which each strategy succeeds.
- **Defense analysis**: We evaluate practical training-time defenses (removal and reweighting) and document their failure modes at higher poisoning rates.
- **The accuracy illusion**: We identify and characterize the dangerous phenomenon where poisoned models maintain high overall accuracy while completely failing to detect attacks, a critical finding for practitioners who rely on accuracy as a health metric.

2 Related Work

This section reviews the literature that motivated and informed our study, organized into four thematic areas: (1) the evolution of network intrusion detection systems and benchmark datasets, (2) foundational research on data poisoning attacks, (3) label-flipping as a specific and effective attack vector, and (4) defense mechanisms proposed to mitigate such threats. By tracing the development of these research threads, we identify the gaps that our cross-dataset evaluation aims to address.

2.1 Network Intrusion Detection Systems and Benchmark Datasets

The need for automated network security monitoring emerged alongside the rapid expansion of the Internet in the 1990s. One of the earliest and most influential systems was **Bro** (now known as Zeek), developed

by Paxson in 1999 [5]. Bro introduced a novel architecture that separated the low-level event engine from a high-level policy script interpreter, enabling flexible, protocol-aware analysis of network traffic in real time. This design philosophy, combining deep packet inspection with scriptable detection logic, established a template that many subsequent systems would follow.

As machine learning techniques matured, researchers began applying them to intrusion detection, hoping to automate the discovery of attack patterns and reduce reliance on hand-crafted signatures. However, Sommer and Paxson (2010) [6] offered a critical assessment of this trend in their influential paper "Outside the Closed World." They argued that many ML-based NIDS evaluations suffered from unrealistic assumptions: closed-world datasets with artificial attack-to-benign ratios, lack of concept drift, and evaluation metrics that failed to capture operational costs. Their work called for more rigorous, realistic evaluation practices, a challenge that remains relevant today and directly motivates our cross-dataset methodology.

The quality and representativeness of benchmark datasets has been a persistent concern. The **KDD Cup 1999** dataset [1], derived from the 1998 DARPA intrusion detection evaluation, became the de facto standard for over a decade despite well-documented limitations including redundant records, unrealistic traffic distributions, and outdated attack patterns. Tavallaee et al. (2009) [2] addressed some of these issues by creating the **NSL-KDD** dataset, which removed duplicate records and rebalanced the data, though fundamental concerns about age and realism remained.

More recent efforts have produced datasets that better reflect contemporary network environments, including UNSW-NB15 [3], CIC-IDS2017 [4], CUPID [7], and CIDDS-001 [8]. Recent survey work by Khraisat et al. (2019) [9] and Chou and Jiang (2021) [10] has synthesized this landscape, cataloging available datasets, comparing detection techniques, and identifying open challenges. Goldschmidt and Chudá (2025) [11] further systematized dataset limitations, providing recommendations that guided our selection of complementary benchmarks. A key insight from this literature is that conclusions drawn from a single dataset rarely generalize. This motivates our decision to evaluate across multiple diverse benchmarks with different traffic characteristics, attack distributions, and class imbalance profiles.

2.2 Data Poisoning Attacks

While considerable effort has focused on building accurate intrusion detection models, less attention has been paid to their robustness against adversarial manipulation of training data. **Data poisoning** refers to attacks where an adversary corrupts the training set to degrade model performance or induce targeted misclassifications at inference time.

The seminal work by **Biggio, Nelson, and Laskov (2012)** [12] established the theoretical and algorithmic foundations for poisoning attacks against Support Vector Machines. They demonstrated that an adversary could craft malicious training points using gradient ascent on the validation error surface, significantly increasing the classifier's test error. Crucially, their attack assumed only that the adversary could inject data into the training pipeline, a realistic threat model for systems that collect training data from untrusted sources such as honeypots or user-reported samples. This work showed that the standard machine learning assumption of well-behaved, i.i.d. training data does not hold in security-sensitive settings.

Building on this foundation, **Steinhardt, Koh, and Liang (2017)** [13] developed a theoretical framework for certified defenses against data poisoning. Their key contribution was constructing upper bounds on the worst-case loss a defender can suffer against any poisoning attack within a specified budget. They showed that defense effectiveness is highly dataset-dependent: while MNIST and Dogfish image datasets proved resilient (achieving at most 4% error even with 30% poisoned data under an oracle defense), the IMDB sentiment corpus was driven from 12% to 23% error with only 3% poisoned data. This finding suggests that high-dimensional datasets with many irrelevant features are more vulnerable. It has important implications for network intrusion detection, where feature spaces are often large and heterogeneous.

Several comprehensive surveys have synthesized the growing literature on poisoning attacks. **Wang et al. (2022)** [14] provide an extensive taxonomy covering both traditional machine learning and deep learning systems. They categorize attacks by adversarial goals (untargeted vs. targeted), knowledge assumptions (white-box, gray-box, black-box), and attack mechanisms (label poisoning, feature poisoning, model poisoning). Their presentation of the bilevel optimization framework, where the outer optimization crafts poisoned samples to maximize validation loss while the inner optimization trains the victim model, provides a unified lens

for understanding diverse attack strategies. More recently, **Zhao et al. (2025)** [15] extended this analysis specifically to deep learning, discussing emerging threats to large language models and identifying open challenges in defense design.

2.3 Label-Flipping Attacks

Among poisoning strategies, **label-flipping** (also called label poisoning) stands out for its simplicity and effectiveness. In this attack, the adversary corrupts only the labels of training samples while leaving the features unchanged. Despite its simplicity, label-flipping can cause substantial damage, particularly when applied in a targeted manner by flipping labels of a specific class to create blind spots in the classifier.

Jebreel et al. (2022) [16] studied label-flipping in the context of federated learning, where malicious participants can poison their local data before contributing model updates. They demonstrated that label-flipping is both easy to perform and difficult to detect, significantly degrading accuracy on the source class (the class whose labels are flipped). Their analysis revealed that the attack's impact scales with both the proportion of malicious participants and the number of flipped samples. While their proposed defense targets federated settings specifically, their characterization of label-flipping's effectiveness applies broadly.

Chang, Dobbie, and Wicker (2023) [17] developed efficient algorithms for label-flipping attacks on tabular data, demonstrating that such attacks remain potent even when the adversary operates under tight computational or query budgets. Their work is particularly relevant to NIDS, which typically process tabular flow-level features.

A critical insight from this literature is that label-flipping attacks can be **targeted** by selectively flipping labels of attack traffic to benign, thereby suppressing detection of specific intrusion types while maintaining acceptable overall accuracy. This "accuracy illusion" makes the attack particularly dangerous in operational settings where defenders may monitor aggregate metrics without examining per-class performance.

2.4 Defense Mechanisms

Defending against data poisoning requires identifying and mitigating the influence of corrupted samples before or during training. The literature has explored several approaches:

Outlier removal is perhaps the most intuitive defense: identify samples that deviate significantly from the expected data distribution and exclude them from training. Steinhardt et al. (2017) [13] analyzed defenses that remove outliers outside a feasible set defined by class centroids. While effective against some attacks, they showed that defenses relying on empirical (potentially poisoned) centroids can be subverted when the attacker controls a larger fraction of the data.

Loss-based filtering identifies suspicious samples by their training loss: samples that consistently incur high loss may be mislabeled or adversarially crafted. This approach is computationally inexpensive and can be applied to any differentiable model. However, it assumes that poisoned samples behave differently from clean samples in terms of loss, which is an assumption that sophisticated attacks may violate.

Reweighting offers a softer alternative to removal: rather than discarding suspicious samples entirely, reduce their contribution to the training objective. This preserves potentially useful information while limiting the influence of outliers. Koh and Liang (2017) [18] developed influence functions that estimate each training sample's effect on test predictions, providing a principled basis for identifying and down-weighting harmful samples.

Ensemble disagreement exploits the observation that poisoned samples often induce inconsistent predictions across independently trained models. By training multiple models with different initializations or architectures and flagging samples where predictions disagree, defenders can identify ambiguous or adversarially manipulated regions of the feature space. This strategy draws on the query-by-committee framework introduced by Seung, Opper, and Sompolinsky (1992) [19].

2.5 Gaps in Prior Work and Our Contributions

Despite the substantial body of work reviewed above, several gaps remain:

1. **Limited cross-dataset evaluation.** Most poisoning studies evaluate on a single dataset, making it

difficult to assess whether findings generalize across different traffic distributions, feature spaces, and class imbalance profiles. Sommer and Paxson's critique of closed-world evaluation applies equally to adversarial robustness studies.

2. **Narrow model coverage.** Many studies focus on a single model family (e.g., SVMs, neural networks) without systematically comparing how poisoning affects models with different inductive biases, from linear classifiers to ensemble methods to deep networks.

3. **Disconnect between poisoning research and NIDS.** While poisoning attacks have been extensively studied in image classification and spam filtering, systematic evaluations on network intrusion detection benchmarks remain scarce. The unique characteristics of NIDS data, including high dimensionality, severe class imbalance, temporal structure, and heterogeneous attack types, warrant dedicated investigation.

4. **Defense effectiveness under varying attack intensities.** Prior work often evaluates defenses at fixed poisoning rates without characterizing how effectiveness degrades as attack intensity increases.

Our work addresses these gaps through a **systematic cross-dataset study** of label-poisoning attacks on ML-based NIDS. By evaluating four benchmark datasets (UNSW-NB15, CIC-IDS2017, CUPID, CIDDS-001), five model architectures (Logistic Regression, Random Forest, MLP, CNN, RNN), five poisoning strategies (class hiding, feature-targeted, influence-aware, disagreement-based, temporal window), and two defense mechanisms (removal, reweighting) across multiple poisoning rates (5%, 10%, 20%), we provide comprehensive empirical evidence on the vulnerability of NIDS to training-time attacks and the conditions under which defenses succeed or fail.

3 Experimental Setup

3.1 Datasets

In this study, we employ four widely recognized benchmark datasets for network intrusion detection systems (NIDS): UNSW-NB15, CIC-IDS2017, CUPID, and CIDDS-001, selected to enable a comprehensive evaluation across diverse network environments, attack types, and temporal characteristics. These datasets collectively represent a broad range of network architectures and traffic patterns, with UNSW-NB15 providing modern synthetic traffic with realistic background noise, CIC-IDS2017 capturing a wide spectrum of attack scenarios in a controlled experimental setting, CUPID offering real-world IoT network traffic with explicit temporal dependencies, and CIDDS-001 reflecting enterprise-scale network behavior. Together, they cover complementary aspects of the intrusion detection problem: UNSW-NB15 and CIC-IDS2017 supply large-scale, feature-rich data suitable for assessing model capacity and generalization, whereas CUPID enables the evaluation of temporally informed poisoning strategies through its timestamped flows. CIDDS-001, which is natively multi-class and subsequently converted to a binary formulation in this work, provides a rigorous test of preprocessing robustness. All four datasets are well-established benchmarks in the NIDS literature, supporting the reproducibility of experimental results.

Moreover, the datasets natively encompass a diverse range of attack categories, including Denial of Service (DoS), Distributed Denial of Service (DDoS), port scanning, brute-force attacks, web-based exploits, and botnet activity. While these fine-grained labels reflect the heterogeneity of real-world adversarial behaviors, in this work all datasets are systematically processed into a unified binary classification setting (benign versus malicious), as detailed in the subsequent sections, to enable consistent evaluation across datasets.

