Available online at https://sextante-ensb.com

Revista Sextante

ISSN : 2665-3923

Volume: 34 Issue: 01 (2026)

Adversarial Machine Learning Attacks and Defense Mechanisms in Intelligent Cybersecurity Systems

Waqar Qais[a]

a. Corresponding Author: AI Automation Engineer, BS Computer Science, Bahauddin Zakariya University, Pakistan, (Waqarqais15@gmail.com)

Article History
Received September 06th, 2026
Accepted November 02nd , 2026
Published November 14th, 2026

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC-BY)license https://creativecommons.org/licenses/by/4.0

Abstract
While deep learning is a fundamental part of intelligent cyber security systems including network intrusion detection systems (NIDS), deep learning models can be adversarially manipulated during the training and inference phases. In this paper, a holistic, reproducible assessment of adversarial machine learning (AML) attacks and defenses of a deep neural network (DNN) based NIDS is presented. We formalize a threat model that distinguishes unconstrained feature-space attackers from domain-constrained ones who only tweak 14 traffic features without compromising the malicious payload, and assess evasion (FGSM, PGD-20, Carlini-Wagner L2), black-box transfer using a query-trained substitute, and label-flipping data poisoning across three random seeds on the NSL-KDD benchmark. In the KDDTest+ split, the baseline DNN achieves a 79.0% accuracy and a 90.7% AUC, while an unconstrained PGD attack with $\varepsilon = 0.2$ causes the attack detection rate of the DNN to drop from 67.9% to 13.4%, with 75.9% of the correctly detected attacks being undetected by the C&W-L2 attack. The threat is less affected by domain constraints (detection 42.4% at $\varepsilon = 0.2$). In five defenses, PGD adversarial training and TRADES provide the greatest robustness, with 55.3% and 59.0% detection at $\varepsilon = 0.2$ on a clean-accuracy cost of 1.0 and 3.2 points, respectively, and Gaussian augmentation and a feature-squeezing detector provide only partial protection. At a poisoning rate of 20%, the k-nearest-neighbor sanitizer successfully removes more than 97% of the flipped labels, and restores the vast majority of the lost detection rate for the random forest. We synthesize the results into guidance for building strong, smart cyber defense pipelines.

Keywords; Adversarial Machine Learning, Adversarial Training, Data Poisoning, Deep Learning, Evasion Attacks, Intrusion Detection, Network Security, Robustness, TRADES.

1. Introduction

Today, machine learning (ML) and especially deep learning (DL) is the foundation for a variety of intelligent cybersecurity applications, such as network intrusion detection, malware classification, spam and phishing filtering, and fraud detection. Learned detectors generalise to new variants of attacks, and are scalable to the amount of network telemetry in modern networks, as compared to signature-based tools [1,2]. These benefits, however, are based on an assumption that is rarely true in security: that training and test data come from a benign, stationary distribution. Cybersecurity is inherently adversarial. Attackers adapt to the defender, and a learned detector therefore becomes a new attack surface in its own right [3,4].

Adversarial examples demonstrated that high accuracy neural networks can fail under carefully designed, imperceptible changes [5,6] and such vulnerabilities existed before deep learning in security fields like spam and PDF malware detection [7,8]. Gradual improvement has been made in the attacks [9–12], and many of the defences proposed used gradient masking, which can be defeated by adaptive attacks [13,14], and it was shown that black-box attacks could be successful with the training of substitute models [15]. In the cybersecurity arena, attackers have been shown to successfully attack Android and Windows-based malware detectors [16-18] and ML-based NIDS [19-21].

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Despite this body of work, three gaps limit its practical value for security engineers. First, many NIDS studies move computer-vision attacks without enforcing constraints on which perturbations can be performed by an attacker, and the success rate reported may overestimate the actual threat [1, 21]. Second, often attacks and defenses are tested separately, using a single random seed and without reporting the clean-accuracy and computational cost of each defense. Third, there is a lack of understanding of training-time (poison) and test-time (evasion) threats in the same detector when both are realistic for a retrained system on collected traffic.

This paper tries to fill the above gaps by systematically and reproducibly studying the AML attacks and defenses in a DL-based NIDS. Our contributions are:

- A unified threat model and taxonomy for AML in intelligent cybersecurity systems that distinguishes attacker goal, knowledge, capability and timing, and that separates unconstrained feature-space perturbations from domain-constrained perturbations over 14 attacker-controllable traffic features.
- A multi-attack evaluation of a DNN-based NIDS on NSL-KDD covering FGSM, PGD-20 and C&W-L2 white-box evasion, black-box transfer through a query-trained substitute, and targeted label-flipping poisoning at 5–40%, reported as mean ± standard deviation over three seeds.
- A head-to-head comparison of five defenses (Gaussian data augmentation, FGSM and PGD adversarial training, TRADES, and a feature-squeezing detector) plus k-NN training-set sanitization, quantifying the trade-off between robustness, clean accuracy, false-positive rate and training cost.
- Analysis of which attack families and traffic features are most susceptible, and a set of evidence-based guidelines for deploying robust ML-based cyber defenses.

The remainder of the paper is organized as follows. Section II reviews related work. In section III, the threat model and taxonomy are defined. Section IV describes the methodology. Section V reports the results, Section VI discusses implications and limitations, and Section VII concludes.

2. Background And Related Work

2.1. Foundations Of Adversarial Examples

The first demonstration of the discontinuity of deep networks with respect to small perturbations of the input and the transferability of adversarial examples between models was given by Szegedy et al. [5]. This is explained by Goodfellow et al. [6] for high dimensional spaces, as locally linear behavior and they proposed a single-step fast gradient sign method (FGSM). Papernot et al. have introduced the Jacobian-based saliency map attack (JSMA) for L0-sparse perturbations [12] and Moosavi-Dezfooli et al. have presented DeepFool that estimates minimal perturbations [11]. Carlini and Wagner [9] introduced optimization-based attacks that succeeded against defensive distillation [22] under L0, L2 and L∞ norms. In [10] Madry et al. formulated the problem as a min–max, and proposed projected gradient descent (PGD) as a reference first-order attack. Biggio and Roli [3] traced the field back to earlier work on evasion [7] and poisoning of support vector machines [8] and the NIST taxonomy [4] now has common terminology for evasion, poisoning, privacy and misuse attacks.

2.2. Adversarial Attacks in the Cybersecurity Domain

Security data is different from images in a way that counts for attacks. Features are either categorical, continuous or discrete, semantically coupled and limited by protocol and payload functionality. To tackle the issue of Android malware, Gross et al. [16] modified JSMA to include only additions to features. Demetrio et al. [17] looked at functionality-preserving manipulations of Windows PE files and Pierazzi et al. [18] formalized problem-space attacks, demonstrating that feature-space perturbations have to be realizable as valid objects. IDSGAN [19] is a GAN that generates adversarial traffic records for NIDS, and Alhajjar et al. [20] employed particle swarm optimization, genetic algorithms and GANs to evade eleven classifiers on NSL-KDD and UNSW-NB15; however, many of the published NIDS threat models are unrealistic, according to Apruzzese et al. [21]. This literature is summarized by surveys of Rosenberg et al. [2] and Pitropakis et al. [23] and He et al. [1] and shows that very few of the studies that consider both constrained attacks and defenses have been carried out.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Beyond evasion, the training pipeline is also exposed. In poisoning attacks, data samples are modified to poison a model or mislabelled to introduce a backdoor [8, 24]. The former privacy attacks, including model extraction via prediction APIs [25] and membership inference [26], allow an adversary to replicate a detector, and the latter transfer attacks examined here [15] utilize this replication capability.

