A Secure Adversarial Attack Detector Framework for Monitoring Network Intrusion in IoT Environment

Priyanka Verma, *Senior Member, IEEE*, Ankit Vidyarthi, Anand Mishra, Jabir Ali, Nitesh Bharot, *Senior Member, IEEE*, and John G. Breslin, *Senior Member, IEEE*

***Abstract*—As computer networks expand, the demand for robust security measures intensifies due to increasing threats that jeopardize network integrity and confidentiality. This is particularly critical in the context of the Internet of Things (IoT), where interconnected devices significantly broaden the attack surface. Despite the growing adoption of Intrusion Detection Systems (IDS) enhanced with machine learning (ML), these systems face a critical vulnerability: they can be easily deceived by adversarial attacks specifically crafted to evade detection. This paper addresses the problem of adversarial robustness in ML based IDS, particularly under white-box attack scenarios, by introducing the Adversarial Attack Detector (AAD) framework. AAD incorporates a multi-layered defense mechanism, including a request validator, a machine learning-based Adversarial Discriminator (AD), and an Enhanced Intrusion Detection System (EIDS). The EIDS combines adversarially trained Multi-Layer Perceptrons, Convolutional Neural Networks, and Long Short-Term Memory networks to improve detection accuracy and robustness. The scope of this study is centered on securing IoT environments, where conventional IDS models struggle due to the limitations of constrained devices and heterogeneous data. Our empirical analysis shows that the AAD framework significantly mitigates the impact of various white-box adversarial attack methods, and outperforms traditional and single-model approaches. AAD's components, particularly the EIDS, significantly recover model performance under adversarial settings, improving accuracy from near-failure levels to near-perfect classification in some scenarios.**

***Index Terms*—Intrusion detection, NIDS, adversarial, FGSM, PGD, BIM, DeepFool, white box attack, ensemble learning.**

Received 6 July 2025; revised 30 July 2025 and 10 September 2025; accepted 23 September 2025. Date of publication 26 September 2025; date of current version 8 December 2025. This work was supported in part by the Taighde Éireann - Research Ireland under Grant 12/RC/2289_P2 (Insight) and Grant 21/FFP-A/9174 (SustAIn), and in part by Interreg Atlantic Area co-funded by the European Union under Grant EAPA_0016/2022 (ENEPORTS). *(Corresponding author: Priyanka Verma.)*

Priyanka Verma is with the School of Computer Science, University of Galway, Galway, H91 TK33 Ireland (e-mail: priyanka.verma@universityofgalway.ie).

Ankit Vidyarthi is with the Department of CSE&IT, Jaypee Institute of Information Technology, Noida 201309, India (e-mail: ankit.vidyarthi@jiit.ac.in).

Anand Mishra is with the Department of CSE, NIIT University, Neemrana 301705, India (e-mail: anandr.mishra13@gmail.com).

Jabir Ali is with the Department of CSE, Bennett University, Greater Noida 201310, India (e-mail: jabir.ali@bennett.edu.in).

Nitesh Bharot and John G. Breslin are with the Data Science Institute, University of Galway, Galway, H91 TK33 Ireland (e-mail: nitesh.bharot@niversityofgalway.ie; john.breslin@universityofgalway.ie).

Digital Object Identifier 10.1109/TCE.2025.3614806

I. Introduction

As COMPUTER networks continue to grow in size and complexity, the demand for effective security measures becomes increasingly critical [1], [2]. With the expansion of these networks, the number of connected devices rises exponentially, leading to heightened security threats [3]. This growth is further accelerated by the proliferation of Internet of Things (IoT) devices, which introduce unique security challenges due to their resource constraints, heterogeneity, and often limited built-in protection mechanisms. The availability, integrity, and confidentiality of network assets are seriously jeopardised by these attacks. Global cybercrime costs are projected to reach $10.5 trillion annually by 2025, underscoring the urgent need for enhanced cybersecurity measures [4]. Consequently, safeguarding networks and information systems from potential attacks is paramount. Intrusion Detection Systems (IDS) [5] play an increasingly vital role in IoT ecosystems by monitoring network traffic and identifying potential threats such as malware, network intrusions, and denial-of-service attacks.

The rapid advancement of hardware technology, particularly the advent of high-performance GPUs, has spurred researchers to explore Machine Learning (ML) techniques [6], [7], [8] for these IDS. However, adapting ML-based IDS to large network architectures and complex conditions can be challenging and costly. In contrast, Deep Learning (DL) based IDS, with its multiple hidden layers, can extract deeper network features and has become a focal point for researchers. DL models now dominate the field [9], [10], accounting for over 80% of IDS research [11], thanks to their superior performance.

