IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

23.03.2025

Digital Object Identifier

Multi-Feature Flow-Level Adversarial Attacks Against ML-Based Network Intrusion Detection Systems

IBRAHIM BARIS UYSAL, DURMUS ALI SUCU

ABSTRACT Network Intrusion Detection Systems (NIDS) play a critical role in identifying malicious activities within modern network infrastructures. As machine learning (ML)-based anomaly detection is increasingly used in modern NIDS deployments, new vulnerabilities have emerged in the form of adversarial attacks. These attacks involve subtle modifications to input traffic features—crafted to evade detection without altering the malicious payload. While most adversarial strategies focus on flow-level features or single packet attributes such as inter-packet timing (e.g., TANTRA), the potential of multi-feature flow-level adversarial manipulation remains significantly underexplored.

In this study, we propose a novel adversarial evasion framework that targets multiple flow-level features simultaneously—including TCP flags, time-to-live (TTL), fragmentation, window size, and inter-packet timing—aiming to generate realistic yet undetectable malicious traffic. Unlike prior approaches relying on heuristic search or deep reinforcement learning, we will adopt optimization-based strategies using Genetic Algorithms (GA) to efficiently explore the mutation space and discover adversarial variants that can bypass ML-based NIDS.

To evaluate our method, we plan to use publicly available datasets such as CICIDS2017, CICIDS2018, and test the generated traffic against both trained ML classifiers and open-source IDS platforms. This research aims to bridge the gap identified in recent surveys regarding the lack of flow-level adversarial analysis and provide a transferable, query-efficient, and explainable adversarial attack pipeline suitable for real-world NIDS evaluation and improvement.

INDEX TERMS Adversarial Machine Learning, Network Intrusion Detection Systems (NIDS), Flow-Level Evasion, Genetic Algorithms, Network Security, Traffic Manipulation, Explainable Attacks, ML-based Anomaly Detection.

I. INTRODUCTION

Network Intrusion Detection Systems (NIDS) are a critical component of modern cybersecurity infrastructures. By monitoring network traffic for signs of malicious activity, they provide a crucial defense layer against a wide range of attacks, from port scanning to advanced persistent threats. While traditional signature-based NIDS are effective against known threats, they often fail to detect new, zero-day attacks. To address this, anomaly-based NIDS powered by machine learning (ML) have been widely adopted. These systems learn statistical patterns of legitimate traffic and classify deviations as suspicious.

However, as shown by the growing body of research in adversarial machine learning (AML), ML-based NIDS are not immune to evasion. Carefully crafted inputs—called adversarial examples—can exploit model weaknesses to cause misclassification. In the context of NIDS, this means that attackers can subtly manipulate traffic features so that malicious flows are misclassified as benign. Recent studies such as He et al. (2023) have comprehensively surveyed this emerging threat landscape, identifying the overreliance on flow-level adversarial modifications (e.g., changes in average packet size or duration) and the critical lack of research in flow-level adversarial examples.

A few notable attempts have explored flow-level adversarial attacks. One of the earliest is TANTRA (Timing-based Adversarial Network Traffic Reshaping Attack), which manipulates inter-packet timing to evade detection. However,

TANTRA focuses solely on temporal features, leaving other flow-level fields unexplored. DeepPackGen (Almiani et al., 2022) introduced a GAN-based framework for generating synthetic malicious packets, but its goal was traffic generation rather than evasion. Similarly, packet-level GAN (PL-GAN) focused on generating realistic traffic from scratch rather than targeted evasion. DeepDGA explored adversarial domain generation, but operated at the DNS level rather than general packet headers. As highlighted by He et al., there is still no unified, explainable, and transferable adversarial framework that systematically targets multiple flow-level fields relevant to NIDS.

In this project, we aim to fill this gap by proposing a novel adversarial evasion framework that targets multiple flow-level features. Rather than relying on reinforcement learning or gradient-based methods, which are often computationally expensive and difficult to transfer across models, we leverage Genetic Algorithms (GA) to perform a guided search over the space of packet mutations. This approach allows us to craft adversarial traffic that is both realistic and effective, while requiring fewer queries to the NIDS model. Our goal is to provide a practical, explainable, and extensible method to evaluate and improve the robustness of ML-based NIDS in real-world environments.

