Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

ROBUSTNESS OF TREE-BASED MACHINE LEARNING ALGORITHMS AGAINST ADVERSARIAL EXAMPLES IN NETWORK INTRUSION DETECTION SYSTEM (NIDS): A COMPARATIVE STUDY OF RANDOM FOREST AND XGBOOST

David Hamonangan Sirait [1]*, Andy Triwinarko [2]**
* Teknik Informatika, Politeknik Negeri Batam
**Rekayasa Keamanan Siber, Politeknik Negeri Batam
david.h.sirait24404@gmail.com [1], andy@polibatam.ac.id

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

Adversarial machine learning, feature squeezing, random forest, xgboost, zoo attack

ABSTRACT

The security of machine learning-based Network Intrusion Detection Systems (NIDS) is increasingly threatened by Adversarial Machine Learning (AML) attacks capable of manipulating input data to deceive models and evade detection. The RobEns study [1] built a comprehensive adversarial ensemble framework evaluating four black-box attacks against six models on IoT traffic data, ye left three research gaps unresolved: the accuracy of the substitute model used to represent Random Forest was never validated, XGBoost was excluded from the evaluation entirely, and the single-step adversarial training scheme suffered from label leakage. This study addresses those gaps through a direct comparative analysis between Random Forest and XGBoost on the ToN-IoT dataset, using two complementary black-box attack techniques, ZOO Attack and Genetic Adversarial Attack (GAA), together with Feature Squeezing as a data-level defense mechanism. The methodology comprises four phases: data collection and preprocessing, baseline model training, adversarial attack implementation under two feature-encoding schemes, and defense implementation with comparative evaluation using six metrics: Standard Accuracy (SA), Adversarial Accuracy (AA), Robust Accuracy (RA), Attack Success Rate (ASR), Robust Accuracy Gain (RAG), and SA Preservation Rate (SPR). Results show that both models strongly resist ZOO Attack (ASR=0%) owing to their non-differentiable, piecewise-constant decision structure, but respond very differently to GAA: Random Forest retains strong robustness under One-Hot Encoding (ASR=0%) yet becomes highly vulnerable under Ordinal Encoding with StandardScaler (ASR = 99,90%), whereas XGBoost remains consistently vulnerable (ASR = 100%) regardless of the encoding scheme. Feature Encoding (RAG = 28.87%) but offers little benefit in the continuous feature space. These findings indicate that feature representation is as critical as model architecture in determining adversarial robustness, offering practical guidance for designing more resilient tree-based NIDS.

I. INTRODUCTION

Cyberattacks against network infrastructure continue to rise significantly, with projected global losses reaching $10.5 trillion per year by 2025 [2]. Static signature-based detection approaches have proven inadequate against continuously evolving threats, driving the adoption of machine learning-based Network Intrusion Detection Systems (NIDS) [3]. In the IoT context, tree-based algorithms such as Random Forest (RF) and XGBoost have

e-ISSN: 2548-6861

become the primary choice due to their simplicity and efficiency, which suit the resource constraints of IoT devices [4]. However, ML-based NIDS have proven vulnerable to Adversarial Machine Learning (AML), in which carefully crafted mathematical perturbations known as adversarial examples are injected into input data to cause model misclassification and evade detection [5].

The selection of Random Forest and XGBoost as the models evaluated in this study is based on three complementary scientific arguments, rather than mere methodological preference. First, from a practical performance standpoint, recent comparative studies consistently show that these two tree-based models can match or even surpass the accuracy of deep learning models (DNN, CNN-BiLSTM) on network intrusion detection tasks, with far lower inference latency (sub-millisecond) and without requiring a GPU or large training data volumes [4], [6], making them more realistic to deploy on resource-constrained IoT devices compared to complex neural network architectures. Second, from an operational perspective, Random Forest and XGBoost produce interpretable feature importance, enabling security analysts to understand the reasoning behind each classification decision, a critical need in production NIDS that is not always met by black-box models [7]. Third, and the main methodological motivation of this study, the non-differentiable nature of both models is itself the primary object of investigation: unlike neural networks, which have long been the dominant focus of adversarial machine learning research [7], the robustness of tree-based models against adversarial attacks has only begun to receive serious attention in the last five years [8], [9], meaning that a comparative evaluation of Random Forest and XGBoost in an IoT-based NIDS context remains an underexplored research area.

The primary prior study serving as the main reference is RobEns [1], an adversarial ensemble framework that tested four black-box attacks (FGSM, C&W, ZOO, HopSkipJump) against six ML models on three IoT datasets. RobEns made a significant contribution by comprehensively mapping the vulnerabilities of ML-based NIDS, yet left three research gaps unresolved.

First, RobEns applied attacks and defenses uniformly across all models without accounting for fundamental architectural differences. Tree-based models such as Random Forest are non-differentiable, with decisions based on discrete rules rather than differentiable functions, so gradient-based attacks such as FGSM cannot be applied directly. The accuracy of the substitute model in representing Random Forest was neither measured nor controlled in RobEns, meaning the resulting vulnerability evaluation for Random Forest is not fully valid [10].