Despite their advantages, Deep Neural Networks (DNNs) are vulnerable to adversarial attacks [12], [13] and backdoor attacks, where subtle input perturbations can lead to incorrect inferences. Szegedy et al. [14] showed that these adversarial manipulations are often hard to detect, posing risks to DL-based Network Intrusion Detection Systems (NIDS). Attackers can produce adversarial instances by altering small subsets of network features and making requests to the NIDS, adjusting their perturbations based on feedback until the NIDS is bypassed, and remains undetected by the NIDS systems.

These adversarial attacks can be classified into three categories based on the attacker's knowledge: black box [15], white box [16], and grey box [17]. Grey-box attacks need a basic grasp of the structure of the target model as well as

previous knowledge of the training data. Typically, attackers treat a NIDS as a black-box system, where internal workings are hidden, and behavior is learned through queries and feedback.

In white-box attacks, adversaries possess full insight into the model's structure, including its architecture, parameters, and the data used for training. This level of access allows attackers to craft highly effective adversarial examples that can bypass even state-of-the-art NIDS, creating serious security vulnerabilities. This scenario represents the primary threat model we investigate in our research. Techniques such as Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), Basic Iterative Method, DeepFool, and Carlini & Wagner (CW) attacks exploit this knowledge to craft adversarial examples that deceive the DL model. Wang and Wang [18] conducted a feature-level attack (FLA) on a Multi-Layer Perceptron (MLP) model using a white-box approach. Such methods allow malicious flows to mimic benign traffic, evading detection by even the most accurate NIDS.

We present a novel methodology called the Adversarial Attack Detector (AAD) for addressing the problem of white-box attacks, particularly those involving internal actors in IoT environment. The contributions of our paper are summarized as:

- A key contribution of this work is the comprehensive and systematic analysis of white-box adversarial attacks on NIDS by employing a diverse set of state-of-the-art attack methods, including FGSM, PGD, DeepFool, and BIM. This extensive exploration enables the generation of varied adversarial examples, fostering a robust evaluation and training process capable of addressing a wide spectrum of attack strategies.
- The paper introduces a novel Adversarial-Aware Detection (AAD) framework designed to strengthen the robustness of NIDS under white-box attack scenarios. The AAD framework employs a multi-layered architecture that integrates three key components: a validator for initial input screening, an adversarial discriminator for detecting subtle perturbations, and an Enhanced Intrusion Detection System (EIDS) trained with adversarial data. This integrated design provides a comprehensive and adaptive defense mechanism.
- A significant contribution lies in the development of the Enhanced Intrusion Detection System (EIDS), which not only counters adversarial attacks but also operates as a full-featured NIDS. By applying a logical OR operation across detection outputs, EIDS ensures high sensitivity to threats, thereby improving overall detection accuracy and robustness against adversarial manipulations.

While prior research on adversarial robustness in Network Intrusion Detection Systems (NIDS) has primarily focused on adversarial training, ensemble models, or detection-based countermeasures in isolation, this work introduces a unified and multi-layered defense strategy. Unlike standard adversarial training approaches that often rely on a single attack method or model architecture, the proposed framework incorporates a wide range of white-box attack techniques (FGSM, PGD, DeepFool, and BIM) to generate diverse adversarial examples, improving generalization across threats. In contrast to traditional ensemble-based systems that typically use majority voting, the Enhanced Intrusion Detection System (EIDS) employs a logical OR aggregation to boost detection sensitivity. Moreover, the inclusion of a dedicated validator and an adversarial discriminator adds additional layers of defense that are often absent in prior models, enabling early rejection and fine-grained perturbation analysis. This holistic design offers stronger resistance against adaptive attackers and outperforms conventional defenses by combining multiple complementary components in a coordinated framework.

II. Related Work

There exists various strategies to defend DL systems from adversarial instances. Commonly used methods include Network Distillation [19], Adversarial Training (AT) [20], Adversarial Detecting [21], Input Reconstruction [22], Classifier Robustifying [23], Network Verification [24], and an ensemble of them, which work either reactively or proactively. To tackle white-box adversarial attacks, in particular, several defense mechanisms have been proposed. AT, in which the model training is done on a mixture of original and adversarial examples, is one of the most effective strategies. This approach aids the system in recognizing and resisting adversarial perturbations. Another method is gradient masking [25], which involves obscuring the gradient information to make it harder for attackers to calculate effective perturbations. However, this technique may be misleading as attackers may find alternative ways for bypassing the defense. Other approaches include defensive distillation, which smooths the model's decision boundaries, and the use of robust optimization techniques aiming to increase the system's overall robustness.