3.1.1 General Preprocessing Pipeline

A consistent preprocessing pipeline was applied uniformly across all datasets to ensure comparability and maintain experimental rigor. For datasets with predefined training and testing partitions, namely UNSW-NB15 and CIC-IDS2017, the original splits were preserved to remain consistent with prior work and established evaluation protocols. In contrast, for datasets without predefined partitions (CUPID and CIDDS-001), a stratified 70–30 train–test split was employed, ensuring balanced class representation across both subsets. Stratified sampling was further used to preserve the proportional distribution of benign and malicious

samples during partitioning, thereby preventing class imbalance from confounding performance comparisons and ensuring that evaluation metrics reflect genuine detection capability rather than dataset artifacts. All numerical features were standardized using a `StandardScaler` fitted exclusively on the training data, with the learned parameters subsequently applied to the test set to avoid data leakage. For datasets containing categorical attributes, specifically protocol, service, and state fields in UNSW-NB15 and protocol-related fields in CIDDS-001, categorical variables were converted into numerical form using a `LabelEncoder`. Missing values were addressed during the CSV loading stage by identifying and removing invalid timestamps in the CUPID dataset to preserve temporal consistency, and by imputing missing numerical feature values using forward-fill strategies where appropriate.

3.1.2 Dataset-Specific Preprocessing

All datasets were converted into a binary classification setting to ensure consistency across experiments and facilitate uniform evaluation. In this formulation, benign traffic was encoded as class 0, while all malicious traffic, irrespective of the specific attack category, was encoded as class 1. This binary labeling scheme was preferred as it aligns with the core objective of network intrusion detection systems, namely distinguishing normal network behavior from malicious activity, while still leveraging the underlying diversity of attack patterns present in the original datasets. Concretely, for a sample with original label ℓ, the binarized label $y \in \{0, 1\}$ is defined by

$$y \;=\; \mathbb{1}[\ell \neq \texttt{benign}], \tag{1}$$

where $\mathbb{1}[\cdot]$ denotes the indicator function, and the interpretation of "benign" depends on each dataset's native labeling scheme.

For CIC-IDS2017, which originally provides multi-class labels with 0 denoting benign traffic and values 1–14 corresponding to different attack types, all non-benign labels were collapsed into a single malicious class via

$$y \;=\; \mathbb{1}[\ell \neq 0]. \tag{2}$$

Similarly, CIDDS-001, which initially encodes attack information as string-valued categories in an `attack_type` field (e.g., `"benign"`, `"dos"`, `"port_scan"`), required additional processing. In this case, the original attack-type labels were first preserved in a separate `original_label` field to retain compatibility with potential future multi-class analyses. A new binary label column was then derived by mapping all non-benign entries to the malicious class, after which the original categorical attack-type field was removed from the feature set to prevent label leakage during training.

In addition to label processing, dataset-specific metadata and identifier columns were carefully removed to reduce dimensionality and avoid unintended information leakage. For UNSW-NB15, this included the removal of row identifiers and explicit attack category fields that could trivially reveal class membership. In CIC-IDS2017, the target label column was excluded after binary conversion, while all flow-based statistical features were retained. For the CUPID dataset, network identifiers such as source and destination IP addresses and ports were removed to prevent models from overfitting to environment-specific endpoints; timestamp information was also excluded from the feature set except in experiments involving temporal poisoning. Finally, in CIDDS-001, auxiliary identifiers such as attack instance IDs and intermediate label columns introduced during preprocessing were removed prior to model training. Collectively, these dataset-specific preprocessing steps ensured a fair, leakage-free, and methodologically consistent experimental foundation across all evaluated datasets.

3.1.3 Dataset Statistics

After applying the unified preprocessing pipeline described above, including train–test partitioning, feature standardization, label binarization, and dataset-specific column filtering, the dataset characteristics are summarized in Table 1. Exact statistics may vary slightly depending on timestamp validation and filtering.

3.2 Models

Five machine learning architectures spanning both traditional statistical methods and modern deep learning approaches are evaluated within the scope of this project in order to assess poisoning attack effectiveness and

Table 1: Approximate dataset statistics after preprocessing.

Dataset	Total Samples	Train Samples	Test Samples	Feature Dim.	Benign:Attack
UNSW-NB15	∼257,673	∼175,341	∼82,332	42 (enc.)	∼1:1
CIC-IDS2017	∼2,830,743	∼1,981,520	∼849,223	78	∼4:1
CUPID	∼500,000	∼350,000	∼150,000	variable	∼2:1
CIDDS-001	∼1,000,000	∼700,000	∼300,000	variable	∼5:1

defense robustness across models with differing capacities, inductive biases, and computational characteristics. Logistic Regression (LR) and Random Forest (RF) are included as interpretable statistical baselines with well-understood theoretical properties, enabling analysis of whether poisoning attacks primarily exploit architectural limitations or reflect more fundamental vulnerabilities independent of model complexity. In parallel, three neural network architectures are considered to capture increasingly expressive, non-linear decision boundaries: a Multi-Layer Perceptron (MLP) as a canonical fully connected model, a Convolutional Neural Network (CNN) that leverages local feature correlations, and a Recurrent Neural Network (RNN) that imposes a sequential structure on feature representations. Collectively, these models embody diverse inductive biases regarding the organization of network traffic features, allowing systematic investigation of whether specific poisoning strategies disproportionately affect models that assume spatial locality, sequential dependence, or minimal structural prior knowledge.

3.2.1 Logistic Regression (LR)

Logistic Regression is included as a fundamental statistical baseline for binary classification due to its interpretability and well-understood theoretical properties. As a linear model, it enables assessment of whether poisoning attacks primarily exploit architectural simplicity or introduce vulnerabilities that persist even in models with minimal representational capacity. The model computes class probabilities using a linear decision function followed by a sigmoid activation, given by

$$P(y = 1 \mid x) \;=\; \sigma(w^\top x + b) \;=\; \frac{1}{1 + \exp(-(w^\top x + b))}, \tag{3}$$

where w denotes the weight vector and b the bias term. In our experiments, Logistic Regression was trained using the L-BFGS optimization algorithm with an ℓ_2 regularization penalty of strength $C = 1.0$ to prevent overfitting. The maximum number of optimization iterations was set to 1000 to ensure convergence across datasets, and a fixed random seed of 42 was used for reproducibility. Although the task is binary, a one-vs-rest formulation was retained for consistency with the underlying library implementation. Owing to its linear decision boundary and limited capacity, Logistic Regression provides a useful reference point for evaluating the susceptibility of simple classifiers to label manipulation and poisoning-induced boundary shifts.

3.2.2 Random Forest (RF)

Random Forest is selected as a non-linear, ensemble-based statistical model that offers increased robustness to noise and outliers compared to linear classifiers, while retaining a degree of interpretability. Its inclusion allows examination of whether ensemble aggregation mitigates the impact of poisoning attacks or whether systematic label corruption can propagate across multiple base learners. The model consists of an ensemble of 100 decision trees trained using bootstrap aggregation, with each tree constructed on a randomly sampled subset of the training data and a randomly selected subset of features at each split. Gini impurity was used as the splitting criterion, and feature subsampling followed the standard square-root heuristic to encourage diversity among trees. Trees were grown with a minimum split size of two samples and a minimum leaf size of one, allowing fine-grained partitioning of the feature space. Bootstrap sampling was enabled, and all trees were trained in parallel using all available CPU cores. A fixed random seed of 42 was used to ensure reproducibility. While Random Forest models are generally resilient to isolated noisy labels, coordinated poisoning strategies may still influence multiple trees simultaneously, making this model an important intermediary between simple linear classifiers and fully learned neural representations.

3.2.3 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron is employed as a representative fully connected neural network architecture, offering substantially higher expressive power than traditional statistical models while making minimal assumptions about feature structure. This model serves as a general-purpose non-linear classifier and provides a foundation for several poisoning strategies that rely on confidence estimation or loss-based sample selection. The network consists of an input layer matching the feature dimensionality, followed by a single hidden layer with 128 neurons and ReLU activation, after which dropout with probability 0.3 is applied to mitigate overfitting. The output layer comprises two neurons with softmax activation, producing class probability estimates for the binary classification task. Training was performed using the Adam optimizer with a fixed learning rate of 0.001 and a batch size of 128 samples. All MLP models were trained for three epochs using cross-entropy loss, a deliberately limited training regime chosen to reduce memorization of poisoned labels while still capturing general feature patterns. The single-hidden-layer design provides a balance between expressive capacity and training stability, making the MLP particularly suitable for studying how poisoning attacks affect decision boundaries in moderately complex neural models.

3.2.4 Convolutional Neural Network (1D-CNN)

The one-dimensional Convolutional Neural Network (1D-CNN) is included to investigate the effect of spatial inductive biases on poisoning robustness, specifically the assumption that adjacent features in network traffic representations exhibit local correlations. Although network flow features are not spatial in the traditional sense, many features (e.g., packet size statistics, inter-arrival times, and flag counts) are semantically related and often adjacent in the feature vector, making convolutional processing a plausible modeling choice. The architecture reshapes the flat input feature vector into a single-channel one-dimensional signal and applies two successive convolutional blocks. The first convolutional layer employs 64 filters with kernel size three and unit padding, followed by a ReLU activation and max pooling with kernel size two. This is followed by a second convolutional layer with 32 filters using the same kernel configuration, again followed by ReLU activation and max pooling. The resulting feature maps are flattened and passed through a dropout layer with probability 0.3 before a final fully connected layer with softmax activation produces class probabilities. Training is performed using the Adam optimizer with a fixed learning rate of 0.001, a batch size of 128, and cross-entropy loss over three training epochs. By progressively aggregating local feature patterns while reducing dimensionality through pooling, the CNN enables evaluation of whether poisoning strategies that distort localized feature profiles disproportionately affect models with convolutional inductive biases.

3.2.5 Recurrent Neural Network (RNN)

The Recurrent Neural Network (RNN) is incorporated to assess poisoning effects under a sequential inductive bias, in which the input feature vector is interpreted as an ordered sequence rather than a flat representation. While network traffic features are not inherently temporal in their indexed form, imposing a sequential structure allows the model to capture potential long-range dependencies and interactions among feature subsets that may not be easily represented by feedforward or convolutional architectures. The model reshapes the input feature vector into a sequence of fixed-length feature segments, where the sequence length and feature dimensionality are chosen to evenly partition the original input dimension. A single-layer vanilla RNN with a hidden state size of 64 processes this sequence and produces a sequence of hidden states, from which the final hidden state corresponding to the last time step is extracted as a summary representation. Dropout with probability 0.3 is applied to this representation prior to a fully connected output layer with softmax activation for binary classification. The RNN is trained using the Adam optimizer with a learning rate of 0.001, a batch size of 128, and cross-entropy loss over three epochs. This architecture provides an alternative inductive bias compared to CNNs and MLPs, enabling analysis of whether poisoning strategies exploit or interact differently with models that enforce sequential dependency structures.