II. BACKGROUND

Network Intrusion Detection Systems (NIDS) are essential components of cybersecurity architectures, tasked with identifying malicious activities within network traffic. With the growing adoption of machine learning (ML) for anomaly-based NIDS, the systems have become increasingly vulnerable to adversarial machine learning (AML) attacks — deliberate modifications of input traffic designed to evade detection without disrupting functionality [9].

The adversarial manipulation of machine learning models was first highlighted by Dalvi *et al.* [10], who demonstrated that small crafted perturbations could successfully fool spam classifiers. Subsequent work by Lowd and Meek [11] formalized the notion of adversarial reverse engineering, developing algorithms to learn enough about a classifier to mount effective attacks even without full model access. These early explorations laid the foundation for adversarial machine learning, culminating in Szegedy *et al.* [12]'s discovery of adversarial examples in deep neural networks.

A. EVOLUTIONARY COMPUTATION IN ADVERSARIAL EXAMPLE GENERATION

Among various approaches for generating adversarial examples, evolutionary computation methods — particularly Genetic Algorithms (GAs) — have proven highly effective in constrained and black-box attack settings. GAs are optimization techniques inspired by natural selection that evolve candidate solutions through operations such as crossover and mutation [13].

Alzantot *et al.* [14] introduced GenAttack, one of the first frameworks to apply GAs for adversarial attack generation without requiring gradient information, making it especially suitable for black-box scenarios. Xu *et al.* [15] proposed generic stochastic mutation frameworks for evaluating classifier robustness under adversarial manipulation, while Nguyen *et al.* [16] demonstrated the effectiveness of evolutionary methods in crafting synthetic images that fool convolutional neural networks.

In the specific domain of network security, Sheatsley [17] adapted adversarial crafting methods to constrained feature spaces, recognizing that many network protocol features cannot be freely perturbed without breaking functionality.

B. GENETIC ALGORITHMS FOR ADVERSARIAL ATTACKS ON NIDS

Building on these foundations, Alhajjar *et al.* [18] applied GAs for adversarial sample generation targeting ML-based NIDS. Their methodology involved evolving perturbations on selected mutable network features, carefully maintaining functionality constraints. The generated adversarial samples achieved high evasion rates across multiple classifiers, including Decision Trees, Random Forests, and Support Vector Machines, on benchmark datasets such as NSL-KDD and UNSW-NB15.

Other evolutionary approaches, such as AdversarialPSO by Mosli *et al.* [19], have demonstrated effectiveness in black-box adversarial traffic generation, although Particle Swarm Optimization differs fundamentally from GAs in exploration-exploitation dynamics. Furthermore, GAN-based methods such as MalGAN [20] have been proposed for adversarial malware generation, but typically require more extensive model information.

III. METHODOLOGY

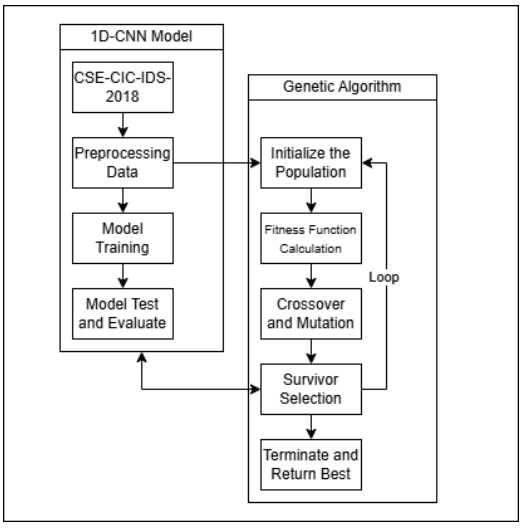


Figure 1: System Architecture

The overall system structure for the adversarial attack framework is illustrated in Figure 1. The workflow involves dataset preprocessing, training a 1D-CNN model, generating

adversarial mutations using a Genetic Algorithm (GA), and evaluating the attack success rate against the trained model. This project proposes a comprehensive framework for generating adversarial network traffic capable of evading machine learning-based Network Intrusion Detection Systems (NIDS) by modifying multiple flow-level features. The core objective is to craft minimally altered yet highly effective adversarial samples that exploit model weaknesses without compromising packet functionality. The methodology is structured into several key components, encompassing feature selection, mutation space definition, optimization-based attack generation, and systematic evaluation across various ML classifiers. ut affecting packet functionality. The methodology we aim to implement consists of the following planned components:

A. FEATURE SELECTION AND MUTATION SPACE

In this study, we focus on mutating selected statistical flow features extracted from network traffic, rather than directly manipulating low-level packet header fields. The mutation space is defined over a subset of features that are critical for the classification decision of ML-based NIDS models. Specifically, we target numerical attributes such as the forward and backward packet length means, overall packet length mean, TCP PUSH flag ratios, initial TCP window sizes, and similar aggregate metrics.