Second, XGBoost was not included in RobEns, making it impossible to determine whether the vulnerabilities observed in Random Forest were a model-specific weakness or a general characteristic of tree ensemble architectures.

Third, Adversarial Training in RobEns used single-step FGSM, which causes label leakage, where information from the true label leaks into the adversarial examples, resulting in an artificially inflated robust accuracy, as explicitly acknowledged in RobEns ([1], Section 4.4) yet left unresolved [5].

This study is designed as a direct continuation of RobEns to address these three gaps, which also constitute the main source of novelty in this research. The first novelty lies in the explicit validation of the equivalence between the substitute model and the original Random Forest prior to attack evaluation, something RobEns did not do. The second novelty is the inclusion of XGBoost as a direct comparator to Random Forest within an identical adversarial evaluation framework, allowing isolation of whether observed vulnerabilities are model-specific or a general characteristic of tree ensemble architecture. The third novelty is the selection of ZOO Attack and Genetic Adversarial Attack, both of which are free from the label leakage issue that affects the single-step FGSM adversarial training scheme in RobEns. With these three novel contributions, this study is not intended to replace RobEns, but rather to extend and deepen RobEns's findings specifically in the context of tree-based models, positioning it as complementary and continuation research relative to the previously established evaluation framework. The focus is placed on two tree-based models (Random Forest and XGBoost) using two carefully selected adversarial attack techniques — ZOO Attack and Genetic Adversarial Attack — along with one data-based defense mechanism, Feature Squeezing. The selection of these two attack techniques is based on the following strong scientific arguments.

ZOO Attack (Zeroth-Order Optimization Attack) was chosen because it is the only score-based attack retained from RobEns [10]. ZOO operates without requiring access to the model's gradient; it numerically estimates the gradient via finite differences using only the model's probability output. This makes ZOO directly compatible with non-differentiable tree-based models, without requiring a substitute model. The key methodological advantage of ZOO in this study is its ability to provide a direct comparison baseline against the ZOO results already reported in RobEns for the Random Forest model, allowing the impact of architectural differences (Random Forest vs XGBoost) to be isolated without changing the attack type. In addition, ZOO produces adversarial examples in a normalized continuous feature space, which is highly relevant for tabular NIDS data, where each feature represents a network traffic characteristic that can be subtly manipulated.

Genetic Adversarial Attack was chosen as the second attack technique because it represents an evolutionary optimization paradigm that is fundamentally different from

ZOO. Genetic algorithm-based attacks use selection, crossover, and mutation operators to explore a discrete, non-continuous perturbation space, a characteristic highly relevant to tree-based models whose decisions consist of discrete thresholds at each node [11], [12]. Unlike gradient-based or zeroth-order attacks that assume a smooth decision space, Genetic Attack naturally handles the discontinuous decision boundaries characteristic of Random Forest and XGBoost. By using a fitness operator that measures the degradation of the original class probability, Genetic Attack can find effective adversarial examples even when the model's decision landscape is highly irregular. The combination of ZOO (gradient-free, score-based) and Genetic Attack (evolution-based, purely black-box) provides comprehensive evaluation coverage from two orthogonal attack paradigms, making the vulnerability evaluation of Random Forest and XGBoost more representative and scientifically valid.

Feature Squeezing was chosen as the sole defense mechanism for several solid scientific reasons. First, Feature Squeezing is a data-level defense that works by reducing the numerical precision of features through bit-depth compression, degrading adversarial perturbations that rely on high precision before they reach the model [13]. This type of defense is model-agnostic, independent of the model's internal architecture, and can therefore be applied identically and fairly to both Random Forest and XGBoost without any modification to either model. Second, Feature Squeezing was already used in RobEns, so its results can be directly compared against the RobEns baseline, enabling quantification of improvements specific to the tree-based model context. Third, from a computational efficiency perspective, Feature Squeezing requires no model retraining or heavy optimization iterations; it only processes input data before it is fed to the model, allowing rapid implementation without computational resource constraints. Fourth, Feature Squeezing serves as a pure experimental control, since this defense does not touch the model itself, the measured impact on robust accuracy directly reflects each model's inherent ability to handle precision-reduced data, rather than being an artifact of changes to model parameters.

The study is limited to a single representative dataset (ToN-IoT) to allow in-depth exploration of two adversarial attack techniques and a defense mechanism on both tree-based models within the available research timeframe.

II. METHOD

The research methodology is designed to replicate and extend the RobEns experimental pipeline with components specific to tree-based models, using two attack techniques (ZOO Attack and Genetic Adversarial Attack) and one defense mechanism (Feature Squeezing).