3.3 Poisoning Strategies

Data poisoning attacks aim to degrade a model's generalization performance by deliberately corrupting the training data, and in this work we implement five distinct poisoning strategies that target different aspects of the learning process. All strategies are evaluated under three poisoning rates (5%, 10%, and 20% of the training set), resulting in three poisoned variants per strategy and dataset, in addition to a clean baseline.

Each strategy enforces a global poisoning budget, whereby exactly a fixed proportion of training samples is selected for label manipulation regardless of the underlying selection mechanism. Denoting the poisoning rate by $p \in (0, 1)$ and the number of training samples by N, the poisoning budget is defined as

$$B \;=\; \lfloor pN \rfloor \,. \tag{4}$$

When the candidate set produced by a strategy is smaller than the required budget, the remaining samples are filled via uniform random selection from non-candidates.

Poisoning is applied through bidirectional label flipping, such that benign samples are relabeled as malicious and malicious samples are relabeled as benign. For a selected training sample with original binary label $y \in \{0, 1\}$, the poisoned label $\tilde{y}$ is given by

$$\tilde{y} \;=\; 1 - y. \tag{5}$$

This operation simultaneously introduces false positives and obscures true attack patterns, thereby blurring the decision boundary in both directions. For each dataset, this procedure yields a comprehensive evaluation set consisting of the clean dataset and multiple poisoned variants generated using class hiding, feature-targeted, influence-aware, and disagreement-based strategies, with an additional temporal window–based strategy applied exclusively to the CUPID dataset due to its timestamp information. In total, this design produces between 16 and 19 training datasets per original dataset, enabling systematic assessment of poisoning effectiveness across strategies, poisoning intensities, and model architectures.

3.3.1 Class Hiding

Class hiding is included as a baseline poisoning strategy due to its simplicity and its ability to introduce label noise without relying on domain knowledge or model-specific information. The strategy operates by randomly selecting B training samples (Eq. 4) uniformly from the training set and applying bidirectional label flipping (Eq. 5). Sample selection is performed uniformly at random across the training set using a fixed random seed to ensure reproducibility. Because all samples have equal probability of being poisoned, the strategy is class-agnostic and does not preferentially target either benign or attack instances. The resulting poisoned dataset simultaneously introduces false negatives by concealing attack samples and false positives by corrupting benign traffic. Although class hiding does not exploit structural weaknesses of specific models, it serves as an important reference point for assessing model sensitivity to unstructured label noise and for distinguishing the effects of random corruption from more targeted poisoning strategies.

3.3.2 Feature-Targeted Poisoning

Feature-targeted poisoning leverages domain knowledge about network traffic characteristics to identify training samples that are likely to be informative for intrusion detection models and therefore high-impact poisoning targets. Instead of selecting samples uniformly at random, this strategy defines a set of dataset-specific feature predicates designed to capture common attack behaviors or anomalous traffic patterns, such as unusually short connection durations, elevated packet rates, protocol-specific flag patterns, or large payload sizes. For a given dataset, let $\mathcal{P} = \{\pi_1, \ldots, \pi_m\}$ denote the set of predicates, where each predicate $\pi_j(\cdot)$ maps a feature vector to a boolean value. The candidate set is formed as the union of all predicate matches:

$$\mathcal{C} \;=\; \bigcup_{j=1}^{m} \{\, i \in \{1, \ldots, N\} \;:\; \pi_j(x_i) = \texttt{true} \,\}. \tag{6}$$

From $\mathcal{C}$, samples are selected up to the global poisoning budget B; if $|\mathcal{C}| \geq B$, a subset of size B is sampled uniformly from $\mathcal{C}$, whereas if $|\mathcal{C}| < B$, all candidates are included and the remaining $B - |\mathcal{C}|$ samples are drawn uniformly from $\{1, \ldots, N\} \setminus \mathcal{C}$. Bidirectional label flipping is then applied to the selected instances (Eq. 5). By focusing on samples exhibiting features that are both semantically meaningful and highly discriminative for attack detection, this strategy aims to distort the association between characteristic attack patterns and malicious labels. As a result, models trained on the poisoned data may learn incorrect correlations, leading to systematic misclassification of future traffic that exhibits similar feature profiles and effectively creating blind spots in the learned detection logic. The dataset-specific feature predicates used in this study are summarized in Table 2.

Table 2: Dataset-specific feature predicates used for feature-targeted poisoning.

Dataset	Predicate Name	Features Used	Rationale
UNSW-NB15	tcp_short	proto, dur	TCP scanning attacks often exhibit short connection durations
UNSW-NB15	http_service	service	HTTP is a common attack surface
UNSW-NB15	high_rate	rate	DoS/DDoS attacks typically generate elevated packet rates
CIC-IDS2017	short_high_packets	Flow Duration, Total Fwd Packets	Scanning and flooding behavior
CIC-IDS2017	syn_pattern	SYN Flag Count	Indicative of SYN flood attacks
CIC-IDS2017	large_packets	Average Packet Size	Large payload or exfiltration-like behavior
CUPID	short_high_volume	Flow Duration, Total Fwd Packets	IoT-specific burst and attack patterns
CUPID	syn_pattern	SYN Flag Count	DDoS indicator
CIDDS-001	tcp_short	proto, duration	Fast connection scanning
CIDDS-001	high_packets	packets	Traffic volume anomalies

3.3.3 Influence-Aware (Confidence-Based) Poisoning

Influence-aware poisoning targets training samples that are most influential for shaping the model's decision boundary, based on the observation that samples with low prediction confidence tend to lie near regions of classification uncertainty. In this strategy, a baseline Multi-Layer Perceptron (MLP) is used to estimate prediction confidence for each training instance, with the architecture described in Section 3.2.3 and chosen for its computational efficiency and sufficient expressive capacity. The baseline MLP is trained for two epochs on the clean training data using the Adam optimizer with a learning rate of 0.001 and a batch size of 128. After training, confidence scores are computed for all training samples as the maximum predicted class probability:

$$\mathrm{conf}(x_i) \;=\; \max_{c \in \{0,1\}} \; p_\theta(y = c \mid x_i), \tag{7}$$

where $p_\theta(\cdot \mid x)$ denotes the model's softmax output. Samples are ranked in ascending order of $\mathrm{conf}(x_i)$, and the lowest B samples (Eq. 4) are selected for poisoning; bidirectional label flipping is then applied (Eq. 5). By corrupting instances that lie close to the decision boundary, this strategy forces the model to shift or distort its boundary during retraining, potentially creating localized regions of systematic misclassification.

3.3.4 Disagreement-Based Poisoning

Disagreement-based poisoning exploits instability in the learning process by targeting samples on which multiple independently trained models produce conflicting predictions. The underlying hypothesis is that such samples represent ambiguous regions of the feature space or areas where the learning algorithm is sensitive to initialization and stochasticity. Two MLP models with identical architectures are trained independently for two epochs on the clean training data using different random seeds. For each training sample x_i, let $\hat{y}_i^{(1)}$ and $\hat{y}_i^{(2)}$ denote the predicted labels from the two models, and let $\mathrm{conf}^{(1)}(x_i)$ and $\mathrm{conf}^{(2)}(x_i)$ denote their confidence scores (Eq. 7). Candidate samples are defined as those for which the predicted labels disagree:

$$\mathcal{C}_{\mathrm{dis}} \;=\; \{\, i \;:\; \hat{y}_i^{(1)} \neq \hat{y}_i^{(2)} \,\}. \tag{8}$$

These candidates are ranked by a disagreement score, defined as the absolute difference in confidence:

$$s_i \;=\; \left|\mathrm{conf}^{(1)}(x_i) - \mathrm{conf}^{(2)}(x_i)\right|. \tag{9}$$

The top B candidates by s_i are selected for poisoning; if $|\mathcal{C}_{\mathrm{dis}}| < B$, the remaining budget is filled by uniform random sampling from non-candidates. Bidirectional label flipping is then applied (Eq. 5). By poisoning samples that already induce disagreement between models, this strategy amplifies learning instability and can lead to inconsistent decision boundaries, particularly in regions where benign and malicious behaviors overlap.

3.3.5 Temporal Window Poisoning

Temporal window poisoning is designed to exploit the temporal structure inherent in network traffic and is applicable exclusively to the CUPID dataset, which includes reliable timestamp information. The strategy operates under the premise that anomalous or attack-related traffic often manifests as bursts or deviations within localized time windows. Network flows are first sorted chronologically, and samples with invalid timestamps are removed to preserve temporal integrity. The timeline is then segmented into fixed-duration windows of five minutes. For each window w, summary statistics are computed over a set of temporal anomaly features $\mathcal{F}$ (e.g., flow duration, packet counts, and byte volumes), including the mean $\mu_{w,f}$ and standard deviation $\sigma_{w,f}$ for each feature $f \in \mathcal{F}$. Each sample x_i belonging to window $w(i)$ is assigned a deviation score based on standardized distances:

$$d_i \;=\; \sum_{f \in \mathcal{F}} \left| \frac{x_{i,f} - \mu_{w(i),f}}{\sigma_{w(i),f} + \varepsilon} \right|, \tag{10}$$

where $\varepsilon > 0$ is a small constant to ensure numerical stability. Samples are ranked globally by d_i, and the top B highest-deviation samples are selected and subjected to bidirectional label flipping (Eq. 5). By targeting traffic that deviates strongly from local temporal norms, this strategy aims to corrupt the association between time-localized anomalies and malicious behavior, potentially degrading the model's ability to detect bursty or time-sensitive attacks.

3.4 Training Procedures

Training procedures differ between neural network models and statistical models due to their distinct optimization mechanisms and convergence behaviors. Accordingly, separate but internally consistent training pipelines were adopted for each model family to ensure fair comparison across poisoning strategies and datasets.

3.4.1 Neural Network Models (MLP, RNN, CNN)

All neural network models (MLP, 1D-CNN, and RNN) were trained using a unified optimization and evaluation pipeline. Training was performed using the Adam optimizer with default momentum parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and numerical stability constant $\epsilon = 10^{-8}$. A fixed learning rate of 0.001 and a batch size of 128 samples were used across all experiments. Models were trained for three epochs using the cross-entropy loss function

$$\mathcal{L}(\theta) \;=\; -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}), \tag{11}$$

where N denotes the batch size, $C = 2$ is the number of classes, $y_{i,c}$ is the one-hot encoded ground-truth label, and $\hat{y}_{i,c}$ is the predicted probability for class c. During training, mini-batches were processed in shuffled order, gradients were computed via backpropagation, and parameters were updated after each batch using Adam. Following completion of the final epoch, models were evaluated on the held-out test set without gradient computation, and standard performance metrics including accuracy, confusion matrices, and class-specific measures were recorded.

Regularization was applied exclusively through dropout with probability 0.3 during training, which was disabled at evaluation time. No explicit weight decay or additional ℓ_2 regularization was employed. The choice to limit training to three epochs was deliberate, as extended training increases the likelihood of memorization

of poisoned labels, which can artificially amplify the observed impact of poisoning attacks. All neural models were trained using single-precision floating-point arithmetic (FP32) on a CUDA-enabled GPU when available, otherwise on CPU.