The choice of these features is motivated by their significant influence on flow-based anomaly detection systems, while ensuring that the fundamental packet structure remains valid and the attack traffic remains plausible. Each adversarial sample is generated by applying candidate mutations on these selected features within bounded perturbation ranges, maintaining domain validity (e.g., normalized feature limits between [0,1]). This structured mutation space allows effective evasion attempts without violating network protocol semantics or triggering basic flow-level validations.

B. OPTIMIZATION-BASED ATTACK GENERATION

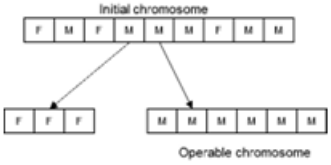
Genetic Algorithm (GA): Genetic Algorithms (GAs) are a class of evolutionary algorithms inspired by the process of natural selection. They are designed to solve optimization and search problems by mimicking biological evolution, employing operations such as selection, crossover (recombination), and mutation. GAs are particularly effective for solving complex problems where the search space is large, multimodal, and poorly understood.

In GAs, potential solutions to an optimization problem are encoded as chromosomes. A chromosome is typically represented as a string or array of genes, where each gene corresponds to a parameter of the solution. The algorithm begins with an initial population of chromosomes, often generated randomly. Each chromosome is evaluated using a fitness function that measures its quality relative to the optimization goal.

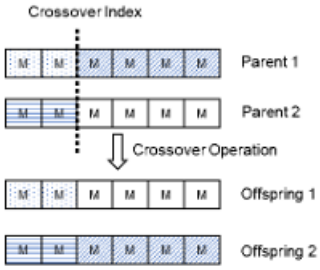
Selection mechanisms, such as tournament selection or roulette wheel selection, are used to preferentially choose fitter individuals for reproduction. Before applying evolutionary operators, each chromosome is divided into a mutable and an immutable part (see Figure 2a). Crossover operations combine portions of two parent chromosomes to produce offspring by exchanging their mutable parts (see Figure 2b). Mutation operations introduce small random changes into selected mutable genes, promoting exploration of the solution space (see Figure 2c).

Through iterative application of selection, crossover, and mutation over multiple generations, the population evolves toward increasingly optimal solutions. Elitism is often employed to ensure that the best solutions are preserved between generations. Termination conditions, such as a maximum number of generations or convergence of the population, determine when the algorithm stops.

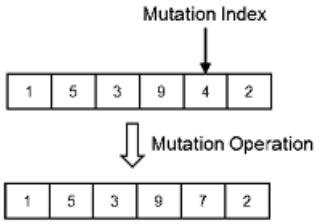
GAs have been successfully applied across a wide range of fields, including engineering design, machine learning, scheduling, and bioinformatics, due to their flexibility and robustness in handling complex optimization tasks.



(a) Initial chromosome transformation into operable (mutable) and fixed (immutable) parts.



(b) Crossover operation between two parent chromosomes to generate offspring.



(c) Mutation operation on a selected gene in the chromosome.

Figure 2: Illustration of genetic algorithm operations: (a) chromosome transformation, (b) crossover operation, and (c) mutation operation.

C. PLANNED EVALUATION

We plan to evaluate the proposed methods using the following setup:

- **ML-based classifiers:** We aim to train surrogate models such as Logistic Regression, MLP, and Random Forest using CICIDS2017, CICIDS2018 and NSL-KDD datasets.
- **Open-source NIDS platforms:** Snort and Suricata will be considered for downstream testing.
- **Evaluation metrics:** Evasion success rate, number of queries, and perturbation magnitude will be measured.

Tools such as `Scapy` and `tcpreplay` are intended to be used for crafting and replaying traffic, while feature extraction will be performed using `CICFlowMeter` or custom scripts. Through these planned experiments, we aim to gain insights into how ML-based NIDS can be improved against low-level, multi-feature adversarial manipulations.