A. Methodology Overview

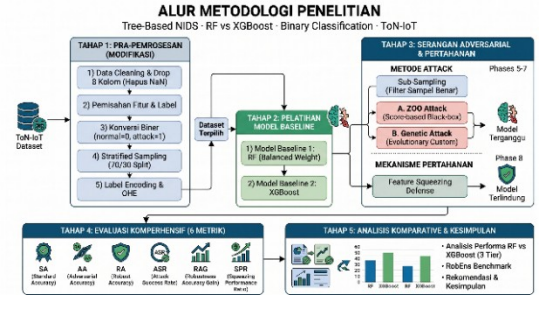

Figure 1. Research Methodology Flow

This study was conducted in four sequential phases: (1) Data Collection and Preparation, (2) Baseline Model Training, (3) Adversarial Attack Implementation, and (4) Defense Implementation and Comparative Evaluation. The entire implementation uses Python 3.10+ with the scikit-learn, XGBoost, and Adversarial Robustness Toolbox (IBM ART ≥ 1.17) libraries.

B. Data Collection and Preparation

The dataset used is ToN-IoT. Data acquisition was automated via the kagglehub library API to maintain repository consistency and research reproducibility.

Dataset	Publisher	Year	Classes	Features	Records	Role
ToN-IoT	Cyber Range Lab, UNSW	2020	9	31	~22 million	Main Dataset

Table 1. Summary of the Dataset Used

1. Data Acquisition and Cleaning

   The ToN-IoT dataset was downloaded from the Kaggle repository arnobbhowmik/ton-iot-network-dataset using kagglehub. If multiple CSV files were present, they were all merged using pd.concat(ignore_index=True). Non-informative features removed included src_ip, dst_ip, src_port, dst_port, timestamp, transaction_id, attack_cat, and type. Rows with NaN or Inf values were removed using .dropna().

2. Feature and Label Separation

   The ToN-IoT dataset was partitioned into features (X) and a binary target label (y). The label was derived from the type column, converted into binary format: all 9 attack types (backdoor, ddos, dos, injection, mitm, password, ransomware, scanning, xss) were merged into a single attack class, while normal traffic remained a separate class. The result is two classes: normal and attack.

3. 70/30 Stratified Split

   Stratified sampling ensures that the proportion of each attack class is preserved identically in the training and testing sets, avoiding evaluation bias caused by class imbalance.

4. Label Encoding and One-Hot Encoding

   The target label was derived from the type column via manual conversion to binary using a lambda function: y_binary = y.apply(lambda x: ‘normal’ if x

e-ISSN: 2548-6861

== ‘normal’ else ‘attack’). Encoding was performed manually and explicitly: normal = 0 (negative/benign class) and attack = 1 (positive/malicious class), without LabelEncoder() so that class semantics remain transparent. The result: normal = 50,000 samples (23.69%) and attack = 161,043 samples (76.31%), a ratio of 3.22:1. Categorical features were transformed using pd.get_dummies(drop_first=True) after the split (post-split) to prevent data leakage, with column alignment using .align(join=‘left’, axis=1, fill_value=0).

C. Baseline Model Training

Both models were trained using isolated static hyperparameters without GridSearchCV, aimed at efficiently mapping the generalization capacity of the bagging (RF) and sequential boosting (XGBoost) architectures within the available time budget.

Parameter	Value	Description
n_estimators	100	100 independent parallel trees
max_depth	15	Depth limit to reduce overfitting
random_state	42	Experiment reproducibility
n_jobs	-1	Full parallelism across all CPU cores
class_weight	balanced	Automatic class imbalance handling

Table 2. Random Forest Hyperparameter Configuration

Parameter	Value	Description
n_estimators	100	Number of boosting rounds
max_depth	6	Shallow depth to keep weak learners simple
learning_rate	0.1	Optimization step-size shrinkage coefficient
eval_metric	mlogloss	Multi-class cross-entropy objective function
subsample	1.0	100% of data used per round
colsample_bytree	1.0	100% of features used per tree
random_state	42	*Experiment reproducibility*

Table 3. XGBoost Hyperparameter Configuration

Baseline evaluation reports: Train Accuracy (overfitting detection), Standard Accuracy (SA), Precision, Recall, and F1-Score (macro-averaged) per model per dataset.

D. Adversarial Attack Implementation

Two attack techniques were applied in a black-box scenario. Because ZOO Attack has high computational complexity, a sub-sample of n=1,000 was used for the ZOO Attack and Genetic Adversarial Attack evaluations. The computational complexity of ZOO Attack on high-dimensional data such as images (e.g., MNIST: 784 features; CIFAR-10: 3,072 features) has generally constrained the sub-sample sizes used in prior studies. The tabular ToN-IoT dataset, with 31 features, has a ZOO computational load of only ~4% of MNIST, so n=1,000 remained feasible within the research timeframe while providing a tighter margin of error (±3.10%, Wilson score 95% CI) and more adequate representation for minority attack classes (at least ~8 samples per class with stratified sampling). Sub-sampling was performed using StratifiedShuffleSplit with two critical requirements: (1) only samples correctly classified by the baseline model were included — consistent with the ASR formula and the RobEns precedent (Alkadi et al., 2024), which explicitly excludes already-misclassified samples — so the ASR denominator is consistently n=1,000 for both models; and (2) the class distribution within the 1,000 samples is proportional to the original y_test distribution to account for class imbalance in both datasets. Sampling was performed separately for RF and XGBoost, since the set of correctly classified samples may differ between models. Models were wrapped using SklearnClassifier (RF) and XGBoostClassifier from IBM ART, with clip_values=[float(np.min(X_train)), float(np.max(X_train))].