For DL models, server breaches are especially dangerous. A compromised model allows attackers to gain white-box access, enabling them to craft adversarial examples against which there are no effective defenses. This scenario is particularly devastating for organizations that have invested substantial time and resources into developing proprietary models. Shan et al. [26] addressed the problem of post-breach recovery for DNN models. Shan et al. introduced Neo, a novel system generating new versions of compromised models and includes an inference-time filter to remove adversarial instances created using existing compromised models. Neo detects the overfitting of attacks to the compromised model used in their development by gently offsetting the classification surfaces of several model versions using latent distributions.

Gungor et al. [27] proposed a stacking ensemble learning framework that is more resilient against adversarial attacks than single DL methods. It employed four of attack methods and ten different DL models. The results suggest that models based on RNNs demonstrate greater robustness, while CNN architectures show high vulnerability to attacks The most robust individual ML method varies depending upon dataset or attack methodology. Javeed et al. [5] presents an explainable and resilient IDS specifically designed for Industry 5.0. The proposed system integrates advanced neural network

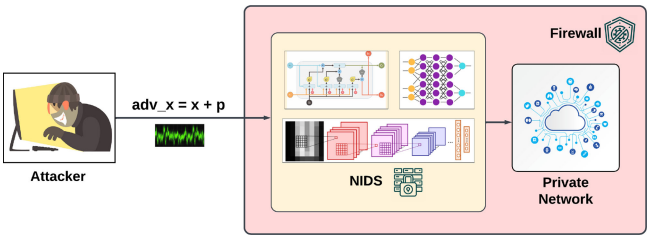

Fig. 1. Launching an adversarial attack.

architectures, including BiLSTM and Bi-GRU, along with fully connected layers and a softmax classifier.

Wu et al. [28] uncovered two key insights: (1) the presence of larger generalization gaps at hidden layers and (2) the potential effectiveness of minimizing these gaps to mitigate white-box Membership Inference Attack (MIA). Based on these insights, it proposed a novel defense method called Nirvana. Nirvana selects a hidden layer with significant generalization gaps and employs a multi-sample convex combination of features during training. This approach fortifies the model against attacks seeking to infer membership status from the model's responses. Experimental results on CIFAR100, Purchase100, and Texas datasets demonstrated a balanced trade-off between utility and privacy.

Alhussien et al. [29] proposed a novel set of domain constraints for network traffic that preserve the statistical and semantic relationships between traffic features while ensuring the validity of the perturbed adversarial traffic. Four types of constraints are identified for it: distribution-preserving constraints, feature mutability and value constraints, and feature dependence constraints. Using two intrusion detection datasets, Alhussien et al. assessed the effects of them on white-box and black-box attacks.

Zhou et al. [30] proposed a generative adversarial network (GAN)-based Siamese neuron network (GSNN) to defend against white-box adversarial attacks in modulation classification. In this, a generator (G) is trained to create perturbations which could deceive the discriminator (D), while the D aims for correctly classifying original and adversarial instancess generated by the G. The D, implemented as a Siamese network, transforms the classification problem into a pairwise comparison task. Upon completing the GSNN training, it used a distance-based nearest class mean (NCM) classifier to perform the classification task.

III. Preliminaries

Adversarial attacks provide tiny, sometimes unnoticeable modifications to the input data in order to take advantage of flaws in DL models. With healthcare, these perturbations may lead to incorrect findings from the model. The repercussions of such attacks are vast, affecting fields ranging from autonomous driving to healthcare. Developing strong defences requires an understanding of attacks from adversaries and their varied dynamics. Fig. 1 gives the general scenario of launching adversarial attacks.

Adversarial attacks could be divided based on their objectives: In targeted attacks the attacker aims for misleading the model to classify the input as a desired, wrong class. This requires more precise perturbations but can be more damaging if the targeted class has significant implications whereas in non-targeted attacks the goal is to cause any incorrect classification. These attacks are generally easier to perform since any deviation from the correct class is considered successful. Evasion attacks [31] occur during the model's deployment phase. The attacker creates adversarial examples that bypass the model's defenses and evade detection. These attacks are particularly concerning for real-time systems like biometric authentication [32] and autonomous vehicles [33]. Unlike evasion attacks, poisoning attacks [34] happen during the training phase. Attackers introduce malicious data in training data, leading the model in learning undesired patterns. This can lead to a degraded performance or specific vulnerabilities in the deployed model.

A. FGSM Method

FGSM is a technique used in generating adversarial instances, that are slightly perturbed inputs built for fooling a DL model in making undesired predictions. It takes advantage of gradient of the loss function with respect to the input data in creating these perturbations.