IV. IMPLEMENTATION

A. DATASET PREPARATION

In this study, we utilized two publicly available benchmark datasets for network intrusion detection research:

- **CICIDS2017 Dataset** [22]: The CICIDS2017 dataset, published by the Canadian Institute for Cybersecurity (CIC), contains network traffic data generated in a realistic environment, including both benign activities and diverse types of modern attacks such as DDoS, brute force, web attacks, infiltration, and botnet activity. The traffic captures were collected over a period of five days, featuring labeled flow records extracted using CICFlowMeter. Each record includes over 80 statistical features derived from network packets. The dataset is made available for research purposes under the CIC's data sharing license [24].
- **CICIDS2018 Dataset** [23]: The CICIDS2018 dataset extends the 2017 version by incorporating additional modern attack scenarios (e.g., cryptojacking, infiltration, SQL injection) across multiple days of simulated realistic network activity. It features a diverse set of benign and malicious traffic patterns captured from user behavior emulation, IoT devices, and enterprise services. The dataset includes detailed labeled flow features, supporting both binary and multi-class intrusion detection tasks. The CICIDS2018 dataset is distributed under the same CIC data sharing license [24].

Preprocessing steps across both datasets included:

- Removal of records containing missing or infinite values.
- Normalization of numerical features to standard ranges. To avoid label leakage and spurious correlations, two families of variables were removed before training:

Identity / metadata fields – i.e., the raw 5-tuple (source and destination IP addresses and ports), the transport-layer protocol indicator, and absolute timestamps. Rationale: these fields can trivially reveal the ground-truth class (e.g., certain attack hosts or ports appear only in malicious flows) and therefore inflate performance in an unrealistic way.

Derived statistics – any attribute that is a direct arithmetic transformation of more basic counters, such as per-direction means, standard deviations and variances of packet length, average segment sizes, or byte/packet rates per second. Rationale: retaining only the underlying counters (total packets, total bytes, min/max sizes, etc.) eliminates multicollinearity, simplifies the GA repair step, and prevents the model from double-counting identical information.

B. FEATURE SELECTION AND MUTATION SPACE

Mutation-feature selection Only attributes that satisfy all three conditions below were admitted to the GA search space.

- Attacker-controllable after the three-way handshake. Handshake-only fields (e.g., initial window size) and absolute identifiers (IP/port) were excluded because a mid-flow adversary cannot modify them without resetting the connection.
- Primitive rather than derived. We retain counters and extremal statistics (totals, mins, maxes) and drop any feature that is merely an arithmetic transformation (means, variances, rates). This avoids collinearity and greatly simplifies post-mutation repair.

Applying this filter to the 62 numeric columns that remained after preprocessing (Section 2) produced the following 17-item white-list, grouped by semantic category:

- **Volume & size** {`TotLen Fwd Pkts`, `TotLen Bwd Pkts`, `Tot Fwd Pkts`, `Tot Bwd Pkts`, `Fwd Pkt Len Max`, `Fwd Pkt Len Min`, `Bwd Pkt Len Max`, `Bwd Pkt Len Min`}
- **Timing** {`Flow Duration`, `Flow IAT Mean`, `Flow IAT Std`, `Fwd IAT Mean`, `Bwd IAT Mean`}
- **TCP flags** {`Fwd PSH Flags`, `Bwd PSH Flags`, `ACK Flag Cnt`, `FIN Flag Cnt`}

Together these variables expose the principal dimensions an active attacker can plausibly manipulate: traffic volume, segmentation strategy, pacing, and signalling semantics.

C. PROTOCOL-AWARE CONSTRAINT DESIGN

To keep every mutated feature vector representative of a physically valid TCP flow, we apply the following post-mutation constraints in a dedicated repair step:

leftmargin=*,label=–

- **Integer casting:** All packet, byte, and flag counters are rounded to the nearest nonnegative integer (e.g., 2.7 ACKs $\rightarrow$ 3).
- **Flag within packet bounds:** `Fwd/Bwd PSH Flags`, `ACK Flag Cnt` and `FIN Flag Cnt` may not exceed their respective packet counts; at most two FINs are permitted (one per host).
- **MTU bound:** `Pkt Len Max/Min` $\leq$ 1460 B to comply with Ethernet MTU 1500 B (TCP MSS $\approx$ 1460 B).