Parameter	Value	Justification
confidence	0.0	Perturbation is sufficient to cross the decision boundary without excessive margin
learning_rate	0.01	Step-size descent; a small value for precise gradient estimation
max_iter	30	Maximum iterations per sample; balances effectiveness and efficiency
binary_search_steps	5	Binary search for the perturbation penalty constant
initial_const	1e-3	Initial base constant for the penalty function
abort_early	True	Early stopping once convergence is reached before max_iter
use_resize	False	Disabled: this feature is for 2D image data, not relevant for tabular NIDS
use_importance	False	Disabled: could disrupt tabular feature

		interdependency integrity
nb_parallel	10	10 parallel computation processes for time efficiency
batch_size	1	One sample processed per batch for stability
Scenario	Black-box score-based	Accesses only probability output; requires no gradient or model architecture

Table 4. ZOO Attack Configuration

Parameter	Value	Justification
max_iter (generations)	100	Evolution iteration limit; sufficient for convergence on tabular data
pop_size (population size)	40	Number of individuals per generation; balances diversity and efficiency
eps (epsilon, ε)	0.1	L∞ perturbation bound; consistent with the ZOO configuration for comparability
mutation_rate	0.1	Per-gene mutation probability; 10% to avoid local optima without disrupting good solutions
discrete_features	True for flag/port features	Discrete mutation for network features with rigid values (flag count, port)
Fitness function	1 − P(y_true | x')	Measures degradation of the original class probability; larger = more effective perturbation
Scenario	Pure black-box	Requires only predict_proba(); no gradient, architecture, or model structure needed

Table 5. Genetic Adversarial Attack Configuration

E. Feature Squeezing Defense Implementation

Feature Squeezing was implemented as the sole defense mechanism using the FeatureSqueezing object from the IBM ART library. This defense was applied to adversarial examples from both attacks (ZOO and Genetic Attack) for both models (RF and XGBoost), producing four defense scenarios:

1. FS+ZOO+RF
2. FS+ZOO+XGBoost
3. FS+Genetic+RF
4. FS+Genetic+XGBoost.

Parameter	Value	Justification
bit_depth	4	Precision reduced to 4-bit; lower than RobEns (5-bit) to maximize the squeezing effect on score-based attacks
clip_values	[min(X_train), max(X_train)]	Clipping range based on the training data distribution; prevents out-of-distribution features
detection_threshold (θ)	0.1	Prediction difference: if |f(x) − f(x_sq)| > 0.1, the input is classified as adversarial
Applied to	X_adv (ZOO) and X_adv (Genetic)	Defense applied to adversarial examples from both attacks for both models
Library	art.defences.preprocessor.FeatureSqueezing	Official IBM ART implementation; pip install adversarial-robustness-toolbox

Table 6. Feature Squeezing Defense Configuration

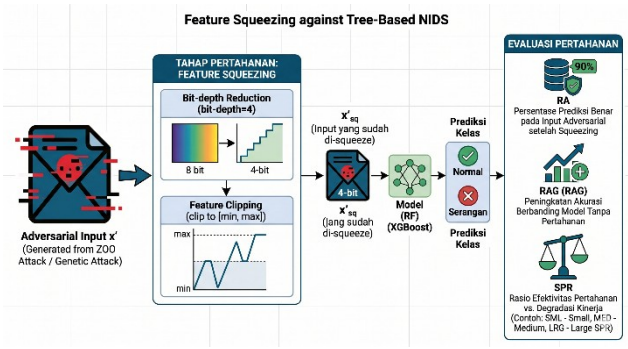

Figure 2. Feature Squeezing Defense Mechanism Flow Diagram

Evaluation procedure: (1) Compute SA_baseline from the entire clean X_test using the baseline model; (2) Apply Feature Squeezing to X_adv to produce X_adv_sq; (3) Evaluate the model on X_adv_sq to obtain RA; (4) Compute RAG = RA − AA; (5) Apply Feature Squeezing to the entire X_test_full (not just the attack subset) to produce X_test_full_sq, then evaluate the model on X_test_full_sq (squeezed clean data) to obtain SA_post; (6) Compute SPR = SA_post / SA_baseline × 100%. SA_post must be computed from the full X_test (not from the 1,000-sample attack subset) so that SPR representatively reflects the impact of Feature Squeezing on the model's general

e-ISSN: 2548-6861

performance on clean data; (7) Verify SPR ≤ 100% as a label-leakage check.