2.3. Defense Mechanisms

Defenses fall into four families. (i) Adversarial training augments training with adversarial examples: FGSM-based training [6], PGD-based training [10], ensemble adversarial training with transferred perturbations [27], and TRADES, which regularizes the boundary error with a KL-divergence term [28]. (ii) Detection and input transformation, e.g., MagNet [29] and feature squeezing [30] is designed to identify and correct problematic inputs. (iii) Certified defenses like randomized smoothing [31] give provable guarantees in a norm ball. (iv) Gradient-smoothing approaches such as defensive distillation [22] were later broken [9], and Athalye et al. [13] showed that seven of nine ICLR 2018 defenses relied on obfuscated gradients. Since then, AutoAttack [14] is considered standard for reliable robustness evaluation. A typical baseline against poisoning is training-set sanitization, which removes points that disagree with their neighbourhood [4, 23].

2.4. Research Gap

Table 1 compares representative studies along the dimensions examined in this paper. Most NIDS studies test attacks with no defense or one defense against one attack, and few report domain-constrained perturbations, poisoning, and variance among multiple seeds and cost of defense. The present study covers all of these dimensions on a common detector.

Table 1: Comparison of Representative AML Studies in Cybersecurity

Study	Domain	Evasion	Constr.	Poison.	Defenses	Multi-seed
[16]	Malware	JSMA	✓	✗	Distill., AT	✗
[19]	NIDS	GAN	✓	✗	✗	✗
[20]	NIDS	PSO/GA/GAN	Partial	✗	✗	✗
[21]	NIDS	Survey/model	✓	✗	✗	—
[18]	Malware	Problem-space	✓	✗	AT	✗
[17]	Malware	Byte-level	✓	✗	✗	✗
[1]	NIDS	Survey	—	—	Survey	—
This work	NIDS	FGSM/PGD/C&W/transfer	✓	✓	5 + sanit.	✓ (3)

3. Threat Model and Taxonomy

Based on the security-evaluation strategy of Biggio and Roli [3] and the taxonomy for Adversarial Machine Learning (AML) from NIST [4], an adversary can be characterized as having a goal, a knowledge, a capability and an attack timing. Fig. 1 places the attacks and defenses studied here in this taxonomy.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

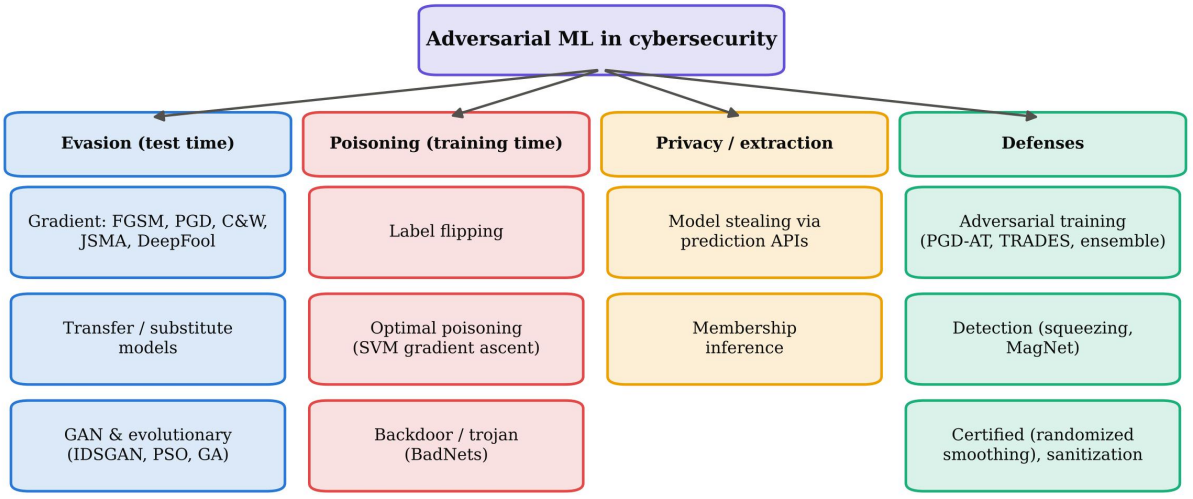

Fig. 1. Taxonomy of adversarial machine learning threats and defenses for intelligent cybersecurity systems. Shaded nodes are evaluated experimentally in this paper.

3.1. Attacker Goal

An integrity violation: If the attacker wants malicious flows to be classified as normal (a false negative) and wants to banish alerts on benign traffic. This is the most critical failure for a NIDS: silently allowing intrusions through. In poisoning, the aim is to decrease attack detection rate of the detector for future traffic by corrupting the labels of collected training data.

3.2. Attacker Knowledge

In the white-box setting the attacker knows the architecture, parameters and preprocessing of the detector, which yields a worst-case bound. In the black-box setting the attacker has only label-query access. We follow Papernot et al. [15] and Tramèr et al. [25] and select 30% of the training distribution to attack, train a smaller substitute network on the adversarial examples it returns, and transfer the adversarial examples it creates on the substitute network.

3.3. Attacker Capability and Domain Constraints

Let $x \in [0,1]^d$ be a normalized flow record. The attacker can include a perturbation δ with $\|\delta\|_\infty \leq \varepsilon$ and a binary mask $m \in \{0,1\}^d$ indicating features that can be altered, thereby causing $x' = \text{clip}(x + m \odot \delta, 0, 1)$. The one-hot protocol, service and flag fields which specify the attack are always fixed, whereas in the unconstrained setting m selects all 38 continuous features. In the constrained setting, motivated by [18,19, 21], m is limited to 14 volume, timing and host-statistic features that an attacker can control through padding, throttling or interleaving connections (duration, byte counts, connection counts and same/different service and host rates). Content changes like failed logins, root shells, and file creations are not included as this would change the intrusion. The threat scenarios are summarised in Table 2.

Table 2: Threat Scenarios Evaluated

Scenario	Phase	Knowledge	Capability	Method
S1	Test	White-box	38 feats, $L\infty \leq \varepsilon$	FGSM, PGD-20
S2	Test	White-box	14 feats, $L\infty \leq \varepsilon$	FGSM, PGD-20
S3	Test	White-box	38/14 feats, min L2	C&W-L2
S4	Test	Black-box (labels)	38 feats, $L\infty \leq \varepsilon$	Substitute transfer

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Scenario	Phase	Knowledge	Capability	Method
S5	Training	Data access	Flip 5–40% attack labels	Label-flip poisoning

4. Methadology

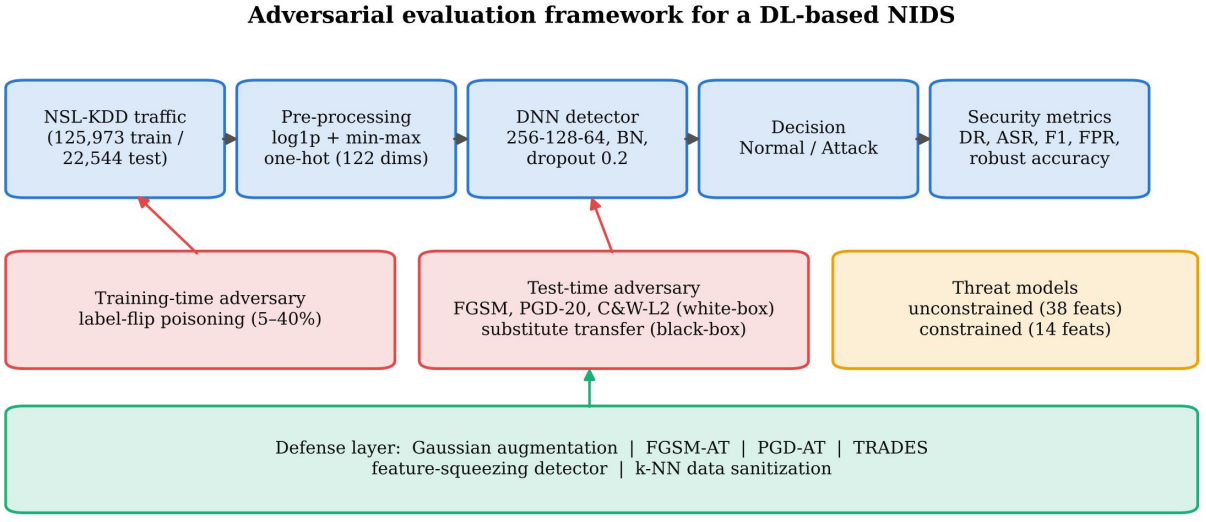

Fig. 2. Overview of the evaluation framework: data pipeline, DNN detector, training- and test-time adversaries, threat models, and the defense layer.