Given a DL model with parameters p, an original input a, and a true label b, the objective is to create an adversarial example adv_a which is near to a but causes the model in misclassifying it.

Initially, the gradient of the loss function $J(P, a, b)$ with respect to the input a is calculated. This gradient indicates how the loss changes with small changes in the input data. Mathematically, this is represented as:

$$\nabla_a J(P, a, b) \tag{1}$$

The gradient $\nabla_a J(P, a, b)$ moves in the direction of the steepest ascent of the loss function. Considering the sign of ∇_a, FGSM determines the direction in which to modify every part of input for increasing loss.

Using 1, the direction for perturbing the input a is determined by the sign of the computed gradient.

$$p = c \cdot \text{sign}(\nabla_a J(P, a, b)) \tag{2}$$

where: c is scalar value which controls the amount of the perturbation. sign$(\cdot)$ is the element-wise sign function that outputs $+1$, -1, or 0.

Small value c ensures that the perturbation is subtle, often unrecognizable to humans, yet sufficient in causing neural network to make a mistake.

Finally, the perturbation p is added to a to generate the adversarial instance adv_x:

$$adv_a = a + p = a + c \cdot \text{sign}(\nabla_a J(P, a, b)) \tag{3}$$

B. BIM Method

BIM is an extension of the FGSM that generates adversarial examples through multiple iterations of small perturbations. Using FGSM iteratively with a smaller step size, BIM refines adversarial perturbation, making it more effective at fooling neural networks.

At the beginning, the adversarial instance adv_a is initialized same as the original input a:

$$adv_a^{(0)} = a \tag{4}$$

For each iteration itr from 1 to I (the total number of iterations), adv_a is updated using FGSM with a small step size α:

$$adv_a^{(itr)} = \text{clip}_{a,\epsilon}\left(adv_a^{(itr-1)} + \alpha \cdot \text{sign}\left(\nabla_{adv_a} J(P, adv_a^{(itr-1)}, b)\right)\right) \tag{5}$$

where: $adv_a^{(itr)}$ is the adversarial example at iteration itr. $\nabla_{adv_a} J(P, adv_a^{(itr-1)}, b)$ is the gradient of the loss function with respect to $adv_a^{(itr-1)}$. α is the step size, a small scalar value able to control the amount of each perturbation. $\text{clip}_{a,\epsilon}(\cdot)$ ensures that the perturbed input remains within an ϵ-ball around the original input a, to maintain the perturbation's imperceptibility:

$$\text{clip}_{a,\epsilon}(adv_a) = \min(\max(adv_a, a-\epsilon), a+\epsilon) \tag{6}$$

The clipping function 6 ensures that the total perturbation across all iterations does not exceed the specified ϵ-ball around a, keeping perturbation subtle and imperceptible.

C. PGD Method

PGD is powerful and widely applied method for generating adversarial examples. It extends the BIM by incorporating projection for ensuring that the perturbations stay into a specified norm bound. PGD is often considered the strongest first-order adversary in AT contexts.

Given a DL model with parameters P, an original input a, a true label b, and a perturbation bound ϵ, the goal of PGD is finding an adversarial instance adv_a which maximizes the loss function while ensuring that the perturbation stays within the ϵ-ball around a. Finally the adversarial example adv_a to the a with a random perturbation is initialized:

$$adv_a^{(0)} = a + p \quad \text{where} \quad \|p\|_\infty \leq \epsilon \tag{7}$$

For each iteration itr from 1 to I (the total number of iterations):

the gradient of the loss function $J(P, adv_a^{(itr-1)}, b)$ in respect to the adversarial example $adv_a^{(itr-1)}$ is calculated:

$$g^{(itr)} = \nabla_{adv_a} J\left(P, adv_a^{(itr-1)}, b\right) \tag{8}$$

the adversarial instance is updated by taking a step in direction of the gradient:

$$adv_a^{(itr)} = adv_a^{(itr-1)} + \alpha \cdot \text{sign}\left(g^{(itr)}\right) \tag{9}$$

At the end, the updated adversarial example is projected back into the ϵ-ball around the original input:

$$adv_a^{(itr)} = \text{clip}_{a,\epsilon}\left(adv_a^{(itr)}\right) = \min\left(\max(adv_a^{(itr)}, a-\epsilon), a+\epsilon\right) \tag{10}$$

Here: α is the step size. $\nabla_{adv_a} J(P, adv_a^{(itr-1)}, b)$ is the gradient of the loss function with respect to $adv_a^{(itr-1)}$. $\text{sign}(\cdot)$ is the element-wise sign function.