F. Comparative Evaluation and Analysis

Evaluation was conducted across three tiers to build a comprehensive understanding:

a) **Tier 1 — Per-Model Evaluation**: For each model (RF and XGBoost), SA, AA per attack (ZOO, Genetic), RA per defense scenario (FS+ZOO, FS+Genetic), ASR, RAG, and SPR are reported for both datasets.

b) **Tier 2 — Comparative Analysis of RF vs XGBoost:** A direct comparison of both models across all metrics to answer the question: which model is intrinsically more resistant to ZOO Attack and Genetic Adversarial Attack, and which model benefits more effectively from Feature Squeezing?

c) **Tier 3 — Benchmarking Against RobEns:** This study's ZOO Attack results against RF are compared with the ZOO results against RF in RobEns (Tables 7–9, Alkadi et al., 2024) to validate reproducibility and isolate the impact of differences in evaluation methodology.

Metric	Formula	Purpose	Target
Standard Accuracy (SA)	$\text{SA} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\%$	Baseline performance without attack	Reference
Adversarial Accuracy (AA)	$AA = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(f_\theta(x_{adv,i}) = y_i) \times 100\%$	Performance under attack	Low → effective attack
Robust Accuracy (RA)	$RA = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(f_{\theta_{robust}}(x_{adv,i}) = y_i) \times 100\%$	Performance after defense	RA > AA → effective defense
Attack Success Rate (ASR)	$\text{ASR} = \frac{\lvert\{x \in X_{test}: f(x) = y_{true} \wedge f(x') \neq y_{true}\}\rvert}{\lvert\{ \in X_{test}: f(x) = y_{true}\}\rvert}$	Attack effectiveness on correctly classified samples	ASR > 20%
Robust Accuracy Gain (RAG)	$RAG = RA - AA$	Pure defense contribution	RAG > 10%
SA Preservation Rate (SPR)	$\text{SPR} = \frac{SA_{post-defense}}{SA_{original}} \times 100\%$	Label-leakage detection & SA trade-off	90% ≤ SPR ≤ 100%

Table 7. Evaluation Metrics and Formulas

Note: D(x') = Feature Squeezing applied to adversarial example x'; SA_post = Standard Accuracy after Feature Squeezing is applied to clean data; SA_baseline = Standard Accuracy before defense. SPR > 100% indicates label leakage

III. Results and Discussion

A. Data Preprocessing Results

The ToN-IoT dataset was successfully loaded with 211,043 rows and 44 columns. After removing non-informative columns (src_ip, dst_ip, src_port, dst_port, timestamp, transaction_id, attack_cat, label) and dropping NaN rows, the dataset retained 10 traffic classes under the type column, dominated by normal traffic (50,000 samples, 23.69%) and nine attack types (161,043 samples combined, 76.31%), including a severe minority class (mitm, 1,043 samples, 0.49%). After merging all attack types into a single attack class, the binary label distribution is normal = 23.69% and attack = 76.31%, an imbalance ratio of 3.22:1.

A stratified 70/30 split produced 147,730 training and 63,313 testing samples. One-Hot Encoding (OHE) applied post-split expanded the feature space from 44 to 863 columns, while an alternative preprocessing path using OrdinalEncoder combined with StandardScaler retained a compact, dense representation of only 38 features. Both representations were carried forward into baseline training and adversarial evaluation to isolate the effect of feature representation on robustness.

B. Baseline Model Training Results

Table 8 summarizes the training results of Random Forest (RF) and XGBoost on the ToN-IoT dataset using the OHE representation. Both models achieved a Train Accuracy above 99.4% with a Standard Accuracy (SA) nearly identical to the training accuracy, producing an overfitting gap below 0.1% for both models, well within the very good range and indicating strong generalization rather than memorization. AUC-ROC values close to 1.0 (RF = 0.9991, XGBoost = 0.9999) confirm that both models separate the normal and attack classes almost perfectly.

Metric	Random Forest	XGBoost
Train Accuracy	99.47%	99.51%
Standard Accuracy (SA)	99.43%	99.44%
Overfitting Gap	0.04%	0.07%
AUC-ROC	0.9991	0.9999
Precision (Attack)	0.9948	0.9990
Recall (Attack)	0.9977	0.9936
F1-Score (Attack)	0.9963	0.9963

Table 8. Baseline Performance of RF and XGBoost on ToN-IoT (OHE)

Random Forest achieved a higher attack recall, detecting more true attacks, while XGBoost achieved a higher attack precision, producing fewer false alarms. Since missed attacks (false negatives) are generally more costly than false alarms in an NIDS context, RF's confusion-matrix behavior is examined further below.

Table 9 compares the confusion-matrix error rates of both models under the two preprocessing schemes on the full 63,313-sample test set. RF consistently achieved the lowest False Negative Rate (FNR) in both schemes, meaning fewer real attacks slipped through undetected, while XGBoost consistently achieved the lowest False Positive Rate (FPR), meaning fewer normal-traffic false alarms. Switching from

OHE to Ordinal Encoding plus StandardScaler improved both metrics for both models, most notably reducing RF's FNR from 0.23% to 0.07%.