Fig. 2 summarizes the evaluation framework, which connects the data pipeline, the detector, the two adversaries and the defense layer.

4.1. Dataset and Preprocessing

We employ NSL-KDD [32] which is a clean version of KDD Cup 99 and is widely used as a standard test set in adversarial NIDS research [19,20]. Each record has 41 features: 38 numerical and three categorical (protocol type, service, flag). Attack labels are divided into four families: denial of service (DoS), probing (Probe), remote-to-local (R2L) and user-to-root (U2R). We use KDDTrain+ for training and KDDTest+ for testing, with 17 attack types that are not covered while training. This requires more effort than a random split and is more reflective of deployment. Table III gives the class distribution. Categorical fields are one-hot encoded (84 binary columns), twelve heavy-tailed count features are log-transformed with log(1+x), and all numerical features are min–max scaled to [0,1] using statistics from the training set only, which yields d = 122 input dimensions. We mention that there are newer datasets, e.g., UNSW-NB15 [33] and CIC-IDS2017 [34] and discuss generalization in Section VI.

Table 3: Class Distribution of Nsl-Kdd Splits

Class	KDDTrain+	%	KDDTest+	%
Normal	67,343	53.46	9,711	43.08
DoS	45,927	36.46	7,458	33.08
Probe	11,656	9.25	2,421	10.74
R2L	995	0.79	2,885	12.80
U2R	52	0.04	69	0.31

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Class	KDDTrain+	%	KDDTest+	%
Total	125,973	100.00	22,544	100.00

4.2. Target Detector

The detector is a fully connected DNN with three hidden layers of 256, 128 and 64 units. Each layer uses batch normalization, ReLU activation and dropout (p = 0.2), followed by a two-way softmax output. It is trained with Adam (learning rate 10^{-3}, weight decay 10^{-5} , batch size 256) for 12 epochs with cross-entropy loss. As a reference, we train a random forest (RF, 200 trees), logistic regression (LR) and k-nearest neighbors (kNN, k = 5).

4.3. Attack Formulations

Let f_θ be the detector, L the cross-entropy loss function and y = 1 the attack label. FGSM [6] only steps in the direction of the sign of the gradient:

$$x' = \mathrm{clip}_{[0,1]}(x + \varepsilon\, m \odot \mathrm{sign}(\nabla_x \mathcal{L}(f_\theta(x), y))) \tag{1}$$

PGD [10] iterates (2) for T = 20 steps with step size α = 2.5ε/T from a random start inside the ε-ball, projecting onto the ball Π after each step:

$$x^{(t+1)} = \Pi_{\mathcal{B}_\infty(x,\varepsilon)}\left(x^{(t)} + \alpha\, m \odot \mathrm{sign}\left(\nabla_x \mathcal{L}(f_\theta(x^{(t)}), y)\right)\right) \tag{2}$$

The C&W-L2 attack [9] looks for a minimum-norm evasion under the following change of variables and minimizes it x′ = ½(tanh(w)+1) with the restriction of using only masked features.

$$\min_w \ \|x' - x\|_2^2 + c \cdot \max\left(Z(x')_y - \max_{j \neq y} Z(x')_j + \kappa,\ 0\right) \tag{3}$$

Where Z denotes the logits, κ = 0, and c ∈ {1, 10}. This is run with 200 iterations of Adam with learning rate 0.01 on 1500 attack flows correctly identified by the clean model. The substitute network (128-64 hidden units) is trained with 37,791 query–label pairs, and FGSM or PGD examples constructed on this network are replayed against the target DNN, RF and LR. In targeted label-flip poisoning, prior to training, a fraction of ρ = {5, 10, 20, 30, 40}% of the attack-labeled training records are relabeled as normal.

Table 4 compares the attack algorithm based on the knowledge required, the norm constrained and the cost per flow in our implementation. Single-step FGSM is the least expensive but also the weakest, while PGD takes 20 forward–backward passes and C&W takes 200 optimizer steps and yields a near-minimal perturbation with no hard budget limitation. The transfer attack does not require gradient access to the target but a query budget to train the substitute, requiring a connection between evasion and the model-extraction threat [25].

Table 4: Characteristics of the Evaluated Attack Algorithms

Attack	Knowledge	Norm	Iterations	Target access
FGSM [6]	White-box	L∞ (budget ε)	1	Gradients
PGD-20 [10]	White-box	L∞ (budget ε)	20 + rand. start	Gradients
C&W-L2 [9]	White-box	L2 (minimized)	200 (Adam)	Logits
Transfer [15]	Black-box	L∞ (budget ε)	1 / 20 on substitute	Labels (37,791 queries)
Label flip	Data access	—	—	Training data

4.4. Defense Formulations

The saddle-point problem of Madry et al. [10] is addressed by adversarial training (AT):

$$\min_\theta \ \mathbb{E}_{(x,y)}\left[\tfrac{1}{2}\mathcal{L}(f_\theta(x), y) + \tfrac{1}{2}\max_{\|\delta\|_\infty \le \varepsilon} \mathcal{L}(f_\theta(x + m \odot \delta), y)\right] \tag{4}$$

where the inner maximization is performed by FGSM (FGSM-AT) or 7-step PGD (PGD-AT) using ε = 0.1, and clean and adversarial losses are combined equally so as to maintain clean accuracy. TRADES [28] instead minimizes

$$\min_\theta \ \mathbb{E}\left[\mathcal{L}(f_\theta(x), y) + \beta \max_{x' \in \mathcal{B}(x,\varepsilon)} \mathrm{KL}\left(f_\theta(x) \,\|\, f_\theta(x')\right)\right] \tag{5}$$

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

With $\beta = 6$ and the same inner budget. Gaussian augmentation adds $N(0, 0.1^2)$ noise to modifiable features during training, which is the base-classifier training step of randomized smoothing [31]. The feature-squeezing detector [30] compresses each feature to 3 bits, and returns a flag if the L1 distance between the softmax outputs of the original and squeezed features is too large, defined as the feature having a 5% false-positive rate on clean training features. To resist poisoning, k-NN sanitization trains an 11-nearest-neighbour classifier on a 40,000-sample subset of the (possibly poisoned) training set, and removes all training records whose label differs from the majority label of their neighbourhood.

4.5. Evaluation Metrics

Accuracy, precision, recall (attack detection rate, DR), F1, false-positive rate (FPR) and ROC-AUC are reported for the entire test set. When attacked, the Strong DR is the proportion of all 12,833 attack flows that are still detected when the flow is perturbed. Because about one third of KDDTest+ attacks are already missed by the clean model, we also report the conditional attack success rate,

$$\mathrm{ASR} = \frac{|\{x : f(x) = 1 \wedge f(x') = 0\}|}{|\{x : f(x) = 1\}|} \quad (6)$$

Which counts only evasions the attacker actually caused. In the case of transfer attacks, ASR is calculated independently per target model with respect to the flows that the model detected prior to the attack.