3.4.2 Statistical Models (LR, RF)

Statistical models were trained using the standard fitting procedures provided by the scikit-learn library, reflecting their fundamentally different optimization paradigms. Logistic Regression models were optimized using the L-BFGS solver with ℓ_2 regularization and a maximum of 1000 iterations, although convergence was typically achieved well before this limit. Random Forest models were trained as ensembles of 100 decision trees, each constructed independently using bootstrap sampling and greedy split selection based on Gini impurity. Feature subsampling and bootstrap aggregation were employed to encourage ensemble diversity, and training was parallelized across all available CPU cores to improve efficiency. Unlike neural networks, statistical models do not rely on epoch-based gradient descent; Logistic Regression converges through quasi-Newton optimization, while Random Forest training proceeds through recursive tree construction. These distinct training dynamics enable analysis of poisoning effects across fundamentally different learning mechanisms.

3.5 Defense Mechanisms

To mitigate the impact of poisoning attacks on model training, we implement two complementary defense strategies: *removal* and *reweighting*. Both defenses identify suspicious training samples based on their loss values under a reference model, motivated by the assumption that poisoned or mislabeled samples tend to incur disproportionately high loss. The two strategies differ in how such samples are handled, either by excluding them entirely from training or by reducing their influence while retaining them in the dataset.

3.5.1 Removal Defense

The removal defense is motivated by the observation that high-loss samples are more likely to be corrupted, mislabeled, or adversarial, and that eliminating such samples can effectively clean the training data. For neural network models, per-sample loss estimation is performed using a temporary MLP trained on the poisoned dataset for two epochs. After this brief training phase, per-sample cross-entropy losses ℓ_i are computed for each training instance. To prevent bias arising from class imbalance, losses are processed in a class-aware manner: within each class $c \in \{0, 1\}$, samples are ranked by loss and the top $p\%$ highest-loss samples are identified for removal. Denoting the index set of class-c samples by $\mathcal{I}_c$ and its size by $|\mathcal{I}_c|$, the class-wise removal count is

$$B_c \;=\; \lfloor p\,|\mathcal{I}_c| \rfloor\,, \tag{12}$$

and the removed set for class c is defined as the B_c largest-loss samples in $\mathcal{I}_c$. A new training dataset is then constructed excluding these samples, and the final model (matching the original task architecture) is trained on the cleaned dataset using the standard three-epoch procedure.

For statistical models, per-sample loss estimation is computed using stratified K-fold cross-validation with $K = 4$ to reduce memorization bias. In each fold, the model is trained on the training split and evaluated on the held-out split, producing out-of-fold predicted probabilities $\hat{p}_i = P_\theta(y = 1 \mid x_i)$. A cross-entropy loss is then computed per sample as

$$\ell_i \;=\; -\Big(y_i \log(\hat{p}_i + \varepsilon) + (1 - y_i) \log(1 - \hat{p}_i + \varepsilon)\Big), \tag{13}$$

with a small ε for numerical stability. Class-aware removal (Eq. 12) is then applied prior to retraining the final model on the filtered dataset. Overall, the removal defense is expected to reduce the influence of poisoned samples at the cost of reducing the effective training set size, which may in turn affect generalization performance.

3.5.2 Reweighting Defense

The reweighting defense follows the same underlying motivation as removal, namely that high-loss samples are more likely to be adversarial, but seeks to mitigate their impact without discarding data entirely. For neural network models, the same temporary MLP-based loss estimation procedure used in the removal defense is

employed to obtain per-sample losses ℓ_i, which are then processed in a class-aware manner to identify the top $p\%$ highest-loss samples per class. Instead of removing these samples, we assign them a reduced weight $\alpha = 0.1$, while all other samples retain weight 1.0. Denoting the weight for sample i by w_i, the weighted training objective becomes

$$\mathcal{L}_w(\theta) \;=\; \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \, \ell_i(\theta), \tag{14}$$

where $\ell_i(\theta)$ is the per-sample cross-entropy loss (Eq. 11 without batch averaging). The final model is then trained for three epochs using this weighted objective, thereby reducing the influence of suspicious samples while preserving potentially informative data.

For statistical models, per-sample loss values are estimated using four-fold stratified cross-validation (Eq. 13), after which class-aware sample weights are assigned following the same scheme. Model training then proceeds using the weighted fitting interface provided by the learning algorithm. The choice of $\alpha = 0.1$ reflects a compromise between completely ignoring suspicious samples (equivalent to removal) and treating them equally to clean samples, enabling partial retention of information while substantially limiting their impact on the learned decision boundary.

4 Evaluation

This section presents the evaluation methodology, performance metrics, and experimental scope used to assess the impact of poisoning attacks and the effectiveness of the proposed defense mechanisms. The evaluation is designed to capture not only overall classification performance, but also the specific error modes induced by poisoning, with particular emphasis on attack detection reliability.

4.1 Evaluation Methodology

The experimental evaluation follows a structured protocol aimed at analyzing three core aspects: baseline performance on clean (unpoisoned) datasets, degradation in performance under different poisoning strategies and poisoning rates, and recovery of performance when defenses are applied. For each dataset–model pair, a clean baseline is first established. Poisoned variants of the training data are then generated according to the strategies described in Section 3.3, and models are retrained and evaluated under identical conditions. Finally, defense mechanisms are applied to the poisoned datasets, and models are retrained to quantify performance recovery relative to both poisoned and clean baselines. To enable systematic analysis across the large experimental space, all evaluation outputs are recorded in a structured JSON format. Each result file stores the dataset, model, poisoning strategy, poisoning rate, defense configuration, test-set performance metrics, confusion matrix statistics, and relevant training metadata. This standardized storage format facilitates automated aggregation, comparison, and post hoc analysis across datasets, models, and attack scenarios.

4.2 Evaluation Metrics

Model performance is evaluated using a suite of complementary metrics derived from the confusion matrix, rather than relying solely on overall accuracy. While accuracy is commonly reported, it can be misleading in the presence of class imbalance or asymmetric error costs, both of which are characteristic of network intrusion detection settings. For example, a classifier that predicts all traffic as benign may achieve high accuracy on benign-dominated datasets while completely failing to detect attacks.

We report the full binary confusion matrix consisting of true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP), where benign traffic corresponds to the negative class and malicious traffic corresponds to the positive class:

$$\begin{bmatrix} \mathrm{TN} & \mathrm{FP} \\ \mathrm{FN} & \mathrm{TP} \end{bmatrix}. \tag{15}$$

Overall accuracy is computed as

$$\mathrm{Acc} \;=\; \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{16}$$

Class-specific precision, recall, and F1-score for the malicious (attack) class are computed as

$$\text{Precision}_1 = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall}_1 = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{17}$$

$$\text{F1}_1 = 2 \cdot \frac{\text{Precision}_1 \cdot \text{Recall}_1}{\text{Precision}_1 + \text{Recall}_1}. \tag{18}$$

In NIDS applications, false negatives (missed attacks) are typically more costly than false positives (false alarms). Accordingly, we emphasize attack recall (true positive rate) and the false negative rate,

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}, \tag{19}$$

when interpreting poisoning impact and defense effectiveness. In addition to per-class metrics, macro-averaged and weighted-averaged scores are reported to account for both balanced and imbalanced class distributions. Macro averages treat benign and attack classes equally, whereas weighted averages reflect the underlying class prevalence, jointly providing a detailed view of how poisoning attacks affect different error modes and how defenses alter these trade-offs.

4.3 Interpretation of Evaluation Results

When analyzing experimental results, poisoning success is primarily indicated by a reduction in attack recall, an increase in the false negative rate, and a corresponding decrease in the F1-score of the attack class. In some cases, poisoning may also increase false positives, reducing attack precision and signaling confusion between benign and malicious traffic. Defense effectiveness is assessed by the extent to which these metrics recover toward clean baseline values, particularly with respect to attack recall and false negative rate. Conversely, overly aggressive defenses may manifest as inflated false positive rates, indicating a trade-off between security sensitivity and operational usability. These interpretative principles are applied consistently across all reported experiments.

4.4 Experimental Scale and Coverage

The evaluation encompasses four benchmark datasets (UNSW-NB15, CIC-IDS2017, CUPID, and CIDDS-001), five learning models (Logistic Regression, Random Forest, MLP, CNN, and RNN), and multiple poisoning strategies, including class hiding, feature-targeted poisoning, influence-aware poisoning, disagreement-based poisoning, and temporal window poisoning (for CUPID). Each poisoning strategy is evaluated at three poisoning rates (5%, 10%, and 20%), both with and without defenses. Two defense mechanisms (removal and reweighting) are applied alongside a clean baseline with no poisoning. This results in several hundred distinct experimental configurations, providing comprehensive coverage of attack surfaces, model architectures, and defensive behaviors.

5 Results

This section presents a comprehensive analysis of our experimental findings, organized by dataset to facilitate direct comparison across poisoning strategies, model architectures, and defense mechanisms. We first establish baseline performance on clean data, then examine how each dataset responds to poisoning attacks, and conclude with a cross-dataset synthesis highlighting key patterns and vulnerabilities.

5.1 Baseline Performance on Clean Data

Before examining poisoning effects, we establish baseline performance for all model-dataset combinations trained on clean (unpoisoned) data. Table 3 summarizes test accuracy, macro F1-score, and attack recall for each configuration.

Several observations emerge from the baseline results. CIC-IDS2017 achieves uniformly high performance across all models, with attack recall exceeding 91% for all architectures and Random Forest achieving near-perfect detection (99.7%). UNSW-NB15 shows moderately lower accuracy (81–87%) but maintains excellent attack recall (97–99%), indicating that while overall classification is more challenging, malicious

Table 3: Baseline performance on clean (unpoisoned) training data across all datasets and models. Attack recall measures the proportion of malicious traffic correctly identified.

Dataset	Model	Accuracy	Macro F1	Attack Recall
CIC-IDS2017	CNN	0.981	0.971	0.984
	RNN	0.984	0.975	0.972
	MLP	0.978	0.964	0.927
	LR	0.971	0.951	0.914
	RF	0.999	0.998	0.997
UNSW-NB15	CNN	0.847	0.838	0.987
	RNN	0.817	0.805	0.971
	MLP	0.823	0.810	0.984
	LR	0.809	0.794	0.971
	RF	0.873	0.867	0.986
CUPID	CNN	0.957	0.935	0.816
	RNN	0.955	0.934	0.845
	MLP	0.957	0.936	0.820
	LR	0.916	0.872	0.679
	RF	0.990	0.986	0.971
CIDDS-001	CNN	0.997	0.561	0.069
	RNN	0.997	0.499	0.000
	MLP	0.997	0.499	0.000
	LR	0.997	0.563	0.069
	RF	0.999	0.948	0.893

traffic is reliably detected. CUPID exhibits strong overall performance but lower attack recall (68–97%), reflecting the inherent difficulty of IoT traffic classification.

Most notably, CIDDS-001 reveals a critical baseline failure: despite achieving 99.7% accuracy, most models exhibit catastrophically low attack recall (0–7%), with macro F1-scores near 50%. This indicates that models are predicting nearly all traffic as benign, achieving high accuracy due to the extreme class imbalance (∼367:1 benign-to-attack ratio) while completely failing to detect attacks. Only Random Forest achieves meaningful attack detection (89.3% recall) on CIDDS-001, suggesting that ensemble methods may be more robust to extreme imbalance. This baseline failure has important implications: CIDDS-001 results cannot be meaningfully interpreted in terms of poisoning-induced degradation because the models are already non-functional as intrusion detectors.