Model	Encoding	FPR	FNR
RF	OHE	1.67%	0.23%
RF	Ordinal+SS	0.52%	0.07%
XGBoost	OHE	0.31%	0.64%
XGBoost	Ordinal+SS	0.31%	0.45%

Table 9. Confusion-Matrix Error Rates on the Full Test Set (attack = positive class)

C. Adversarial Attack Implementation Results

For both ZOO Attack and Genetic Adversarial Attack (GAA), 1,000 correctly classified attack samples were drawn via Stratified Shuffle Split (random_state = 42) separately for each model, so that only samples the baseline model could already detect were used to evaluate attack success.

1) ZOO Attack: In the targeted-attack scenario, forcing every attack sample toward the normal class, ZOO Attack failed completely under both preprocessing schemes: ASR = 0.00%, AA = 100.00%, and Average Perturbation (L-infinity) = 0.0000 for both RF and XGBoost, even after enlarging the optimization budget (learning_rate = 0.1, max_iter = 100, binary_search_steps = 10). This indicates that the zeroth-order gradient estimation used by ZOO cannot find a direction that shifts predictions across the piecewise-constant decision boundaries of tree ensembles when a specific target class must be reached.

The untargeted-attack scenario, which only requires any change in prediction, produced a markedly different result. Under Ordinal Encoding plus StandardScaler, XGBoost's Adversarial Accuracy dropped from near-100% to 20% (ASR = 80%, average perturbation L-infinity = 0.0364), whereas RF retained an Adversarial Accuracy of 95% (ASR = 5%, L-infinity is approximately 0.00005). Under OHE without normalization, both models remained almost fully resilient (ASR under 3%). Table 10 summarizes the untargeted-ZOO results.

Model	Encoding	ASR	AA
RF	OHE	0.00%	100.00%
RF	Ordinal+SS	5.00%	95.00%
XGBoost	OHE	0.50%	99.50%
XGBoost	Ordinal+SS	80.00%	20.00%

Table 10. ZOO Attack Results, Untargeted Scenario

This pattern is explained by the dimensionality and scale of the feature space: normalizing all features to a comparable scale makes ZOO's numerical gradient estimate far more stable and effective, while the high-dimensional, sparse, discrete space produced by OHE (862 binary features) makes it difficult for small perturbations to shift a tree ensemble's decision path.

2) Genetic Adversarial Attack (GAA): Because GAA relies on evolutionary search (selection, crossover, mutation) rather than gradients, it remained effective even in the discrete OHE feature space, but its success was highly dependent on both encoding scheme and model architecture. Table 11 reports the targeted-attack results, which are the focus of this experiment.

Model	Encoding	ASR	AA	Pert.
RF	OHE	0.00%	100.00%	0.183
RF	Ordinal+SS	99.90%	0.10%	0.099
XGBoost	OHE	100.00%	0.00%	0.100
XGBoost	Ordinal+SS	100.00%	0.00%	0.097

Table 11. Genetic Adversarial Attack Results, Targeted Scenario

RF's robustness collapsed almost entirely once the feature space changed from discrete OHE to continuous Ordinal plus StandardScaler (ASR 0% to 99.90%), while XGBoost remained fully vulnerable (ASR = 100%) under both schemes. This is the central finding of the adversarial evaluation: RF's apparent robustness against GAA under OHE is a property of the discrete, high-dimensional feature space, since continuous mutations rarely produce a valid one-hot vector capable of crossing RF's decision boundary, rather than an inherent property of the Random Forest algorithm itself. XGBoost's sequential boosting mechanism, in contrast, accumulates small per-tree contributions in a way that remains exploitable regardless of feature representation.

D. Feature Squeezing Defense Results

Feature Squeezing (per-feature clipping, bit-depth = 16) was evaluated across four attack-model combinations under both preprocessing schemes. Table 12 reports Robust Accuracy (RA) and Robust Accuracy Gain (RAG = RA minus AA); the SA Preservation Rate (SPR) remained above 96% in every configuration, confirming that the defense does not degrade clean-data performance or indicate label leakage.