4.6. Experimental Setup

All experiments are conducted with PyTorch 2 and scikit-learn 1.9 on a CPU-only computer (2 vCPUs, 8 GB RAM), and the results are repeated for three different seeds (0, 1, and 2). Table 5 lists the main hyperparameters. The perturbation budget ε is swept over {0.01, 0.02, 0.05, 0.10, 0.15, 0.20} in normalized feature units.

Table 5: Experimental Hyperparameters

Component	Setting
DNN	122-256-128-64-2, BN, ReLU, dropout 0.2
Optimizer	Adam, lr 1e-3, wd 1e-5, batch 256, 12 epochs
FGSM / PGD	$\varepsilon \in [0.01, 0.2]$; PGD: 20 steps, $\alpha = 2.5\varepsilon/20$, rand. start
C&W-L2	$c \in \{1, 10\}$, $\kappa = 0$, 200 Adam steps, lr 0.01
Substitute	122-128-64-2, 30% query budget, 10 epochs
FGSM-AT / PGD-AT	$\varepsilon = 0.1$, PGD-7 inner loop, 50/50 loss mix
TRADES	$\varepsilon = 0.1$, $\beta = 6$, 7 inner KL-PGD steps
Gaussian aug.	$\sigma = 0.1$ on modifiable features
Feature squeezing	3-bit depth, threshold at 5% clean FPR
Sanitization	k = 11, 40k reference subset
Seeds	0, 1, 2 (mean ± std reported)

5. Results and Analysis

5.1. Baseline Detection Performance

Table 6: Clean Performance on Kddtest+ (%, Mean ± Std Over 3 Seeds)

Model	Accuracy	Precision	Recall (DR)	F1	FPR	AUC
DNN (target)	79.0 ± 1.0	93.5 ± 2.2	67.9 ± 1.3	78.6 ± 1.0	6.2 ± 2.2	90.7 ± 1.5
Random forest	77.3 ± 0.4	96.9 ± 0.1	62.1 ± 0.6	75.7 ± 0.5	2.6 ± 0.0	96.1 ± 0.1
Logistic regression	74.3 ± 0.0	91.2 ± 0.0	60.7 ± 0.0	72.9 ± 0.0	7.7 ± 0.0	77.4 ± 0.0
kNN (k = 5)†	77.0	92.2	65.0	76.3	7.2	82.2

† kNN evaluated with seed 0 only (deterministic).

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Table 6 shows that the DNN is the strongest clean detector, with 79.0 ± 1.0% accuracy, 78.6 ± 1.0% F1 and 90.7 ± 1.5% AUC at an FPR of 6.2 ± 2.2%. The RF has a higher AUC (96.1%) and a lower FPR (2.6%), but its recall is lower (62.1%) which means that it misses more intrusions. The absolute recall values are typical for KDDTest+, whose novel attack types (mostly R2L and U2R) are hard to detect from the training distribution [32]. The reference point for all robustness results below are these figures.

5.2. White-Box Evasion Attacks

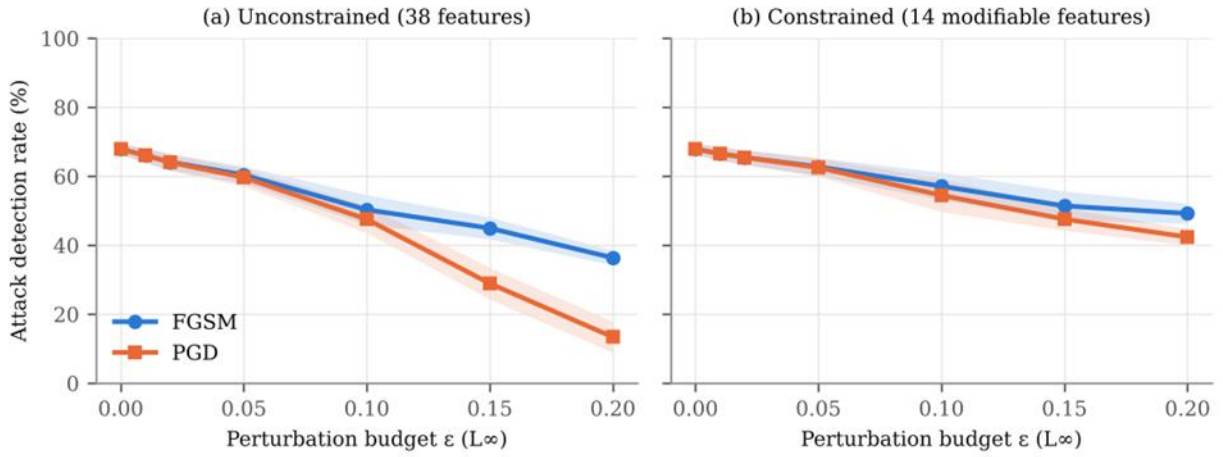

Fig. 3. Attack detection rate of the undefended DNN under FGSM and PGD-20 as a function of the L∞ budget ε. Shaded bands show ±1 std over three seeds. (a) Unconstrained attacker; (b) attacker restricted to 14 modifiable features.

Table 7: Conditional Attack Success Rate (%) of White-Box Attacks on the Undefended Dnn

ε	FGSM (unc.)	PGD (unc.)	FGSM (con.)	PGD (con.)
0.01	2.7 ± 0.7	2.7 ± 0.6	1.9 ± 0.6	1.9 ± 0.6
0.02	5.5 ± 1.5	5.6 ± 1.5	3.6 ± 0.7	3.6 ± 0.7
0.05	11.0 ± 1.6	12.1 ± 1.7	7.6 ± 1.8	7.9 ± 1.8
0.10	25.9 ± 4.5	30.0 ± 4.3	15.9 ± 3.7	19.8 ± 4.4
0.15	33.8 ± 3.6	57.3 ± 6.0	24.3 ± 4.2	29.9 ± 3.5
0.20	46.5 ± 2.5	80.3 ± 5.1	27.6 ± 3.6	37.5 ± 3.4

Fig. 3 and Table 7 show that the DNN is highly sensitive to small, targeted perturbations. With unconstrained access, PGD-20 raises the conditional ASR from 12.1% at ε = 0.05 to 30.0% at ε = 0.1 and 80.3% at ε = 0.2, which drives the overall detection rate down from 67.9% to 13.4%. For large budgets, FGSM is no longer as effective (46.5% ASR at ε = 0.2), which is enough evidence to demonstrate that iterative attacks are required for a reliable evaluation [10, 14]. With 14 controllable features, the attacker can achieve a success rate of 37.5% at ε = 0.2, which is a decrease of 42.8 percentage points. Despite this, about 38% of intrusions still can be concealed with the use of volume, timing and host statistics manipulation only. The gap between the two settings quantifies how much unconstrained evaluations overstate the threat [21] while showing that realistic attackers remain dangerous.