The projection step ensures that the total perturbation remains within the ϵ-ball, maintaining the perturbation's imperceptibility.

The final adversarial example $adv_a = adv_a^{(I)}$ is obtained after I iterations.

D. DeepFool Method

DeepFool is designed for finding the minimal perturbation needed for changing the classification of an input. It iteratively linearizes classifier's decision boundary and finds the smallest perturbation that can push the input across this boundary.

Given a binary classifier B with parameters P, an original input a, and a true label $b \in \{0, 1\}$, DeepFool aims finding tiny perturbation p such that the perturbed input $adv_a = a + p$ is misclassified. Assume the decision boundary is defined by $B(a) = 0$.

Initially the adversarial example $adv_a^{(0)}$ is assigned to be the original input a:

$$adv_a^{(0)} = a \tag{11}$$

For each iteration itr, until the classifier's decision changes:

The gradient of the classifier is computed with respect to the input:

$$\nabla B\left(adv_a^{(itr)}\right) = \left.\frac{\partial B}{\partial a}\right|_{adv_a^{(itr)}} \tag{12}$$

The perturbation needed to cross the linearized decision boundary is computed:

$$p^{(itr)} = -\frac{B\left(adv_a^{(itr)}\right)}{\|\nabla B\left(adv_a^{(itr)}\right)\|^2} \cdot \nabla B\left(adv_a^{(itr)}\right) \tag{13}$$

Finally, the updated adversarial example is given as:

$$adv_a^{(itr+1)} = adv_a^{(itr)} + p^{(itr)} \tag{14}$$

At end, it is check if the classifier's decision changes:

$$\text{if } B\left(adv_a^{(itr+1)}\right) \neq B(x), \text{ stop.} \tag{15}$$

The final adversarial example $adv_a = adv_a^{(itr+1)}$ is obtained after the decision boundary is crossed.

IV. Proposed Framework

In this section, we introduce Adversarial Attack Detector (AAD), as shown in Fig. 2. AAD is designed to be highly resilient against adversarial attacks and enhances intrusion detection capabilities. Unlike conventional IDS), which are often vulnerable to adversarial manipulations, AAD provides superior protection and reliability in identifying and mitigating such threats.

The primary advantage of AAD lies in its dual functionality. Firstly, it serves as a robust defense against adversarial attacks, which are intentional attempts by attackers to falsify information in order to get around security mechanisms. Secondly, AAD contributes significantly to intrusion detection. Traditional IDS frameworks often struggle with accurately detecting and responding to intrusions when adversarial attacks are involved. However, AAD detects a broader range of malicious activities.

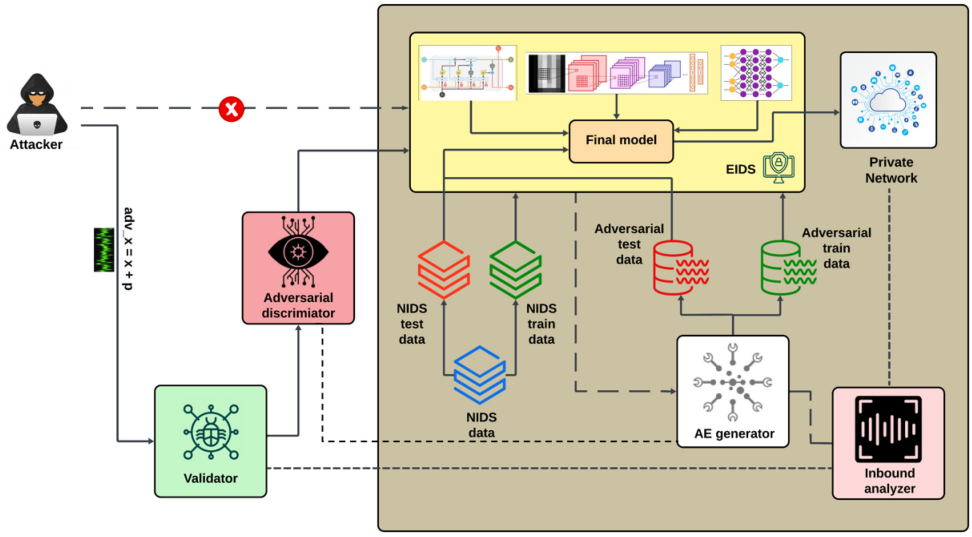

Fig. 2. Proposed AAD methodology.

Initially, instead of routing the traffic directly from the sender to the traditional IDS, it is first passed through a validator. This validator has a set of predefined rules for incoming traffic, which are generated using an inbound analyzer. The validator uses these rules to assess the authenticity & legitimacy of the incoming traffic. If the traffic meets the predefined criteria, it is then forwarded to the adversarial discriminator.