Scenario	Encoding	AA	RA	RAG
FS+ZOO+RF	OHE	100.00%	99.50%	-0.50%
FS+ZOO+XGB	OHE	99.50%	98.50%	-1.00%
FS+GAA+RF	OHE	100.00%	99.90%	-0.10%
FS+GAA+XGB	OHE	0.00%	28.87%	+28.87%
FS+ZOO+RF	Ordinal+SS	86.50%	90.00%	+3.50%
FS+ZOO+XGB	Ordinal+SS	5.00%	26.00%	+21.00%
FS+GAA+RF	Ordinal+SS	0.10%	3.00%	+2.90%
FS+GAA+XGB	Ordinal+SS	0.00%	0.40%	+0.40%

Table 12. Feature Squeezing Defense Results (SPR at least 96% in all scenarios)

Feature Squeezing was only meaningfully effective (RAG at least 10%) in two scenarios: FS+GAA+XGBoost under OHE (RAG = 28.87%) and FS+ZOO+XGBoost under Ordinal plus StandardScaler (RAG = 21.00%). Both successful cases involved XGBoost, whose sharper, more precision-dependent decision boundaries make its adversarial examples easier to push back across the boundary through bit-depth quantization. For RF, adversarial accuracy was already near 100% before defense in most scenarios, leaving little room for the defense to demonstrate improvement; in the one scenario where RF was substantially compromised (GAA under Ordinal plus StandardScaler), Feature Squeezing recovered only 2.90 percentage points, indicating that continuous, low-magnitude perturbations in a dense feature space are difficult to detect or remove through simple quantization.

E. Comparative Analysis: Random Forest vs. XGBoost

e-ISSN: 2548-6861

Table 13 consolidates the comparison across all experiments. RF and XGBoost achieve comparable baseline accuracy, but diverge sharply in adversarial robustness: RF is more resistant to both ZOO Attack and Genetic Adversarial Attack overall, particularly under the discrete OHE representation, while XGBoost is consistently more vulnerable to GAA regardless of encoding scheme and more vulnerable to untargeted ZOO Attack once features are normalized.

Aspect	Random Forest	XGBoost
Standard Accuracy	99.43-99.82%	99.44-99.58%
ZOO (targeted), both encodings	ASR = 0%	ASR = 0%
ZOO (untargeted, Ordinal+SS)	ASR = 5%	ASR = 80%
GAA (targeted, OHE)	ASR = 0%	ASR = 100%
GAA (targeted, Ordinal+SS)	ASR = 99.90%	ASR = 100%
Best FS recovery (RAG)	+3.50%	+28.87%

Table 13. Summary Comparison of RF and XGBoost Across All Experiments

Two conclusions follow from this comparison. First, resilience to gradient-estimation-based attacks such as ZOO under a targeted objective is a general property of tree ensembles, shared by both bagging (RF) and boosting (XGBoost) architectures, rather than a trait unique to either algorithm. Second, feature representation is at least as influential as model architecture in determining adversarial robustness: RF's robustness against Genetic Adversarial Attack is highly conditional on the feature space being discrete and high-dimensional (OHE), whereas XGBoost's sequential boosting mechanism remains exploitable under any representation tested. These results indicate that securing tree-based NIDS requires attention to both the choice of classification algorithm and the design of the feature-encoding pipeline, since the latter can change a model's measured robustness by up to two orders of magnitude in Attack Success Rate.

IV. CONCLUSION

This study evaluated the robustness of Random Forest (RF) and XGBoost against Zeroth-Order Optimization (ZOO) Attack and Genetic Adversarial Attack (GAA), together with a Feature Squeezing defense, on the ToN-IoT dataset. Based on the implementation, testing, and analysis carried out, the following conclusions and recommendations are drawn.

A. Conclusions

1) Data preprocessing has a substantial effect on adversarial robustness. One-Hot Encoding (OHE) without normalization produces a more discrete, high-dimensional feature representation that is harder for attack algorithms to exploit, whereas OrdinalEncoder combined with StandardScaler produces a continuous feature space that is easier to optimize against.
2) The targeted variant of ZOO Attack failed to generate any adversarial example across all test scenarios. Even after enlarging the optimization budget (learning rate, max iterations, binary search steps), the attack could not produce a perturbation or change any prediction, indicating that its objective function is substantially harder to optimize when samples must be forced into one specific target class.
3) The untargeted variant of ZOO Attack performed better than its targeted counterpart. Under OrdinalEncoder + StandardScaler preprocessing, it achieved an Attack Success Rate (ASR) of 80% against XGBoost, versus only 5% against Random Forest, confirming that an untargeted objective, which only requires misclassification rather than a specific target class, is inherently easier to optimize.
4) Genetic Adversarial Attack (GAA) achieved a substantially higher success rate than ZOO Attack. Under OrdinalEncoder + StandardScaler preprocessing, GAA reached an ASR near 100% against both models, showing that evolutionary optimization can effectively explore the feature space even without gradient information.
5) Under OHE without normalization, Random Forest showed strong resistance to GAA (ASR = 0%), while XGBoost was fully compromised (ASR = 100%). This indicates that RF's bagging-based ensemble structure is more stable against perturbations than XGBoost's boosting mechanism within a one-hot feature space.
6) Feature Squeezing preserved clean-data accuracy well, with a Sample Preservation Rate (SPR) above 96% in every scenario, but its effectiveness at reducing attack success was limited. A Robust Accuracy Gain (RAG) meeting the target threshold was achieved only for Feature Squeezing against ZOO Attack on XGBoost after StandardScaler preprocessing, and for Feature Squeezing against GAA on XGBoost after OHE preprocessing.
7) Overall, Random Forest was more resilient than XGBoost across the tested adversarial scenarios. The combination of Random Forest with OHE (no normalization) was the most robust configuration observed, consistently producing the lowest attack success rates against both ZOO Attack and Genetic Adversarial Attack.