5.2 CIC-IDS2017: The Accuracy Illusion

CIC-IDS2017 serves as our primary dataset for demonstrating the severity of label poisoning attacks due to its combination of strong baseline performance and significant vulnerability to targeted attacks. The key finding is the emergence of an *accuracy illusion*, where poisoned models maintain deceptively high overall accuracy while completely failing to detect attacks.

5.2.1 Class Hiding Attack

Table 4 presents the impact of class hiding (random label flipping) on CNN performance, illustrating the progressive collapse of attack detection capability.

The results reveal a striking pattern. At 5% poisoning, attack recall drops from 98.4% to 93.3%, which is noticeable but manageable. However, at 10% poisoning, a catastrophic collapse occurs: attack recall plummets to 16.2%, meaning the model misses 84% of all attacks. At 20% poisoning, the collapse is complete: attack recall reaches exactly 0%, with all 111,529 attack samples in the test set misclassified as benign (FN = 111,529, TP = 0).

Table 4: CNN performance on CIC-IDS2017 under class hiding attack at varying poisoning rates, with and without defenses.

Poison %	Defense	Accuracy	Macro F1	Attack Recall	FN Count
0% (Clean)	—	0.981	0.971	0.984	1,783
5%	None	0.976	0.963	0.933	7,475
5%	Removal	0.977	0.965	0.941	6,582
5%	Reweight	0.977	0.965	0.938	6,917
10%	None	0.855	0.683	0.162	93,461
10%	Removal	0.840	0.611	0.102	100,150
10%	Reweight	0.850	0.654	0.138	96,138
20%	None	0.803	0.445	0.000	111,529
20%	Removal	0.803	0.445	0.000	111,529
20%	Reweight	0.803	0.445	0.000	111,529

Table 5: Comparison of poisoning strategies on CNN for CIC-IDS2017 at 10% poisoning rate (no defense).

Strategy	Accuracy	Macro F1	Attack Recall
Clean (0%)	0.981	0.971	0.984
Class Hiding	0.855	0.683	0.162
Feature-Targeted	0.973	0.961	0.934
Confidence-Based	0.976	0.965	0.934
Disagreement	0.976	0.964	0.932

Critically, accuracy remains above 80% throughout this collapse. A practitioner monitoring only accuracy would observe a decline from 98.1% to 80.3% and might conclude the model is "somewhat degraded but still functional." In reality, the model is entirely blind to attacks, effectively a useless intrusion detector masquerading as a moderately accurate classifier. This accuracy illusion arises because CIC-IDS2017 has approximately 4:1 benign-to-attack ratio; a model that classifies everything as benign achieves ∼80% accuracy by default.

Defense mechanisms proved largely ineffective. At 10% poisoning, removal defense actually worsened attack recall (from 16.2% to 10.2%), while reweighting provided only marginal improvement. At 20% poisoning, both defenses completely failed, producing identical zero-recall results. This suggests that once the poisoning rate exceeds a critical threshold, current defense mechanisms are overwhelmed.

5.2.2 Comparison Across Attack Strategies

Table 5 compares the effectiveness of different poisoning strategies on CNN at 10% poisoning rate, revealing that class hiding is the most effective attack.

Surprisingly, class hiding, the simplest strategy involving random label flipping, proves dramatically more effective than the sophisticated, model-aware strategies (feature-predicate, confidence-based, and disagreement-based). While class hiding reduces attack recall to 16.2%, the targeted strategies only reduce it to 93–94%, representing minimal degradation from the clean baseline. This paradoxical finding suggests that the random nature of class hiding may produce a more uniform "poisoning" of the feature space, whereas targeted strategies concentrate perturbations in specific regions that models can learn to circumvent.

5.2.3 Model Architecture Comparison

Contrary to the expectation that all models would exhibit similar vulnerability, Table 6 reveals substantial differences in attack recall under class hiding at intermediate poisoning rates.

At 10% poisoning, RNN maintains 56.1% attack recall, which is more than 3.5× higher than CNN's

Table 6: Comparison of all model architectures on CIC-IDS2017 under class hiding attack, highlighting divergent vulnerability patterns at 10% poisoning.

Model	5% Poisoning		10% Poisoning		20% Poisoning	
	Accuracy	Recall	Accuracy	Recall	Accuracy	Recall
CNN	0.976	0.920	0.855	0.162	0.803	0.000
RNN	0.962	0.960	0.913	**0.561**	0.803	0.000
MLP	0.963	0.893	0.841	0.192	0.803	0.000
LR	0.913	0.610	0.843	0.207	0.803	0.000
RF	0.981	0.906	0.889	**0.437**	0.803	0.000

Table 7: Model comparison on UNSW-NB15 under 20% class hiding attack, showing MLP's superior resilience.

Model	Accuracy	Macro F1	Attack Recall	Δ Recall
CNN (Clean)	0.847	0.838	0.987	—
CNN	0.870	0.867	0.926	−0.061
RNN	0.834	0.829	0.911	−0.070
MLP	0.827	0.818	**0.958**	−0.035
LR	0.825	0.821	0.898	−0.041
RF	0.885	0.884	0.900	−0.086

16.2%. Similarly, Random Forest retains 43.7% recall, nearly 3× CNN's performance. This divergence has important practical implications: model selection can provide a meaningful first line of defense against moderate poisoning attacks. However, at 20% poisoning all models converge to complete failure, with attack recall reaching 0% (or 0.001% for RF, detecting only 1 of 111,529 attacks). The defense visualizations in Figures 2–6 illustrate these architecture-specific degradation patterns across all datasets.

Interestingly, the removal defense shows dramatically different effectiveness across architectures. For LR at 10% poisoning, removal defense recovers attack recall from 20.7% to 76.9%, an improvement of 56 percentage points. For RF at 10%, removal achieves 89.8% recall (from 43.7%), approaching clean baseline performance. These successful defense cases stand in stark contrast to the complete defense failure observed for CNN, suggesting that defense effectiveness may depend critically on model architecture.

5.3 UNSW-NB15: Relative Resilience

UNSW-NB15 exhibits substantially greater resilience to poisoning attacks compared to CIC-IDS2017, despite having lower baseline accuracy. Table 7 compares all model architectures under class hiding attack at 20% poisoning, revealing interesting architectural differences.

Even at 20% poisoning, all models maintain attack recall above 89%, representing only a 4–9 percentage point drop from clean baselines. This stands in stark contrast to CIC-IDS2017, where 20% poisoning caused complete detection failure. Most notably, *MLP achieves 95.8% attack recall*, the highest among all architectures and only 3.5 percentage points below its clean baseline. This suggests that MLP's simpler architecture may be more robust to label noise on balanced datasets.

The key difference from CIC-IDS2017 appears to be class balance: UNSW-NB15 has approximately 1:1 benign-to-attack ratio, meaning models cannot achieve high accuracy by simply predicting all traffic as benign. This forces models to genuinely learn discriminative features even under poisoning. Interestingly, CNN's accuracy and macro F1 actually *increased* at 20% poisoning compared to clean baseline (0.870 vs. 0.847), possibly reflecting the model finding alternative decision boundaries when original features become unreliable.

Table 8: CNN performance on CUPID under temporal window poisoning versus class hiding.

Strategy	Poison %	Accuracy	Macro F1	Attack Recall
Clean	0%	0.957	0.935	0.816
Class Hiding	5%	0.949	0.920	0.781
Class Hiding	10%	0.933	0.891	0.725
Class Hiding	20%	0.767	0.434	0.000
Temporal	5%	0.907	0.876	0.807
Temporal	10%	0.857	0.820	0.812
Temporal	20%	0.832	0.776	0.787

Table 9: CIDDS-001 baseline confusion matrix statistics showing near-complete failure to detect attacks for neural network models.

Model	TN	FP	FN	TP	Accuracy	Attack Recall
CNN	1,306,046	247	3,315	247	0.997	0.069
RNN	1,306,293	0	3,562	0	0.997	0.000
MLP	1,306,292	1	3,562	0	0.997	0.000
LR	1,306,047	246	3,316	246	0.997	0.069
RF	1,305,942	351	381	3,181	0.999	0.893

5.4 CUPID: Temporal Poisoning Effects

CUPID is the only dataset with reliable timestamp information, enabling evaluation of temporal window poisoning. Table 8 compares temporal poisoning with class hiding on CNN performance.

Temporal poisoning produces more gradual degradation than class hiding, with attack recall decreasing from 81.6% to 78.7% across the full range of poisoning rates. However, temporal poisoning also causes substantial accuracy degradation (from 95.7% to 83.2%), making the attack more detectable through standard monitoring. Class hiding again produces catastrophic failure at 20% poisoning (0% attack recall), consistent with findings on CIC-IDS2017.

Figure 1 visualizes the macro F1 degradation under temporal window poisoning across all models, showing that MLP and RNN are particularly affected, while Random Forest maintains stronger resilience.

5.5 CIDDS-001: Baseline Failure

As noted in Section 5.1, CIDDS-001 exhibits fundamental baseline failure for most models. Table 9 provides detailed confusion matrix statistics demonstrating this failure.

The test set contains only 3,562 attack samples out of 1,309,855 total (0.27%), creating a $\sim$367:1 class imbalance. Neural network models (CNN, RNN, MLP) and Logistic Regression essentially predict all traffic as benign, achieving 99.7% accuracy while detecting at most 7% of attacks. Only Random Forest achieves meaningful detection (89.3% recall), likely due to its ensemble structure and ability to model minority class patterns through multiple independent trees.

Because the baseline is already non-functional for most models, poisoning results on CIDDS-001 are not meaningful because one cannot "degrade" a detector that already fails to detect. This finding underscores the importance of evaluating NIDS on multiple datasets with varying class distributions and cautions against drawing conclusions from accuracy alone.

5.6 Cross-Dataset Synthesis

Table 10 synthesizes key findings across all datasets, presenting CNN performance under class hiding at 20% poisoning rate.

Table 10: Cross-dataset comparison of CNN under 20% class hiding poisoning.

Dataset	Clean Recall	Poisoned Recall	Δ Recall	Accuracy
CIC-IDS2017	0.984	0.000	−0.984	0.803
CUPID	0.816	0.000	−0.816	0.767
UNSW-NB15	0.987	0.935	−0.052	0.851
CIDDS-001	0.069	—	—	0.997

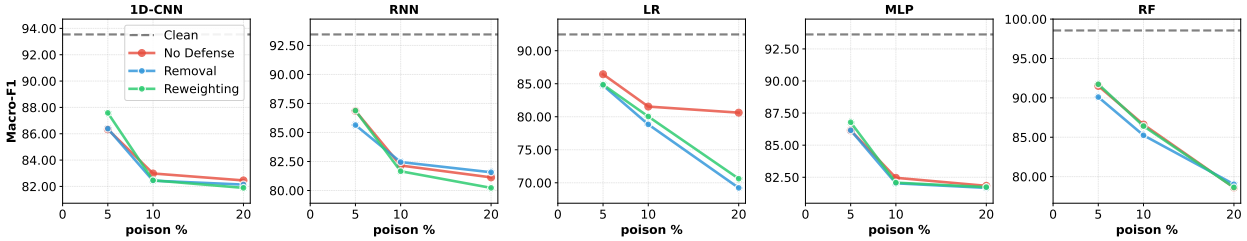

Figure 1: Temporal Window Poisoning Macro F1 for CUPID dataset across all models.