5.3. Minimum-Norm C&W Attack

Table 8: C&W-L2 Results on 1,500 Correctly Detected Attack Flows

Threat model	c	ASR (%)	Median L2	Mean L∞	Time (s)
Unconstrained (38)	1	75.3 ± 2.0	0.803 ± 0.070	0.432 ± 0.030	2.0 ± 0.0
Unconstrained (38)	10	75.9 ± 1.8	0.871 ± 0.060	0.435 ± 0.033	1.9 ± 0.1

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Threat model	c	ASR (%)	Median L2	Mean L∞	Time (s)
Constrained (14)	1	70.2 ± 0.8	0.769 ± 0.059	0.436 ± 0.033	1.8 ± 0.0
Constrained (14)	10	70.8 ± 0.2	0.834 ± 0.054	0.439 ± 0.038	1.8 ± 0.1

The optimization-based C&W-L2 attack is reported in Table VIII, which seeks to find the smallest perturbation to cause evasion. In the unconstrained situation, C&W evades 75.9 ± 1.8% of attack flows correctly identified, while in the constrained situation, it evades 70.8 ± 0.2%. The median L2 norm of the successful perturbations is 0.87, but the distribution is not uniform as the mean L∞ norm is 0.44 – this means that C&W focuses large changes on a small number of influential features, rather than making small changes to all features. Increasing c from 1 to 10 doesn't increase the ASR by more than one point, which suggests that the remaining flows cannot be avoided within a feasible box, not that the optimizer is stalled. The attack is slow (a few seconds per 1,500 flows on a CPU) which poses no obstacle for an opportunistic adversary.

5.4. Vulnerability by Attack Family and Feature

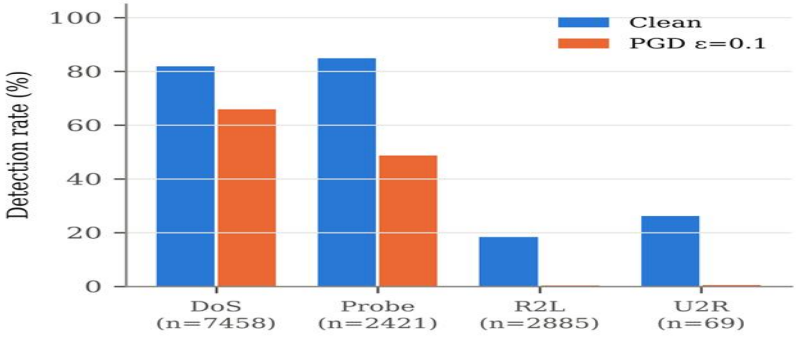

Fig. 4. Detection rate per attack family on clean traffic and under unconstrained PGD (ε = 0.1), mean of three seeds.

Fig. 4 breaks robustness down by attack family. Volumetric families are better detected on clean data (DoS 81.9%, Probe 84.9%) with a drop of 16.1 and 36.2 points at ε = 0.1 under PGD. Difficult to detect R2L and U2R flows (18.2% and 26.1%) are nearly completely undetectable (0.3% and 0.5%). Low and slow host level intrusions are thus the closest to the benign manifold and the easiest to disguise.

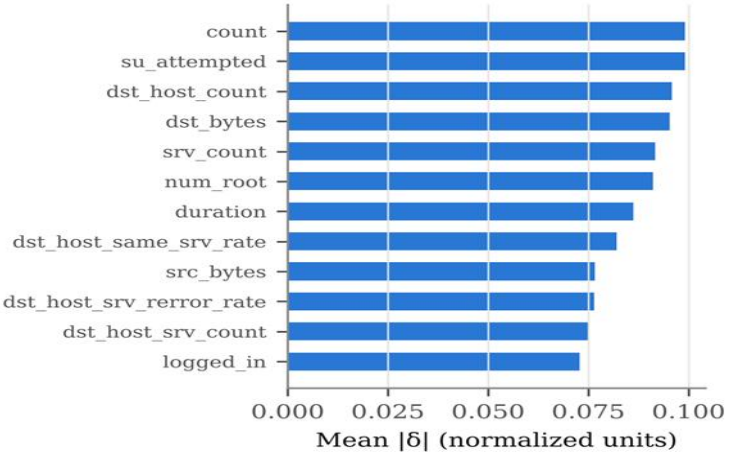

Fig. 5. Twelve numerical features with the largest mean absolute PGD perturbation (ε = 0.1, unconstrained, seed 0).

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

Fig. 5 ranks the features the PGD attacker moves most: count, su_attempted, dst_host_count, dst_bytes, srv_count. The most dominant statistics are connection-count, and some of them are among the 14 features available to the restricted attacker. This is why domain constraints only partially reduce attack success. Fig. 6 projects clean and adversarial attack flows onto the first two principal components of the clean data. Adversarial flows do not necessarily head to a clearly-defined area away from the clean attack flows and is the reason for the poor performance of distribution-based detectors (Section V-G).

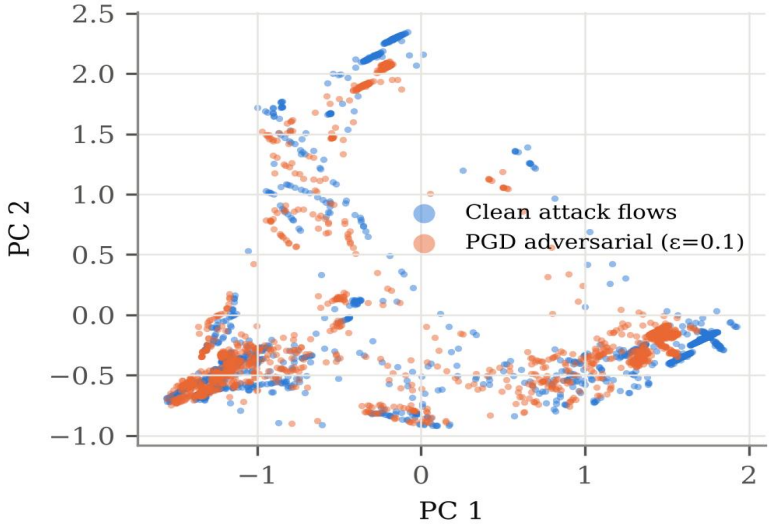

Fig. 6. PCA projection of 1,500 clean attack flows and their PGD adversarial counterparts (ε = 0.1).

5.5. Black-Box Transferability

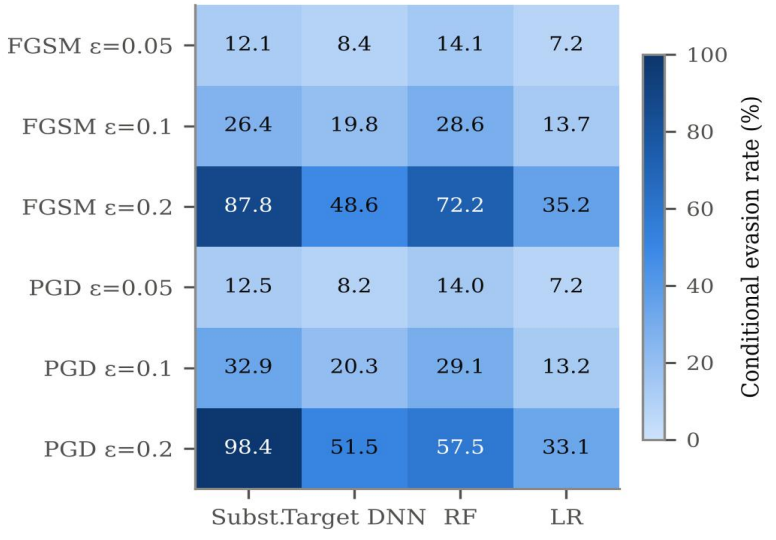

Fig. 7. Conditional evasion rate (%) of adversarial flows crafted on the black-box substitute and transferred to each target model (mean of three seeds).