B. Recommendations

1) Future work should evaluate additional defense mechanisms, such as Adversarial Training, Defensive Distillation, Randomized Smoothing, other input-transformation techniques, or a defense-in-depth combination of methods, and compare their effectiveness against Feature Squeezing.
2) Other classification models, such as LightGBM, CatBoost, Deep Neural Networks, or Transformer-based IDS, should be tested to obtain a more comprehensive picture of adversarial robustness across algorithm families.
3) Adversarial evaluation should be extended with additional attack methods, such as HopSkipJump Attack, Boundary Attack, Square Attack, SimBA, or NES Attack, to more thoroughly characterize each model's vulnerability profile.

4) Future studies should incorporate additional IDS datasets, such as UNSW-NB15, CICIDS2017, CSE-CIC-IDS2018, or Bot-IoT, to improve the generalizability of the findings.
5) From a systems perspective, One-Hot Encoding can be considered a practical strategy for increasing adversarial robustness, since it produces a feature space that is harder for attackers to exploit; however, the resulting increase in feature dimensionality should be weighed against memory and computation costs.
6) The use of StandardScaler should be considered carefully relative to model characteristics and deployment goals. While normalization improves training stability and often improves classification performance, this study shows it can also make optimization easier for certain adversarial attacks, particularly gradient-based and evolutionary-based methods.
7) For real-world deployment, defense should not rely on a single mechanism such as Feature Squeezing alone, but should adopt a multi-layer defense approach combining preprocessing choices, adversarial training, adversarial detection, ensemble modeling, and input validation, to strengthen resilience against both gradient-based attacks such as ZOO and non-gradient-based attacks such as Genetic Adversarial Attack.

ACKNOWLEDGMENT

The authors would like to thank Politeknik Negeri Batam for providing support and facilities that contributed to the completion of this research. The authors also would like to thank Mr. Andy Triwinarko for his valuable guidance, suggestions, and support during the completion of this research.

REFERENCES

[1] S. Alkadi, S. Al-Ahmadi, and M. M. Ben Ismail, "RobEns: Robust Ensemble Adversarial Machine Learning Framework for Securing IoT Traffic," *Sensors*, vol. 24, no. 8, p. 2626, Apr. 2024, doi: 10.3390/s24082626.

[2] "Cybercrime to cost the world $10.5 trillion annually by 2025," Cybersecurity Ventures, 2021. [Online]. Available: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

[3] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A survey on intrusion detection system in IoT networks," *Cyber Secur. Appl.*, vol. 3, p. 100082, Dec. 2025, doi: 10.1016/j.csa.2024.100082.

[4] V. R. Joshi, K. Assa-Agyei, T. Al-Hadhrami, and S. N. Qasem, "Hybrid AI Intrusion Detection: Balancing Accuracy and Efficiency," *Sensors*, vol. 25, no. 24, p. 7564, Dec. 2025, doi: 10.3390/s25247564.

[5] K. He, D. D. Kim, and M. R. Asghar, "Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 1, pp. 538–566, 2023, doi: 10.1109/COMST.2022.3233793.

[6] S. T. Hamidou and A. Mehdi, "Enhancing IDS performance through a comparative analysis of Random Forest, XGBoost, and Deep Neural Networks," *Mach. Learn. Appl.*, vol. 22, p. 100738, Dec. 2025, doi: 10.1016/j.mlwa.2025.100738.

[7] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e4150, Jan. 2021, doi: 10.1002/ett.4150.

[8] Y. Chen, S. Wang, W. Jiang, A. Cidon, and S. Jana, "Cost-Aware Robust Tree Ensembles for Security Applications".

[9] D. Vos and S. Verwer, "Efficient Training of Robust Decision Trees Against Adversarial Examples".

[10] M. Rajhans and V. Khawarey, "Beyond Gradient-Based Attacks: Adversarial Robustness and Explainability Stability in Cybersecurity Classifiers," Jul. 03, 2026, *arXiv*. doi: 10.48550/ARXIV.2607.01679.

[11] S. Dyrmishi, M. Djilani, T. Simonetto, S. Ghamizi, and M. Cordy, "Insights on Adversarial Attacks for Tabular Machine Learning via a Systematic Literature Review," 2025, *arXiv*. doi: 10.48550/ARXIV.2506.15506.

[12] "Adversarial Robustness Toolbox." [Online]. Available: https://adversarial-robustness-toolbox.readthedocs.io/en/latest/

[13] T. T. Nguyen, U. H. Tran, and H. N. Nguyen, "Adversarial attack and defense in AI-powered intrusion detection," *J. Comput. Sci. Cybern.*, vol. 41, no. 4, pp. 387–402, Nov. 2025, doi: 10.15625/1813-9663/22884.