- **Min–max coherence:** `Pkt Len Min` $\leq$ `Pkt Len Max` for each direction.
- **Bytes upper bound:** `TotLen` $\leq$ 1460 B $\times$ `Tot Pkts`; the aggregate payload cannot exceed the packet count multiplied by the MSS limit.
- **Timing consistency:** The overall Flow Duration must be consistent with the average interarrival time (IAT Mean). Concretely, the duration is allowed to deviate by at most 20 percent from (TotalPkts -1) $\times$ IAT Mean, thereby eliminating flows whose timing statistics are physically unrealistic.

During repair we (i) enforce the rules above, (ii) clip extreme values if necessary, and (iii) rescale the vector back to the $[0, 1]$ domain before feeding it to the GA fitness function. Consequently, the search is confined to syntactically and semantically valid TCP flows, facilitating subsequent packet-level synthesis and offline Suricata evaluation.

D. ADVERSARIAL SAMPLE GENERATION VIA GENETIC ALGORITHM

The genetic algorithm (GA) implemented in this study aims to generate minimally perturbed adversarial samples capable of evading detection by ML-based NIDS models. Algorithm 1 outlines the process. An initial population is created by bounded sampling around mutable features. Individuals are evolved over multiple generations using crossover, mutation, and selection strategies. The fitness function prioritizes minimizing the attack detection probability while also reducing the distortion from the original sample. An elite preservation strategy ensures that the best individual survives across generations. The final output is the best adversarial sample found for a given input.

Algorithm 1 Genetic Algorithm for Adversarial Sample Generation

Input: Original sample x, mutable feature indices $\mathcal{M}$, trained model f
Initialize population P of individuals by bounded uniform sampling around $x[\mathcal{M}]$
for generation $g = 1$ to G **do**
  for each individual $i \in P$ **do**
    Construct full sample x_i by inserting i into $x[\mathcal{M}]$
    Predict class probabilities $f(x_i)$
    Compute fitness: minimize attack probability and distortion
  end for
  Select offspring via tournament selection
  Apply two-point crossover to offspring with probability p_c
  Apply Gaussian mutation to offspring with probability p_m
  Clip mutated feature values to $[0, 1]$
  Evaluate fitness of new offspring
  Preserve best elite individual into next generation
  Replace worst individual with elite
end for
Output: Best adversarial individual found

V. RESULTS

In order to comprehensively evaluate the generalizability and effectiveness of the proposed Genetic Algorithm (GA)-based adversarial attack methodology, we conducted experiments across four different machine learning classifiers: 1D Convolutional Neural Network (1D-CNN), Logistic Regression (LR), Random Forest (RF), and XGBoost. For each model, we analyzed the changes in attack probability and class probability distributions before and after adversarial mutation.

The primary goal of the GA attack was to manipulate malicious traffic samples such that their likelihood of being classified as benign significantly increased, while introducing minimal feature modifications.

A. ATTACK PROBABILITY BEFORE AND AFTER MUTATION

The attack probability before and after adversarial mutation was analyzed for each classifier: 1D Convolutional Neural Network (1D-CNN), Logistic Regression (LR), Random Forest (RF), and XGBoost. Figures 3–6 present scatter plots showing attack probabilities before and after the adversarial mutation for each model.

For the 1D-CNN model (Figure 3), a significant proportion of malicious samples experienced a substantial reduction in attack probability, successfully crossing below the classification threshold and being misclassified as benign. Logistic Regression (Figure 4) demonstrated the highest vulnerability, with adversarial mutations leading to

a pronounced shift towards benign classifications, even with minimal feature perturbations.

The Random Forest classifier (Figure 5) showed relatively better resistance compared to CNN and LR, although successful adversarial examples were still generated. Notably, the XGBoost model (Figure 6) exhibited the highest robustness among the tested models, maintaining correct attack classifications for a significant fraction of adversarially mutated samples.

Overall, the GA attack consistently reduced attack detection probabilities across all models, though with varying degrees of success depending on the classifier complexity and decision boundaries.

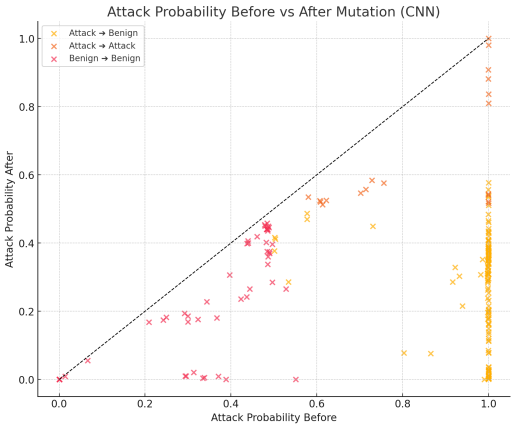


Figure 3: Attack probability before and after adversarial mutation for 1D-CNN model.