The adversarial discriminator is responsible for identifying potential adversarial attacks. If the traffic passes through the adversarial discriminator, it is then sent to the EIDS, which is an adversarially trained ensembled IDS. The EIDS not only checks for adversarial attacks but also identifies any other types of intrusions. Once the traffic clears this stage, it is allowed into the private network of the organization or user. This multi-layered approach ensures robust protection against both adversarial and traditional attacks, enhancing the overall security of the network. There are several components to AAD, which are covered in more detail as follows.

A. Validator

The validator functions as the initial bar of defense against undesired traffic. It operates on a set of predefined rules derived from the knowledge of an inbound analyzer. The inbound analyzer meticulously examines incoming traffic and defines specific rules for each feature of the traffic. These regulations set the standard for what constitutes appropriate and typical traffic. When traffic arrives, the validator compares it against the predefined rules. If any aspect of the traffic falls outside the established parameters, the validator rejects it as an attack. However, if the traffic complies with the guidelines, it moves on to the adversarial discriminator for further scrutiny. This process ensures that only traffic deemed legitimate by the validator proceeds, effectively filtering out potentially harmful data at an early stage. Thus validator acts as a preliminary filter for incoming traffic and applies a set of lightweight statistical and heuristic checks to identify anomalous patterns that may indicate adversarial manipulation. For example, it flags inputs with unusual feature distributions (e.g., excessive feature perturbation beyond a defined threshold or sudden shifts in traffic volume). However, it should be noted that a validator just does the initial check, which may be easier to surpass by white box attacks.

B. Adversarial Discriminator (AD)

AD consists of an RF classifier that uses ensemble learning to build multiple decision trees during training. The final classification is determined by the most common output among these individual trees. One of the key elements of constructing decision trees within an RF is the splitting criterion, which can be based on various metrics such as the one we used, named Gini impurity.

For a node t with K classes, the Gini impurity $GIP(t)$ is:

$$GIP(t) = 1 - \sum_{i=1}^{K} r_i^2 \tag{16}$$

r_i is the proportion of instances of class i in the node t.

In a base tree, the goal is splitting the data at each node in a way that the emerging child nodes have lower impurity than the parent node. For a split S that divides a node t into two child nodes t_L (left) and t_R (right), the Gini gain (reduction in impurity) from the split is:

$$\Delta GIP = GIP(t) - \left(\frac{T_L}{T} GIP(t_L) + \frac{T_R}{T} GIP(t_R) \right) \tag{17}$$

where: $GIP(t)$ is the Gini impurity of the parent node. $GIP(t_L)$ and $GIP(t_R)$ are the Gini impurities of the left and right child nodes, respectively. T is the total number of instances in the parent node. T_L and T_R are the numbers of instances in the left and right child nodes, respectively.

C. AE Generator

Adversarial Example (AE) generator is a pivotal element of the AAD methodology, tasked with creating adversarial

samples to train both the AD and the EIDS. This generator employs several advanced techniques to produce these samples, including the FGSM, PGD, DeepFool, and the BIM. These techniques are all meant to quietly alter input data in a way that skews ML models' predictions, thus simulating the kinds of adversarial attacks that the AAD system must defend against.

By utilizing a variety of adversarial attack methods, the AE generator ensures a diverse and comprehensive set of training examples. This variety is crucial for training the AD and the EIDS to recognize and mitigate different types of adversarial attacks. This comprehensive training approach makes the AAD system is effective in maintaining network security against sophisticated adversarial threats.

D. Inbound Analyzer

The inbound analyzer continuously monitors the traffic entering and leaving the private network. By analyzing this traffic, the inbound analyzer gathers vital information on normal and potentially malicious activity patterns. This information is essential for the AE generator and the validator, as it helps in defining the rules and characteristics for identifying legitimate traffic and generating realistic adversarial examples. The validator uses these insights to establish a baseline of acceptable traffic, ensuring that only genuine data passes through. Meanwhile, the AE generator leverages this information to create adversarial samples that mimic real-world attacks, thereby improving the training and effectiveness of the AD and the EIDS.

E. Enhanced Intrusion Detection System (EIDS)

The EIDS is a sophisticated component of the AAD that leverages an ensemble of ML models to provide robust security against adversarial attacks. This ensemble model combines three distinct types of neural networks: MLP, CNN, and LSTM. Each of these models has unique strengths, making the ensemble highly effective to detect and mitigate a broad set of cyber threats. The MLP excels at handling structured data, the CNN is adept at recognizing patterns in traffic data, and the LSTM is particularly suited for sequential data, capturing temporal dependencies in network traffic.