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

As shown in Fig. 7, a label-only attacker can cause significant evasion even if they know nothing about the internals of the target. The target DNN stops detecting the substituted ones in 51.5 ± 5.4% of the flows it previously detected, the RF 57.5 ± 6.6% and LR 33.1 ± 4.8% of the flows. Transfer rates monotonically increase with increasing ε and they even decrease in the absence of any gradients (the RF), making clear that the use of a non-differentiable detector alone is not a protection [15]. For small budgets (ε = 0.05), the transfer is limited (8.2% on the DNN), and the black-box attacker has to pay a higher cost for the lack of knowledge in the transfer with a bigger perturbation.

5.6. Effectiveness of Defenses Against Evasion

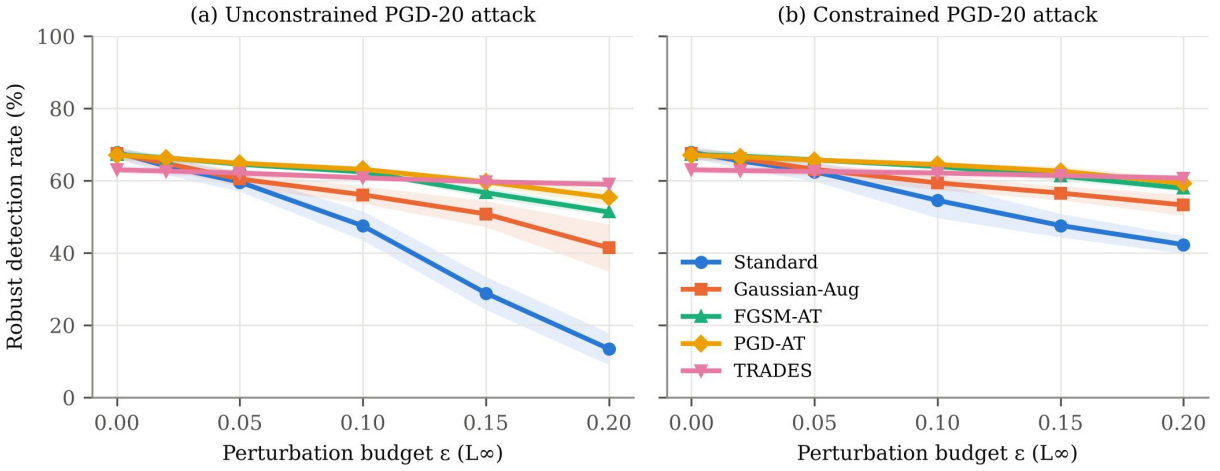

Fig. 8. Robust detection rate under PGD-20 for the five training strategies. (a) Unconstrained attacker; (b) constrained attacker. Shaded bands show ±1 std over three seeds.

Table 9: Defense Comparison: Clean Performance, Robust Detection Rate (%) And Training Cost

Defense	Clean acc.	Clean FPR	FGSM ε=.1	PGD ε=.1	PGD ε=.2	PGD ε=.2 (con.)	Train time
Standard	79.0 ± 1.0	6.2 ± 2.2	50.3 ± 3.5	47.5 ± 3.3	13.5 ± 3.6	42.3 ± 2.0	1.0×
Gaussian-Aug	78.9 ± 0.6	6.1 ± 1.5	57.1 ± 2.0	56.1 ± 2.0	41.4 ± 5.3	53.3 ± 2.4	1.0×
FGSM-AT	78.2 ± 0.6	7.4 ± 0.1	62.9 ± 0.1	62.4 ± 0.1	51.3 ± 0.9	57.9 ± 0.6	2.0×
PGD-AT	78.1 ± 0.4	7.5 ± 0.1	63.4 ± 0.5	63.1 ± 0.5	55.3 ± 1.8	59.3 ± 1.1	4.1×
TRADES	75.8 ± 0.5	7.4 ± 0.0	61.0 ± 0.6	60.8 ± 0.5	59.0 ± 0.5	60.7 ± 0.7	4.2×

Fig. 8 and Table 9 compare the defenses. The undefended model collapses to 13.5% robust DR at ε = 0.2. Gaussian augmentation is superior for large budgets (41.4% when ε = 0.2), but clearly not as good as adversarial training and is quickly dominated by it as ε increases as random noise does not cover for the worst case directions found by PGD. In all three adversarial-training schemes, the robustness curve is levelled off. PGD-AT retains 55.3 ± 1.8% DR at ε = 0.2, TRADES 59.0 ± 0.5% and FGSM-AT 51.3 ± 0.9%, which is 4.1× the undefended DR for PGD-AT. Robustness also extends beyond the training budget (ε = 0.1) to ε = 0.2, and it remains in the constrained setting (PGD-AT and TRADES are around 8 and 3 points away from their clean DR, respectively, at ε = 0.2).

Robustness is not free. PGD-AT reduces the accuracy of clean data by 1.0-point and TRADES by 3.2-point, and the training time is increased by 4.1× and 4.2×. The robustness–accuracy trade-off prediction of Zhang et al. [28]. Fig. 9 plots the trade-off. The best balance is provided by PGD-AT and TRADES provides a flatter robustness curve but with lower clean accuracy at β = 6. Adversarial training raises the FPR only slightly, from 6.2% to at most 7.5%, and it makes the FPR far more stable across seeds (std ≤ 0.1 versus 2.2 points), so the alert load on analysts stays manageable.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

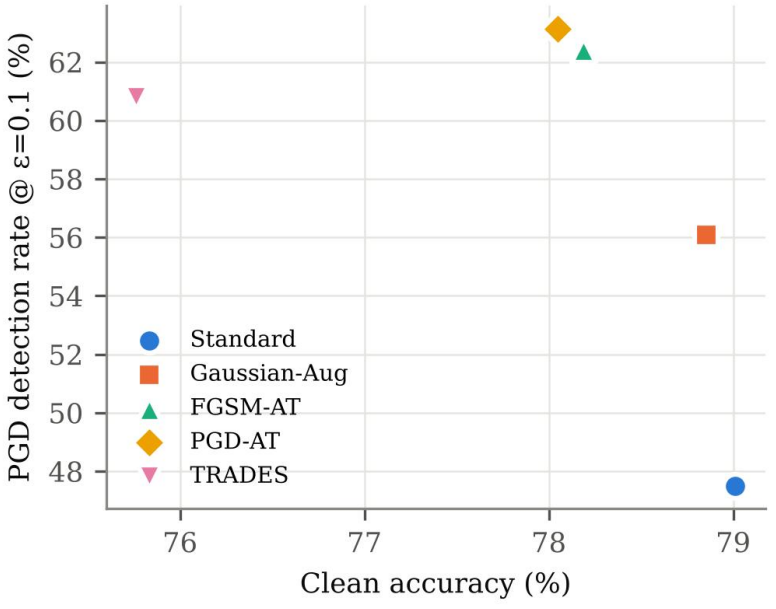

Fig. 9. Robustness–accuracy trade-off of the five training strategies (unconstrained PGD, ε = 0.1).

5.7. Detection-Based Defense: Feature Squeezing

Table 10: Feature-Squeezing Detector Against Pgd (Unconstrained)

ε	Adv. flagged (%)	Clean FPR (%)	Combined DR (%)
0.02	48.3 ± 7.1	7.6 ± 1.9	81.3 ± 3.5
0.05	38.5 ± 4.1	7.6 ± 1.9	75.1 ± 2.9
0.10	22.8 ± 3.5	7.6 ± 1.9	59.5 ± 2.5
0.20	41.5 ± 1.8	7.6 ± 1.9	49.3 ± 3.5

The feature-squeezing detector [30] over the undefended DNN is evaluated in Table 10, where the threshold is set to 5% clean FPR. It only flags successfully adversarial flows in 22.8–48.3% of cases, and its test-set FPR on benign traffic is 7.6%. The combined DR (an attack counts as caught if the classifier detects it or the squeezer flags it) is 59.5% at ε = 0.1 and 49.3% at ε = 0.2, below the 63.1% and 55.3% achieved by PGD-AT without any detector. With a small ε, the combined DR is greater than the clean DR due to the fact that the squeezer also marks some flows that the classifier does not mark, but it incurs additional false alarms. Bit-depth reduction was designed for images, where adjacent pixel values are redundant. Network features have no such redundancy, so squeezing changes benign and adversarial predictions alike. This result is consistent with evidence that input-transformation detectors give weak guarantees and are vulnerable to adaptive attacks [13, 29].