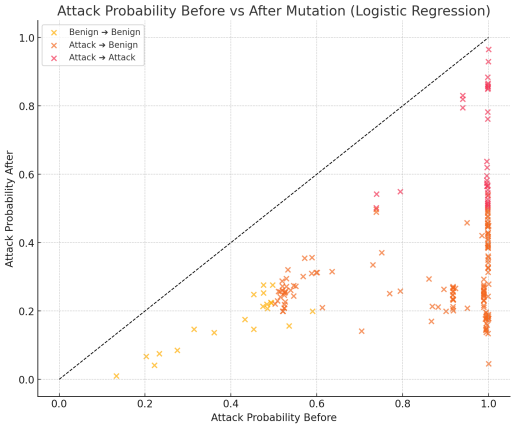


Figure 4: Attack probability before and after adversarial mutation for Logistic Regression model.

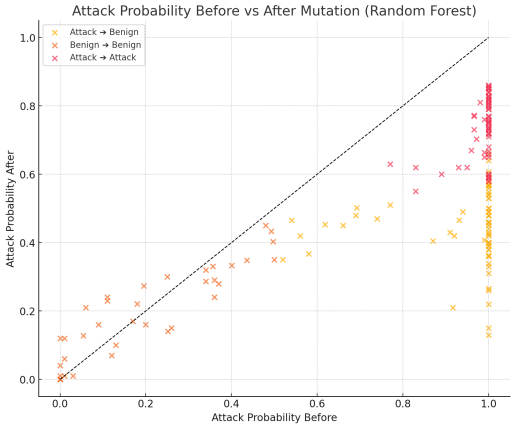


Figure 5: Attack probability before and after adversarial mutation for Random Forest model.

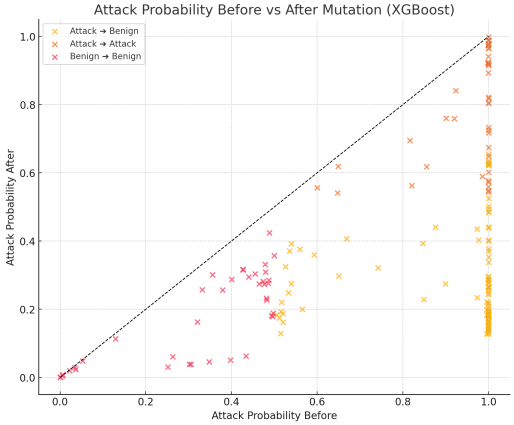


Figure 6: Attack probability before and after adversarial mutation for XGBoost model.

B. AVERAGE CLASS PROBABILITIES BEFORE AND AFTER MUTATION

To further assess the impact of the adversarial mutations, we computed the average predicted probability vectors across all samples before and after mutation. The classification task spanned ten classes, corresponding to:

- Class 0: Benign
- Class 1: Bot
- Class 2: DDoS attacks-LOIC-HTTP
- Class 3: DoS attacks-GoldenEye
- Class 4: DoS attacks-Hulk
- Class 5: DoS attacks-SlowHTTPTest
- Class 6: DoS attacks-Slowloris
- Class 7: FTP-BruteForce
- Class 8: Infiltration
- Class 9: SSH-Bruteforce

For the 1D-CNN model (Figure 7), after mutation, the average probability assigned to the Benign class (Class 0) increased significantly, while the probabilities corresponding to attack classes (Classes 1–9) decreased sharply. Logistic

Regression (Figure 8) exhibited an even more dramatic shift, with predictions overwhelmingly favoring the benign class after mutation.

The Random Forest model (Figure 9) displayed some residual probability mass across attack classes, particularly FTP-BruteForce and Infiltration, indicating partial recognition of adversarial samples. In contrast, the XGBoost model (Figure 10) maintained notable average probabilities for certain attack classes such as DDoS attacks-LOIC-HTTP and DoS attacks-SlowHTTPTest, demonstrating improved robustness compared to other models.