The convolutional layer is the foundational component of a CNN model, functioning by applying filters to the input to obtain its features. Every filter, also known as kernel, slides over the input, doing a convolution operation. This involves producing the dot product between the kernel's weights and the corresponding features of the input. The result is a feature map, which can be mathematically described by the following equation:

$$z_{i,j}^{k} = \sum_{c=1}^{C} \sum_{m=1}^{M} \sum_{n=1}^{N} w_{m,n,c}^{k} \cdot a_{i+m,j+n,c} + \sigma^{k} \tag{18}$$

In this equation: $z_{i,j}^{k}$ represents the value at the (i,j)-th position of the k-th feature map. $a_{i+m,j+n,c}$ is the value at position $(i+m, j+n)$ of the c-th input channel. $w_{m,n,c}^{k}$ denotes the weights of the filter for the k-th feature map, with m and n indicating the filter dimensions, and c representing the input channel. σ^{k} is the bias term for the k-th feature map.

LSTM layers are a type of RNN layer built to model temporal sequences and long-range dependencies more effectively than standard RNNs. LSTMs achieve this by using a unique structure including memory cells and gates.

Dense layers, also known as fully connected layers are fundamental components of neural networks where each neuron is connected to every neuron in the previous and subsequent layers. These layers are typically used for tasks like classification and regression.

For a dense layer with input a, weights W, bias σ, and activation function ϕ, the output b is computed as follows:

$$b = \phi(W \cdot a + b) \tag{19}$$

Here: a is the input vector. W is the weight matrix. σ is the bias vector. ϕ is the activation function, such as ReLU, sigmoid, or softmax.

If the input has n dimensions and the output has m dimensions, then: W is an $m \times n$ matrix. σ is an m-dimensional vector.

Mathematically, the ensemble model can be represented as a combination of the predictions from the MLP, CNN, and LSTM models. Let $f_{MLP}(a)$, $f_{CNN}(a)$, and $f_{LSTM}(a)$ denote the outputs of the MLP, CNN, and LSTM models respectively for an input a. The final prediction $\hat{y}$ of the ensemble model is determined by a logical OR operation across the individual model predictions. This means that if any one of the models predicts the input as an attack, the ensemble system will classify it as an attack. Formally, this can be expressed as:

$$\hat{y} = f_{MLP}(a) \vee f_{CNN}(a) \vee f_{LSTM}(a) \tag{20}$$

where $\hat{y}$ is the final output, and $\vee$ represents the logical OR operation. This approach ensures a high sensitivity to attacks, as the system errs on the side of caution by flagging any suspicious activity detected by any of the models.

The EIDS operates as both a NIDS and a defense mechanism against adversarial attacks. By training the ensemble model on adversarial examples generated by the AE generator, the EIDS learns to identify and counteract sophisticated attack strategies. This training involves minimizing a loss function J that accounts for both the accuracy of detecting standard intrusions and the ability to recognize adversarial manipulations. The robustness of the EIDS is enhanced by the diversity of its constituent models, as each model contributes its strengths to the ensemble, ensuring comprehensive coverage of different attack vectors.

Overall, the AAD provides a highly effective defense system that not only mitigates adversarial attacks but also functions as a robust NIDS.

V. Results & Performance Analysis

A. Experimental Setup

The proposed AAD methodology was developed in Python using TensorFlow 2.13, with the adversarial samples generated via the Adversarial Robustness Toolbox (ART) [35]. The implementation and evaluation were carried out using Python

TABLE I
STATISTICAL DESCRIPTION OF THE CSE-CIC-IDS2018 DATASET

Flow Type	Number of instances	Percentage
Benign	14,94,781	71.280
Type1	4,61,912	22.020
Type2	1,39,890	6.670
Type3	362	0.020
Type4	151	0.010
Type5	53	0.002

TABLE II
STATISTICAL DESCRIPTION OF THE DATASET USED TO PRODUCE ADVERSARIAL SAMPLES

Flow Type	Number of instances	Percentage
Benign	46,00,200	76.670
Type1	13,36,200	22.270
Type2	61,200	1.020
Type3	1200	0.020
Type4	180	0.003
Type5	120	0.002

3.10.11 on an Asus Vivobook gaming laptop with NVIDIA 1650 graphics. The AD was configured using scikit-learn. For each tensorflow-created DL model, we employed the cross-entropy loss function.