The results reveal a clear vulnerability hierarchy: CIC-IDS2017 and CUPID are highly susceptible to class hiding attacks, with complete detection failure at 20% poisoning, while UNSW-NB15 maintains robust detection even under heavy poisoning. The key differentiating factor appears to be class balance: datasets with significant benign majority (CIC-IDS2017 at 4:1, CUPID at 2:1) enable the accuracy illusion, whereas balanced datasets (UNSW-NB15 at 1:1) force models to maintain genuine discriminative capability.

Figures 2–6 provide comprehensive visualizations of defense effectiveness across all model architectures, showing that while defenses provide some protection at low poisoning rates, they consistently fail to prevent catastrophic degradation at higher rates.

6 Discussion

This section interprets our experimental findings in the broader context of NIDS deployment, identifies key lessons for practitioners and researchers, and acknowledges the limitations of our study.

6.1 The Accuracy Illusion: A Critical Warning

Our most important finding is the *accuracy illusion*, the phenomenon where a poisoned model maintains high overall accuracy while completely failing its primary objective of attack detection. On CIC-IDS2017, a CNN trained on 20% poisoned data achieved 80.3% accuracy but 0% attack recall. This is not merely poor performance; it represents a fundamentally broken detector that would provide false assurance to operators.

The accuracy illusion arises from the interaction between class imbalance and targeted poisoning. When benign traffic substantially outnumbers attack traffic (as in most real networks), a model can achieve acceptable accuracy by learning to classify most or all traffic as benign. Label poisoning accelerates this degenerate solution by corrupting the association between attack features and malicious labels, making the benign-only prediction increasingly attractive to the optimizer.

For practitioners, this finding carries an urgent message: **accuracy is not a reliable health metric for NIDS**. Organizations should monitor attack-specific metrics (recall, F1-score, false negative rate) and establish minimum acceptable thresholds. A NIDS with 95% accuracy but 10% attack recall is worse than useless because it provides false confidence while missing 90% of actual intrusions.

6.2 The Paradox of Simple Attacks

Counter to intuition, the simplest poisoning strategy, class hiding (random label flipping), proved far more effective than sophisticated, model-aware strategies. At 10% poisoning on CIC-IDS2017, class hiding reduced

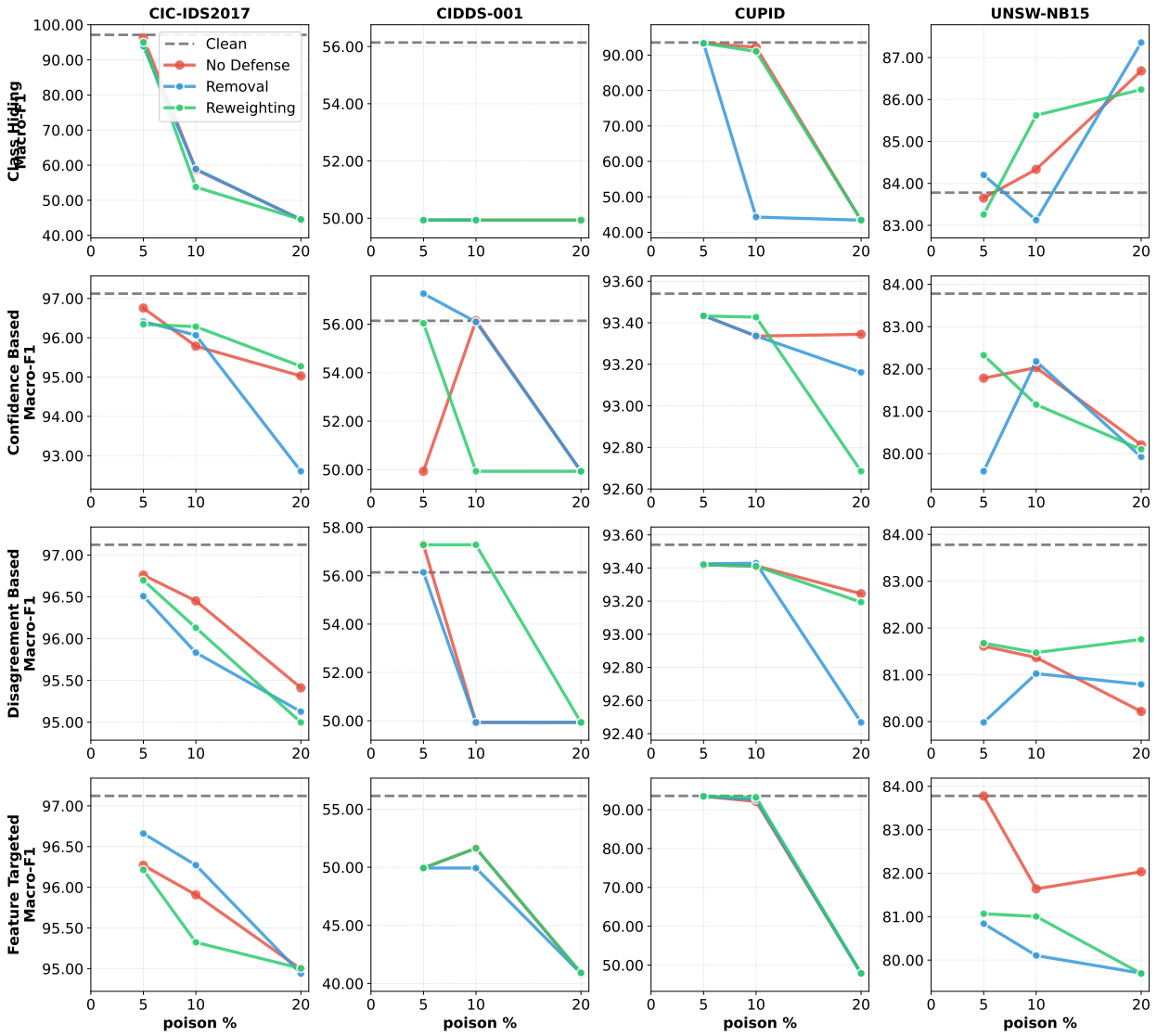

Figure 2: Defense effectiveness for 1D-CNN across datasets and poisoning strategies.

attack recall to 16%, while feature-predicate, confidence-based, and disagreement-based strategies only reduced it to 93–94%.

This paradox may have several explanations. First, random poisoning distributes corruption uniformly across the feature space, preventing the model from learning stable decision boundaries anywhere. Targeted strategies, by contrast, concentrate perturbations in specific regions, leaving most of the feature space unaffected. Second, the targeted strategies were designed based on assumptions (e.g., that high-influence or boundary samples are most important) that may not hold for neural networks trained with early stopping. Third, the models may be learning to ignore or down-weight the specific feature regions targeted by sophisticated attacks.

From a security perspective, this finding is concerning: effective attacks do not require insider knowledge, model access, or sophisticated optimization. An adversary who can flip 10–20% of training labels at random, potentially through simple data poisoning of crowdsourced labels, compromised honeypots, or corrupted logging infrastructure, can render a NIDS effectively blind to attacks.

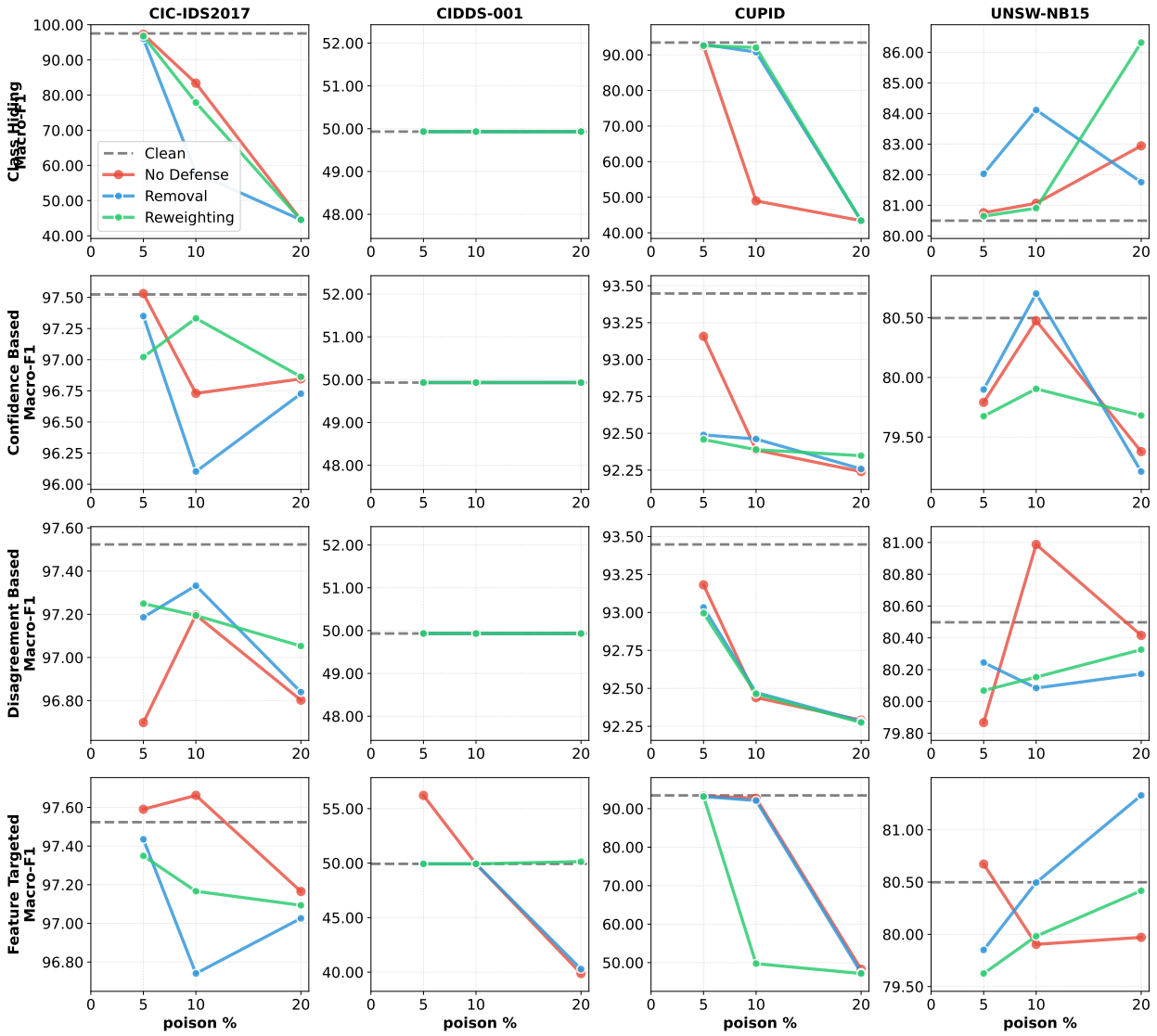

Figure 3: Defense effectiveness for RNN across datasets and poisoning strategies.