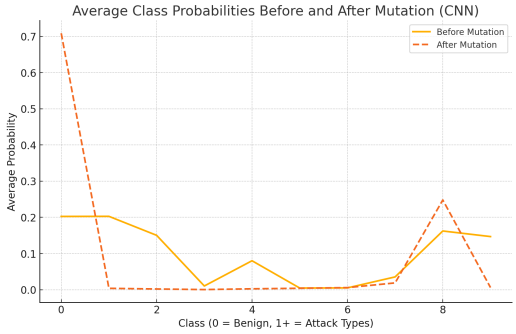


Figure 7: Average class probabilities before and after adversarial mutation for 1D-CNN model.

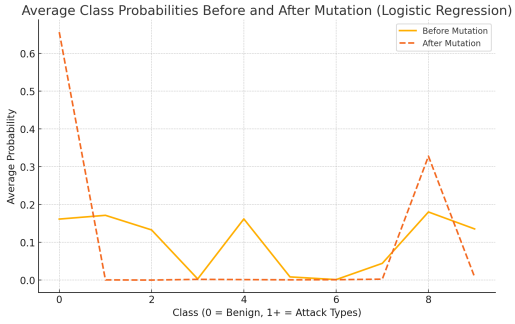


Figure 8: Average class probabilities before and after adversarial mutation for Logistic Regression model.

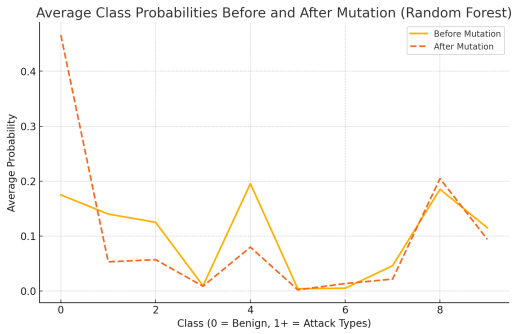


Figure 9: Average class probabilities before and after adversarial mutation for Random Forest model.

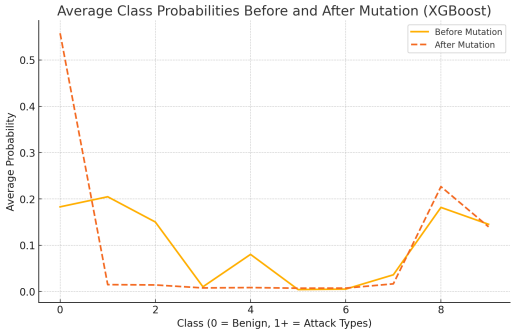


Figure 10: Average class probabilities before and after adversarial mutation for XGBoost model.

Table 1: GA evasion effectiveness on four classifiers ($n = 200$ flows each)

Model	Evasions	Rate (%)	$\overline{\ell_2}$	$\widetilde{\ell_2}$	$\overline{\ell_0}$
1D–CNN	136	68	0.404	0.401	6.13
Logistic Regression	146	73	0.514	0.511	6.49
Random Forest	76	38	0.199	0.200	6.29
XGBoost	98	49	0.228	0.231	6.44

Four quantitative indicators are reported for every classifier:

- **Evasion Rate (%)** – the proportion of test flows that are predicted malicious before mutation and benign afterwards. It is the primary measure of attack success.
- $\overline{\ell_2}$ **(mean ℓ_2 distortion)** – the average Euclidean distance $\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \mathbf{x}'_i\|_2$ between the original feature vector $\mathbf{x}_i$ and its mutated counterpart $\mathbf{x}'_i$. It captures how far, on average, the GA must move a flow in the continuous feature space.
- $\widetilde{\ell_2}$ **(median ℓ_2 distortion)** – the 50th-percentile of the same distance distribution, less sensitive to extreme outliers than the mean.
- $\overline{\ell_0}$ **(mean ℓ_0 distortion)** – the average number of mutated coordinates, $\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \mathbf{x}'_i\|_0$. Because the GA operates on a 17-dimensional white-list, a value of 6.4 indicates that, typically, only six or seven flow attributes (packet counts, sizes or flag counters) need to be altered to achieve evasion.