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

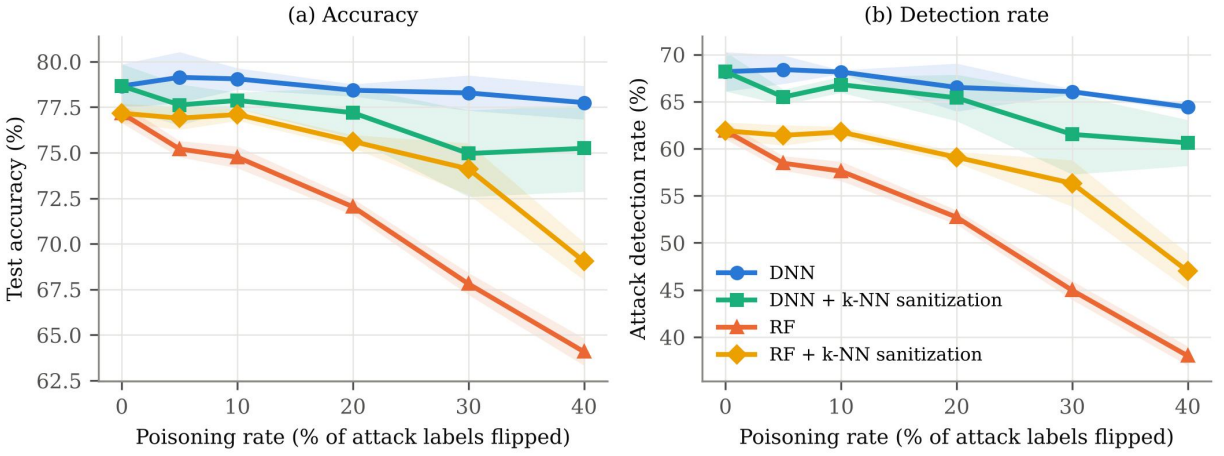

Fig. 10. Effect of targeted label-flip poisoning on (a) test accuracy and (b) attack detection rate for the DNN and RF, with and without k-NN sanitization. Shaded bands show ±1 std over three seeds.

Table 11: Detection Rate (%) Under Label-Flip Poisoning and Sanitization Effectiveness

ρ (%)	DNN	DNN + san.	RF	RF + san.	Poison removed	Clean removed
0	68.2 ± 1.7	68.2 ± 1.7	61.9 ± 0.7	61.9 ± 0.7	—	—
5	68.4 ± 1.2	65.5 ± 0.7	58.5 ± 0.6	61.4 ± 0.9	99.3 ± 0.1	0.4 ± 0.0
10	68.1 ± 0.2	66.8 ± 0.7	57.6 ± 0.8	61.8 ± 0.5	99.2 ± 0.1	0.4 ± 0.0
20	66.5 ± 2.1	65.4 ± 2.0	52.7 ± 0.6	59.1 ± 0.5	97.4 ± 0.2	0.8 ± 0.0
30	66.0 ± 0.2	61.5 ± 3.5	45.0 ± 0.8	56.3 ± 2.0	88.5 ± 0.9	3.1 ± 0.3
40	64.4 ± 0.4	60.6 ± 2.0	38.0 ± 0.8	47.0 ± 1.5	70.7 ± 1.0	7.9 ± 0.4

Fig. 10 and Table 11 report the poisoning results. The RF degrades steadily as more attack labels are flipped: its DR falls from 61.9% to 38.0% at ρ = 40%. The DNN is markedly more tolerant (DR 68.2% to 64.4%), because the large number of redundant DoS and Probe records lets gradient training average out randomly placed noisy labels. k-NN sanitization removes 99.3% of poisoned records at ρ = 5% and 97.4% at ρ = 20%, while discarding fewer than 0.8% of clean records. It is only as precise as poisoning points become local majorities (70.7% removed at ρ = 40%). Up to ρ = 20%, sanitization recovers most of the RF's lost detection rate (52.7% to 59.1%, against 61.9% without poisoning), and it still adds 9.0 points at ρ = 40%. For the already-tolerant DNN, removing hard, boundary-near clean examples doesn't save anything, as it comes at the same price as removing noise. The best defence against poisoning is, then, the learner and sanitisation is most useful for learning high-capacity, memorizing models like tree ensembles.

6. Discussion

6.1. Key Findings

- Accuracy is not security. The most accurate clean detector (DNN, 79.0%) loses most of its detection capability under a white-box PGD attack, so robustness must be measured explicitly rather than inferred from clean metrics.
- Domain constraints matter but do not remove the threat. Restricting the attacker to 14 controllable features lowers PGD success by 42.8 points at ε = 0.2, but 37.5% of detected attacks are still evaded.
- Iterative adversarial training is the most effective evasion defense evaluated here. PGD-AT and TRADES generalize to larger budgets and to the constrained setting at a modest clean-accuracy cost, whereas noise augmentation and feature squeezing give only partial protection and degrade quickly as the budget grows.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

- Black-box transfer works across model families, including non-differentiable RFs. Model confidentiality and obfuscation are therefore not substitutes for robust training.
- Poisoning robustness depends on the learner. Tree ensembles benefit greatly from neighbourhood sanitization, whereas DNNs are intrinsically more tolerant of random label noise.

6.2. Guidelines for Robust Intelligent Cybersecurity Systems

These results lead to five practical recommendations. (1) Evaluate detectors with iterative, adaptive attacks (PGD, C&W, AutoAttack [14]) under an explicit, domain-constrained threat model, and report conditional ASR alongside clean metrics. (2) As the baseline hardening method for DNN detectors, use PGD-based adversarial training, or TRADES if a flatter robustness curve is desired, and allocate approximately 4× the training compute. (3) Implement sanitization and provenance measures in retraining pipelines that consume collected traffic, particularly for trees. (4) Restrict access to query and/or restrict output granularity to slow down substitute training and model extraction [25,26]. (5) Combine learned detectors with rule-based and protocol-level checks so that features an attacker can freely manipulate are cross-validated against invariant evidence.

6.3. Implications for Security Operations

Intelligent cybersecurity systems are rarely static. Retraining of detectors on newly labelled traffic; scoring and passing traffic into security information and event management (SIEM) pipelines; and analysts triage the alerts. From our results, we can see that there are adversarial exposures in each of these stages. The collected traffic can be retrained for poisoning, and the poisoning experiments demonstrate several-fold difference depending on the type of learner. The detector can be queried to create a synthetic counterpart, and the transfer experiments reveal that these synthetic detectors can generate adversarial flows for independently trained models. Thus, adversarial robustness is an ongoing property that should be managed by MLOps tools and practices, including data provenance, data sanitization, robust retraining and red teaming, not as a standalone model property. The relatively low cost of PGD adversarial training (less than two minutes on a CPU in our setting) makes it feasible to incorporate large training in regular retraining cycles.