6.3 Dataset Vulnerability Patterns

Our cross-dataset analysis reveals that vulnerability to label poisoning is not uniform but depends critically on dataset characteristics:

Class imbalance amplifies vulnerability. CIC-IDS2017 (4:1) and CUPID (2:1) exhibited catastrophic failure under 20% poisoning, while UNSW-NB15 (1:1) maintained robust detection. The balanced dataset forces models to learn genuine discriminative features because the "predict all benign" shortcut achieves only 50% accuracy.

Extreme imbalance prevents meaningful evaluation. CIDDS-001 (367:1) exhibited baseline failure: models achieved 99.7% accuracy while detecting fewer than 7% of attacks even on clean data. This highlights the inadequacy of standard training procedures for extremely imbalanced scenarios and suggests that specialized techniques (e.g., oversampling, cost-sensitive learning, anomaly detection formulations) are necessary.

Dataset size interacts with poisoning rate. The absolute number of poisoned samples matters: 20% of CIC-IDS2017's 1.98 million training samples is ∼396,000 corrupted labels, whereas 20% of UNSW-NB15's

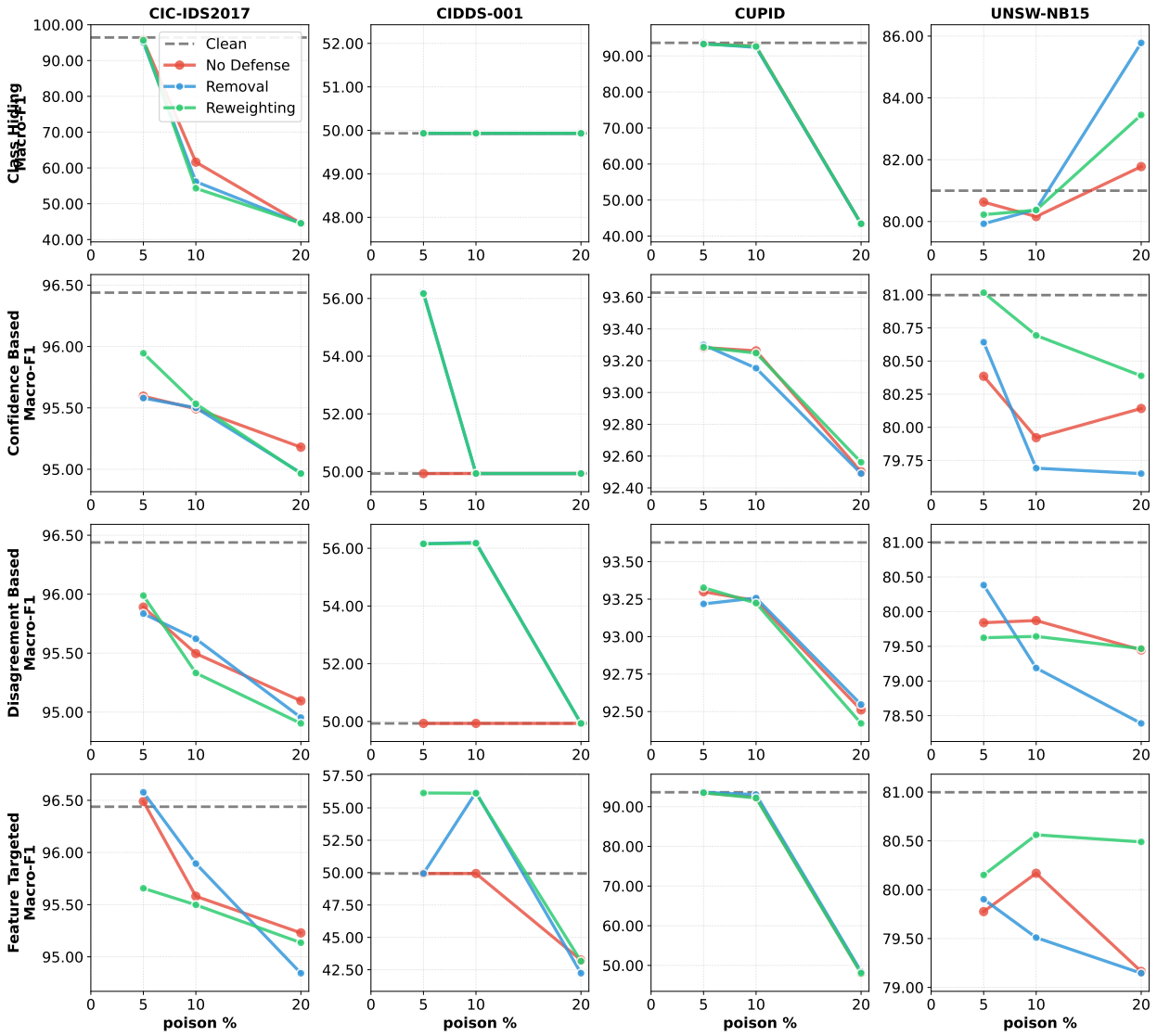

Figure 4: Defense effectiveness for MLP across datasets and poisoning strategies.

175,000 samples is only ∼35,000. Larger datasets may be more vulnerable because they provide more opportunities for poisoning while maintaining sufficient clean samples to prevent obvious model collapse.

6.4 Model Architecture Considerations

While all neural network architectures (CNN, RNN, MLP) ultimately succumb to class hiding at 20% poisoning (achieving 0% attack recall), **substantial differences emerge at intermediate poisoning rates**. On CIC-IDS2017 at 10% poisoning, RNN maintained 56.1% attack recall, which is more than 3.5× higher than CNN's 16.2%. Similarly, Random Forest retained 43.7% recall at 10%, nearly 3× CNN's performance. These differences have important practical implications: model selection provides a meaningful first line of defense against moderate poisoning.

Defense effectiveness also varies dramatically by architecture. For Logistic Regression at 10% poisoning on CIC-IDS2017, the removal defense recovered attack recall from 20.7% to 76.9%, an improvement of 56 percentage points. Random Forest with removal achieved 89.8% recall (from 43.7%), approaching clean baseline performance. These successful defense cases contrast sharply with CNN, where removal actually worsened performance, suggesting that defense effectiveness depends critically on model architecture.

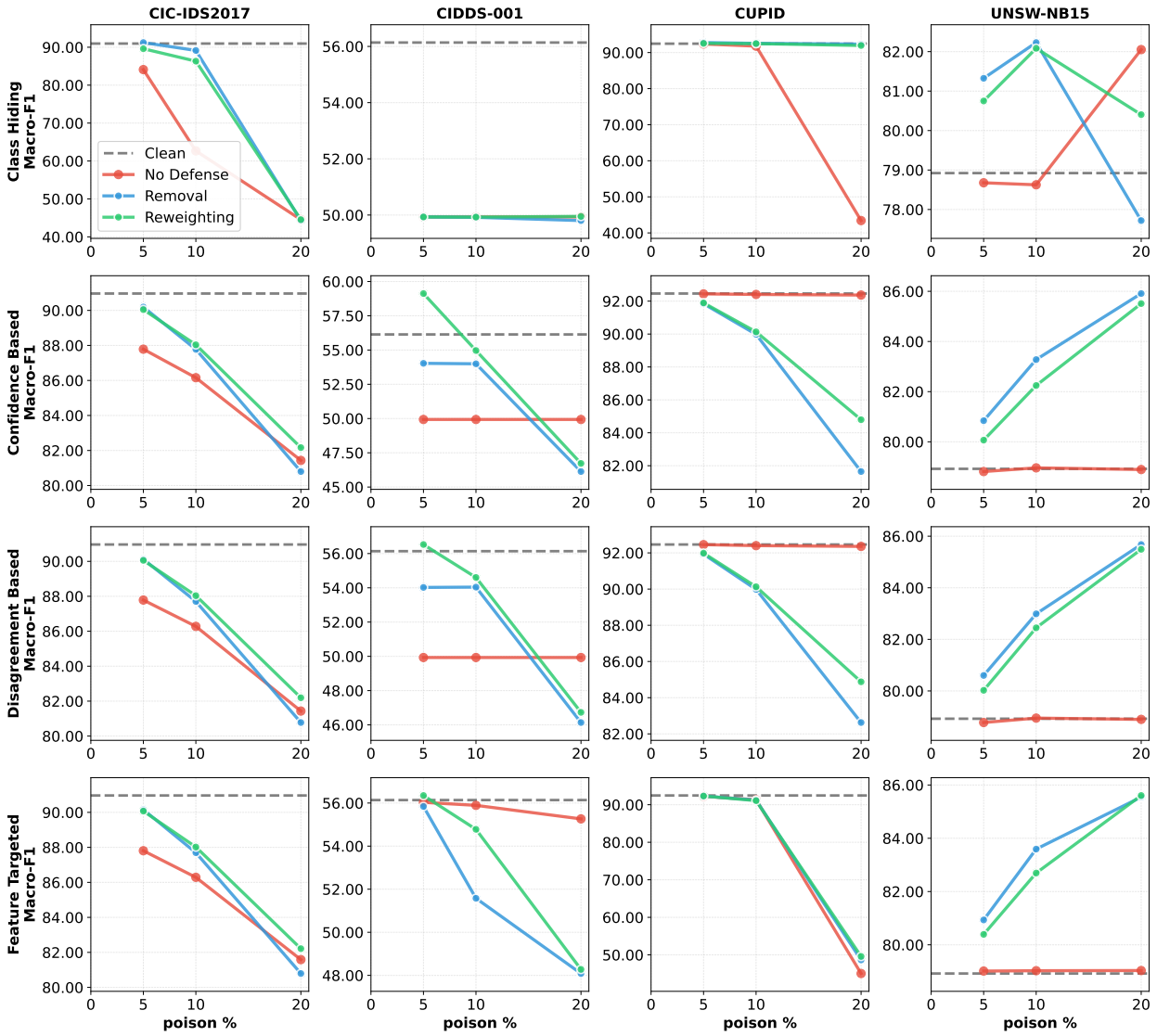

Figure 5: Defense effectiveness for Logistic Regression across datasets and poisoning strategies.

On UNSW-NB15, MLP showed superior resilience, achieving 95.8% attack recall at 20% poisoning, the highest among all architectures and only 3.5 percentage points below its clean baseline. This suggests that simpler architectures may be more robust to label noise on balanced datasets.

Random Forest showed distinct advantages in two scenarios: (1) moderate resilience at intermediate poisoning rates, and (2) meaningful baseline detection on CIDDS-001 where neural networks completely failed. This may reflect the ensemble's ability to maintain diversity; even if some trees are corrupted by poisoned samples, others trained on different bootstrap samples may remain accurate. However, Random Forest is not immune: at 20% poisoning on CIC-IDS2017, it too suffered near-complete failure (detecting only 1 of 111,529 attacks).

6.5 Defense Mechanism Limitations

Both removal and reweighting defenses proved inadequate against moderate-to-heavy poisoning:

Low poisoning rates (5%): Defenses provided modest protection, partially recovering attack recall in some configurations.