Together these metrics answer three complementary questions: (i) How often does the attack succeed? – the evasion rate; (ii) How much does a flow have to change in magnitude? – the ℓ_2 measures; and (iii) How many individual statistics must be manipulated? – the ℓ_0 count. Low distortion concomitant with a high evasion rate signals a particularly effective and stealthy attack strategy. An evasion is counted only when the flow, originally predicted as *malicious*, is re-labelled *benign* after mutation. The metric ignores cases where the flow is reassigned to a different attack class. Consequently, the rates in Table 1 represent a lower bound on the true misclassification effect; a full analysis would require post-GA confusion matrices across all attack categories.

C. GENERALIZATION ACROSS CLASSIFIERS

The Genetic Algorithm successfully generated adversarial samples capable of deceiving all tested machine learning classifiers. Logistic Regression exhibited the highest susceptibility to adversarial attacks, as evidenced by the sharp probability shifts and high evasion rates. In contrast, XGBoost maintained relatively higher resilience, preserving detection capabilities against some perturbed samples. Despite these differences, the overall trend indicated that optimization-based adversarial attacks could generalize effectively across various model architectures.

D. DISCUSSION

The experimental results demonstrate that even relatively simple optimization-based methods, such as Genetic Algorithms, can substantially degrade the detection performance of machine learning-based NIDS models. The adversarial mutations not only lowered the attack detection probability but also shifted model predictions heavily toward benign classifications. These findings underline the urgent need for enhancing adversarial robustness in NIDS systems, particularly in black-box scenarios where attackers exploit model weaknesses without explicit access to model internals.

References

[1] K. He, D. Kim, and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 221–255, 2023.

[2] Z. Li, Q. Lin, S. Zhu, J. Wang, J. Xu, and X. Guan, "TANTRA: Timing-Based Adversarial Network Traffic Reshaping Attack," in arXiv preprint arXiv:2103.06297, 2021.

[3] M. Almiani, Y. Jararweh, S. Otoum, and K. Salah, "DeepPackGen: A Deep Learning-Based GAN for Generating Adversarial Network Traffic at the Packet Level," Computers & Security, vol. 115, 2022.

[4] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.

[5] 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), 2018.

[6] J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence, "Batch Bayesian Optimization via Local Penalization," arXiv preprint arXiv:1505.08052, 2015. [Online]. Available: https://github.com/SheffieldML/GPyOpt

[7] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," Journal of Machine Learning Research, vol. 13, no. 1, pp. 2171–2175, 2012. [Online]. Available: https://github.com/DEAP/deap

[8] P. Biondi, "Scapy," [Online]. Available: https://github.com/secdev/scapy

[9] K. He, D. Kim, and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 221–255, 2023.

[10] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial Classification," *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[11] D. Lowd and C. Meek, "Adversarial Learning," *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[12] C. Szegedy *et al.*, "Intriguing Properties of Neural Networks," *arXiv preprint arXiv:1312.6199*, 2013.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[14] M. Alzantot *et al.*, "GenAttack: Practical Black-box Attacks with Gradient-Free Optimization," *Proceedings of GECCO*, 2019.

[15] W. Xu, Y. Qi, and D. Evans, "Automatically Evading Classifiers," *Network and Distributed System Security Symposium (NDSS)*, 2016.

[16] A. Nguyen, J. Yosinski, and J. Clune, "Deep Neural Networks are Easily Fooled," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] R. M. Sheatsley, *Adversarial Examples in Constrained Domains*, Ph.D. thesis, Pennsylvania State University, 2018.

[18] E. Alhajjar, P. Maxwell, and N. D. Bastian, "Adversarial Machine Learning in Network Intrusion Detection Systems," *arXiv preprint arXiv:2004.11898*, 2020.

[19] R. Mosli, M. Wright, B. Yuan, and Y. Pan, "They Might NOT Be Giants: Crafting Black-Box Adversarial Examples with Fewer Queries Using Particle Swarm Optimization," *arXiv preprint arXiv:1909.07490*, 2019.

[20] W. Hu and Y. Tan, "Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN," *arXiv preprint arXiv:1702.05983*, 2017.

[21] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: From Phenomena to Black-Box Attacks," *arXiv preprint arXiv:1605.07277*, 2016.

[22] Canadian Institute for Cybersecurity (CIC), "CICIDS2017 Dataset," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[23] Canadian Institute for Cybersecurity (CIC), "CICIDS2018 Dataset," [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[24] Canadian Institute for Cybersecurity (CIC), "Data Sharing License," [Online]. Available: https://www.unb.ca/cic/about/data-sharing.html