6.4. Limitations and Threats to Validity

This study has several limitations. NSL-KDD is an outdated detection method, and its flow features do not include modern applications or encrypted traffic. The trends shall be confirmed on the following models: UNSW-NB15 [33], CIC-IDS2017 [34] and raw-packet models. Our constrained threat model is a feature-space approximation. It does not assure that any perturbed vector is a valid sequence of packets, which would entail a problem-space construction [18, 21]. Adversarial training was tested with the same attack family as during training. The reported robust DR might be decreased by stronger adaptive and ensemble attacks [13,14] and certified defenses [31] were not tested. Lastly, three seeds cover variance of training but not variance across different data collections.

7. Conclusion and Future Work

This paper presented a unified evaluation of adversarial attacks and defenses for an intelligent, DL-based network intrusion detection system. We demonstrated that the majority of the detected attacks can be avoided by the white-box threat model, that black-box substitute attacks can be transferred across model families and that label-flip poisoning can significantly reduce the performance of the Tree-based detectors. Among the defenses, PGD adversarial training and TRADES gave the strongest and most consistent robustness, retaining about 55% detection at $\varepsilon = 0.2$ compared with 13% for the undefended model, at a cost of a few points of clean accuracy and roughly four times the training effort. k-NN sanitization removed more than 97% of poisoned labels at poisoning rates up to 20% and restored most of the random forest's lost detection rate. Future research will expand to problem-space attack on raw traffic, multi-class and streaming detectors, certified robustness for tabular security data, and emerging attacks on LLM-based security agents listed in the NIST AML taxonomy [4].

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

8. Data and Code Availability

The NSL-KDD dataset is publicly available from the Canadian Institute for Cybersecurity. The experiment code, per-seed results and the figure and table data are presented in an accompanying supplementary Excel workbook with this manuscript.

References

[1] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," IEEE Commun. Surveys Tuts., vol. 25, no. 1, pp. 538–566, 1st Quart. 2023.

[2] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," ACM Comput. Surv., vol. 54, no. 5, Art. no. 108, pp. 1–36, 2021.

[3] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," Pattern Recognit., vol. 84, pp. 317–331, Dec. 2018.

[4] A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies, and M. Hamin, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST AI 100-2e2025, Mar. 2025.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in Proc. Int. Conf. Learn. Represent. (ICLR), Banff, AB, Canada, 2014.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in Proc. Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, 2015.

[7] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in Proc. Eur. Conf. Mach. Learn. Knowl. Discov. Databases (ECML PKDD), ser. LNCS, vol. 8190, Prague, Czech Republic, 2013, pp. 387–402.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in Proc. 29th Int. Conf. Mach. Learn. (ICML), Edinburgh, U.K., 2012, pp. 1467–1474.

[9] U. Imtiaz, "Dynamic security certification framework for evolving distributed," in Data Science and Security: Proc. IDSCS 2025, vol. 1, 2026, p. 347.

[10] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in Proc. Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, 2018.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 2574–2582.

[12] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), Saarbrücken, Germany, 2016, pp. 372–387.

[13] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in Proc. 35th Int. Conf. Mach. Learn. (ICML), PMLR, vol. 80, Stockholm, Sweden, 2018, pp. 274–283.

[14] V. Gummadavelli, U. Imtiaz, P. A., S. H. Kolla, N. S. Bavirisetty, and S. R. Kondapalli, "Discovering and repairing blind spots in LLM-as-a-judge evaluation of multi-turn AI systems," in Proc. 2026 Int. Conf. Intell. Sustain. AI Syst. (ICOSAAS), Bali, Indonesia, 2026, pp. 1611–1618, doi: 10.1109/ICOSAAS68663.2026.11648543.

[15] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIA CCS), Abu Dhabi, UAE, 2017, pp. 506–519.

[16] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in Proc. Eur. Symp. Res. Comput. Secur. (ESORICS), ser. LNCS, vol. 10493, Oslo, Norway, 2017, pp. 62–79.

[17] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli, "Adversarial EXEmples: A survey and experimental evaluation of practical attacks on machine learning for Windows malware detection," ACM Trans. Privacy Secur., vol. 24, no. 4, Art. no. 27, pp. 1–31, Sep. 2021.

[18] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in Proc. IEEE Symp. Secur. Privacy (SP), San Francisco, CA, USA, 2020, pp. 1332–1349.

[19] A. Kumar, M. A. Siddiqui, S. Jafar, and K. K. S. Nair, "A data-driven predictive framework for reducing medication non-adherence cost in the US healthcare system," American Journal of Multidisciplinary Knowledge Insights, vol. 5, no. 2, pp. 60–75, 2026.

[20] F. Amin, "A cybersecurity-focused risk assessment framework for multi-party digital asset custody operations," in Expert Systems, Applications and Security, V. S. Rathore, P. T. Duc, T. C. Doan, and E. S. Pilli, Eds., NICE-TEAS ASIA 2025, Algorithms for Intelligent Systems. Cham, Switzerland: Springer, 2027, doi: 10.1007/978-3-032-26919-5_34.

[21] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," Digit. Threats: Res. Pract., vol. 3, no. 3, Art. no. 31, pp. 1–19, Sep. 2022.

[22] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in Proc. IEEE Symp. Secur. Privacy (SP), San Jose, CA, USA, 2016, pp. 582–597.

[23] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," Comput. Sci. Rev., vol. 34, Art. no. 100199, Nov. 2019.

[24] A. Kumar and K. K. S. Nair, "The Impact of Cloud-Based Business Analytics on Operational Efficiency in US Small and Medium-Sized Firms," Spanish Journal of Innovation and Integrity, vol. 49, pp. 303–318, 2025.

[25] M. N. Amjad, "Strategic Engineering Management for Sustainable and Resilient US Industrial Infrastructure," International Journal of Business Integrity and Technology Advancements, vol. 1, no. 2, pp. 8–20, 2025..

[26] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in Proc. IEEE Symp. Secur. Privacy (SP), San Jose, CA, USA, 2017, pp. 3–18.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0

[27] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in Proc. Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, 2018.
[28] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. El Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," in Proc. 36th Int. Conf. Mach. Learn. (ICML), PMLR, vol. 97, Long Beach, CA, USA, 2019, pp. 7472–7482.
[29] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), Dallas, TX, USA, 2017, pp. 135–147.
[30] F. Amin, "Sentinel Polity: LLM-grounded policy-as-code for IaC hardening in hyperscale clouds," in Data Science and Security, S. Shukla, H. Sayama, K. Tiwari, J. P. George, and J. V. Kureethara, Eds., IDSCS 2025, Lecture Notes in Networks and Systems, vol. 1947. Cham, Switzerland: Springer, 2026, doi: 10.1007/978-3-032-24360-7_30.
[31] M. A. Rahman, I. Das, M. R. Abir, B. M. T. Haque, A. Al Noman, A. Ahmed, and M. J. Hossen, "Pattern recognition tasks with personalized federated learning," Emerging Science Journal, vol. 10, no. 2, pp. 974–990, 2026, doi: 10.28991/ESJ-2026-010-02-020.
[32] M. Imran, "Explainable AI for cybersecurity risk assessment in US financial institutions," American Journal of Multidisciplinary Knowledge Insights, vol. 2, no. 1, pp. 11–28, 2023.
[33] M. N. Amjad, "Artificial Intelligence-Based Voltage Stability Assessment in Modern Power Systems: Edge Intelligence for Real-Time Monitoring of Smart Electrical Networks," Journal of Business Insight and Innovation, vol. 3, no. 1, pp. 102–115, 2024.
[34] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108–116.

© 2026 The Authors. This article is an Open Access article distributed under the terms and conditions of the Creative Commons Attribution(CC–BY)license
https://creativecommons.org/licenses/by/4.0