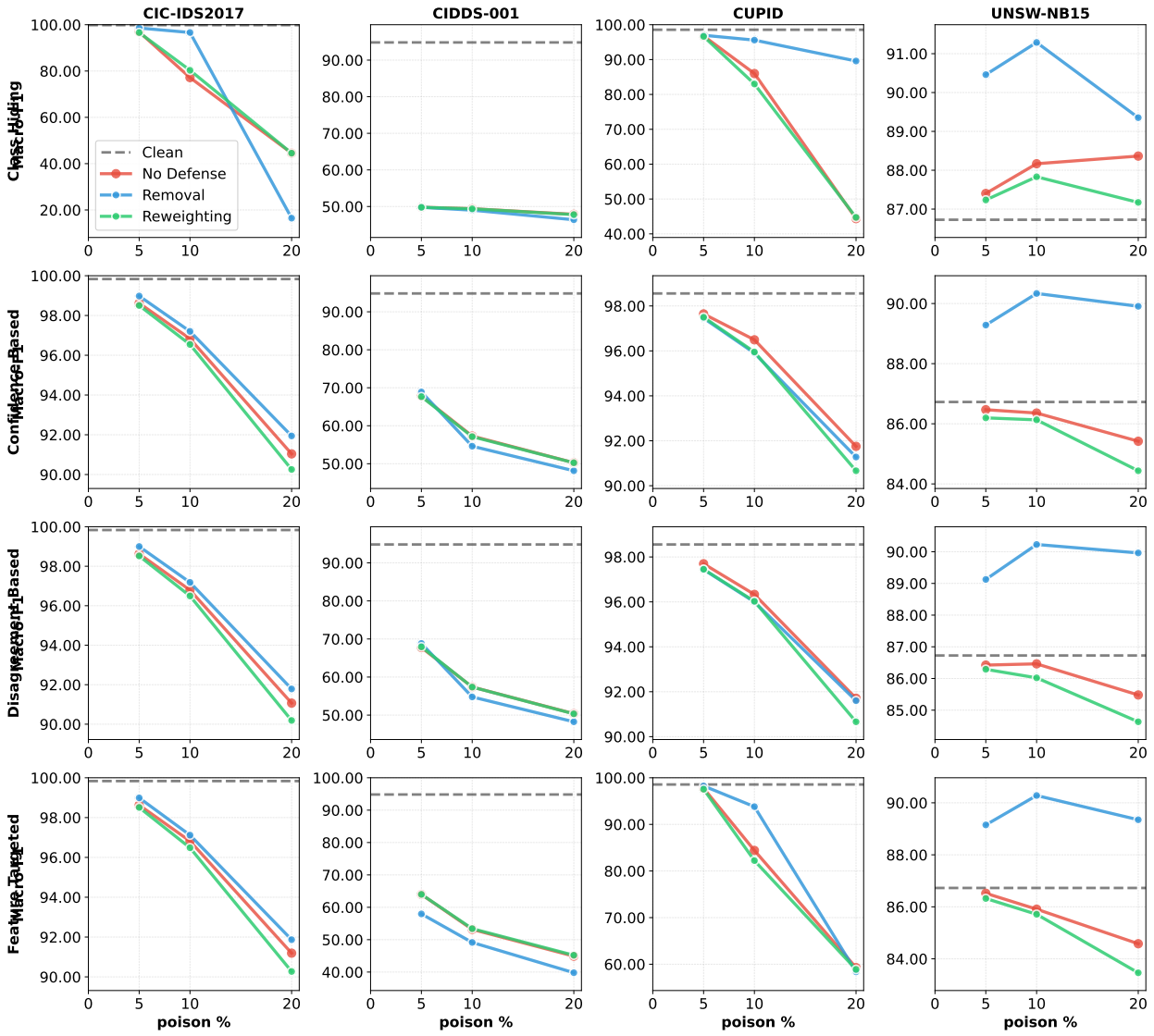

Figure 6: Defense effectiveness for Random Forest across datasets and poisoning strategies.

Moderate poisoning rates (10%): Defenses showed highly architecture-dependent performance. For CNN on CIC-IDS2017, removal defense actually *worsened* attack recall from 16.2% to 10.2%. This counterintuitive result can be explained by the interaction between loss-based filtering and imbalanced data: ideally, loss-based filtering removes poisoned samples, but in deep models like CNNs trained on imbalanced data, the "hard" benign samples (those near the decision boundary or representing rare traffic patterns) often exhibit high loss. Aggressive filtering may inadvertently remove these informative clean samples (false positives in filtering), further degrading the decision boundary and reducing recall. In contrast, for LR and RF, removal defense proved remarkably effective: LR recovered from 20.7% to 76.9% recall, and RF recovered from 43.7% to 89.8%, approaching clean baseline performance. This architecture-specific divergence suggests that simpler models (LR) and ensemble methods (RF) may produce more calibrated loss signals that better distinguish poisoned from hard-but-clean samples.

High poisoning rates (20%): Both defenses completely failed for all architectures, producing results identical to no defense. At this poisoning level, the loss-based identification mechanism cannot distinguish poisoned from clean samples because the model's loss landscape is already dominated by corrupted data.

The fundamental limitation is that both defenses rely on a clean reference model to identify suspicious

samples. When 20% of labels are corrupted, the reference model trained on this data is itself unreliable, leading to circular failure. The architecture-specific success cases at 10% suggest that certain model types may be more amenable to post-hoc defense; further investigation is needed to understand why removal works well for LR and RF but fails for CNN. More robust defenses may require external validation signals, certified training procedures, or fundamentally different learning paradigms.

6.6 Implications for Practitioners

Our findings suggest several practical recommendations for organizations deploying ML-based NIDS:

1. **Monitor attack-specific metrics.** Never rely solely on accuracy. Establish dashboards tracking attack recall, precision, and F1-score, with alerts for significant degradation.
2. **Validate training data provenance.** Treat training data as a security-critical asset. Implement integrity checks, audit trails, and access controls for label databases.
3. **Test with synthetic poisoning.** Before deployment, evaluate model robustness by training on deliberately poisoned variants of the training data. If the model collapses under 10% random label noise, it may be unsuitable for adversarial environments.
4. **Consider class-balanced training.** Resampling or reweighting to achieve approximate class balance may improve robustness by preventing the "predict all benign" shortcut.
5. **Maintain fallback detection.** Do not rely exclusively on ML-based detection. Signature-based and rule-based systems, while less flexible, are not susceptible to label poisoning.

6.7 Limitations and Future Work

Our study has several limitations that suggest directions for future research:

Binary classification only. We converted all datasets to binary (benign vs. malicious) formulation. Multi-class detection, where the model must identify specific attack types, may exhibit different vulnerability patterns.

Limited defense exploration. We evaluated only two defense mechanisms. More sophisticated approaches, including certified defenses, ensemble diversity methods, anomaly-detection-based filtering, and adversarial training paradigms, warrant investigation. Adversarial training, which explicitly incorporates poisoned examples during optimization, represents one of the most promising defense directions but was beyond our current scope.

Static evaluation. We evaluated models at fixed poisoning rates without considering adaptive attacks or defenses. A game-theoretic analysis with iterative attacker-defender interactions would provide deeper insights.

Controlled poisoning. Our experiments assume the attacker can poison exact proportions of training data. Real-world poisoning scenarios may involve more constrained or noisy injection mechanisms.

Single training run. Due to computational constraints, most experiments used single training runs with fixed seeds. Variance analysis across multiple runs would strengthen conclusions.

7 Conclusion

This work presents the first comprehensive cross-dataset evaluation of label poisoning attacks against machine learning-based network intrusion detection systems. Through systematic experiments spanning four benchmark datasets (CIC-IDS2017, UNSW-NB15, CUPID, CIDDS-001), five model architectures (CNN, RNN, MLP, Logistic Regression, Random Forest), five poisoning strategies, and two defense mechanisms, we have characterized the vulnerability landscape of ML-based NIDS to training-time attacks.

Our central finding is the *accuracy illusion*: poisoned models can maintain deceptively high overall accuracy while completely failing to detect attacks. On CIC-IDS2017, class hiding poisoning at 20% reduced attack recall from 98.4% to exactly 0% (the model detected none of the 111,529 attacks in the test set) while accuracy

remained above 80%. This finding has critical implications for NIDS deployment: accuracy is fundamentally unreliable as a health metric, and organizations must monitor attack-specific performance measures.

We identified several key patterns. First, class imbalance amplifies vulnerability: datasets with benign-majority distributions (CIC-IDS2017, CUPID) suffered catastrophic failure under moderate poisoning, while the balanced UNSW-NB15 maintained robust detection. Second, simple attacks are surprisingly effective: random label flipping (class hiding) proved more damaging than sophisticated model-aware strategies, suggesting that effective attacks do not require insider knowledge. Third, current defenses are inadequate: both removal and reweighting mechanisms failed to prevent model collapse at poisoning rates above 10%, highlighting the need for fundamentally new defense approaches.

We also documented a concerning baseline failure on CIDDS-001, where extreme class imbalance (367:1) caused most models to achieve 99.7% accuracy while detecting fewer than 7% of attacks even on clean data. This finding underscores the importance of multi-dataset evaluation and cautions against drawing conclusions from accuracy alone.

For practitioners, we offer concrete recommendations: monitor attack-specific metrics (recall, F1-score) rather than accuracy; validate training data integrity; test model robustness with synthetic poisoning before deployment; consider class-balanced training procedures; and maintain non-ML fallback detection systems. The vulnerability of ML-based NIDS to label poisoning represents a significant operational risk that must be addressed through both technical and procedural safeguards.

Future work should explore certified defenses with provable guarantees, multi-class detection scenarios, adaptive attack-defense dynamics, and the development of poisoning-robust training algorithms. As machine learning becomes increasingly central to network security infrastructure, understanding and mitigating training-time vulnerabilities will be essential to maintaining trustworthy detection capabilities.

8 References

[1] "KDD Cup 1999 Data." `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999. UCI KDD Archive.

[2] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–6, IEEE, 2009.

[3] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[4] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 108–116, SCITEPRESS, 2018.

[5] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[6] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, pp. 305–316, IEEE, 2010.

[7] J. Lawrence, G. Fahimipirehgalin, A. G. West, and A. G. Bardas, "CUPID: A labeled dataset with pentesting for evaluation of network intrusion detection systems," *Journal of Systems Architecture*, vol. 129, p. 102620, 2022.

[8] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, pp. 361–369, 2017.

[9] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[10] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–36, 2021.

[11] P. Goldschmidt and D. Chudá, "Network intrusion datasets: A survey, limitations, and recommendations," *arXiv preprint arXiv:2402.00000*, 2025.

[12] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pp. 1467–1474, 2012.

[13] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, pp. 3517–3529, 2017.

[14] Z. Wang *et al.*, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–36, 2022.

[15] P. Zhao, W. Zhu, P. Jiao, D. Gao, and O. Wu, "Data poisoning in deep learning: A survey," *arXiv preprint arXiv:2501.00000*, 2025.

[16] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," *arXiv preprint arXiv:2207.01982*, 2022.

[17] X. Chang, G. Dobbie, and J. Wicker, "Fast adversarial label-flipping attack on tabular data," *arXiv preprint arXiv:2310.00000*, 2023.

[18] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1885–1894, 2017.

[19] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT)*, pp. 287–294, ACM, 1992.