B. Dataset Description

We perform all experiments (including NID, white-box adversarial attacks, and attacks on NIDS enhanced with the proposed defenses) using the publicly available CSE-CIC-IDS2018 dataset [36]. This dataset includes 14 types of network intrusion traffic flows as well as benign traffic consisting of multiple files but due to computational constraints, we only used 2 of the data files dated 02-16-2018, and 02-23-2018. The attacks are classified as: D_o_S_Hulk (Type1), D_o_S_SlowHTTPTest (Type2), Brute_Force_Web (Type3), Brute_Force_XSS (Type4), SQL_Injection (Type5). The infrastructure for CSE-CIC-IDS2018 consists of 50 machines attempting to intrude on a victim network composed of 420 end hosts and 30 servers. Table I underlines the details of the dataset used for adversarial detection.

Table II describes the data used to generate the adversarial samples for our study. A total of 60,000 adversarial samples were created, with 5,000 samples generated per adversarial method,namely FGSM, PGD, DeepFool, and BIM, for each MLP, CNN, and LSTM classifier.

C. Performance Metrics

Performance evaluation of AAD frameworks, especially when dealing with adversarial attacks, relies on several traditional metrics used for intrusion detection systems, such as Accuracy, Precision, Recall, F1 Score, and Loss [34]. However, apart from traditional metrics in the realm of adversarial attack detection, another vital metric is the success rate. This measure quantifies the model's proficiency in recognizing malicious inputs. Specifically, it's calculated as the percentage of adversarial samples that the model correctly flags as harmful. The formula for success rate in this context can be expressed as:

$$\text{Success Rate} = \frac{\text{Correctly\_Identified\_Adversarial\_Samples}}{\text{Total\_Number\_of\_Adversarial\_Samples}} \quad (21)$$

Together, these evaluation measures offer a thorough assessment of a model's capabilities to correctly classify both benign and adversarial samples and its effectiveness in distinguishing adversarial examples from benign traffic.

TABLE III
NIDS ACCURACY AND LOSS

Model	Accuracy	Validation accuracy	Training Loss	Validation loss
MLP	99.98	99.97	0.01	0.00
CNN	99.92	99.91	0.02	0.00
LSTM	99.80	99.79	0.21	0.01

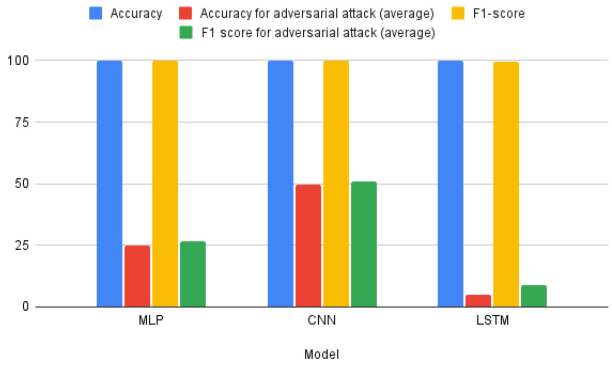

Fig. 3. Comparing the NIDS.

D. Results Analysis

1) Analyzing State-of-Art NIDS Models in Case of Intrusions and Adversarial Attacks: Table III presents the comparison three NIDS models: MLP, CNN, and LSTM. MLP performs best with 99.98% accuracy and lowest loss. CNN follows closely, while LSTM shows slightly lower performance. All models demonstrate high accuracy and low validation loss, indicating their effectiveness in network intrusion detection, with MLP and CNN outperforming LSTM slightly.

Upon launching the adversarial attacks, it is seen that these models become vulnerable and fall prey to attackers as shown in Fig. 3. It compares the performance of the above three models on a standard evaluation set and under adversarial attack conditions. The MLP model shows excellent accuracy (99.98%) and F1-score (99.95%) on standard evaluation, but its performance significantly drops under adversarial attacks, with an accuracy of 24.95% and an F1-score of 26.83%. This indicates that while the MLP performs well under normal conditions, it is highly susceptible to adversarial perturbations.

Similarly, the CNN model exhibits strong performance with an accuracy of 99.92% and an F1-score of 99.83% on standard evaluation. However, it retains better robustness under adversarial attacks compared to the MLP, with an accuracy of 49.76% and an F1-score of 50.8275. The LSTM model, while demonstrating high accuracy (99.80%) and F1-score

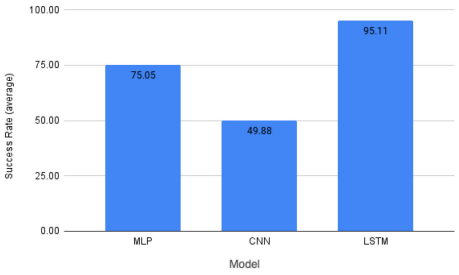

Fig. 4. NIDS success rate.

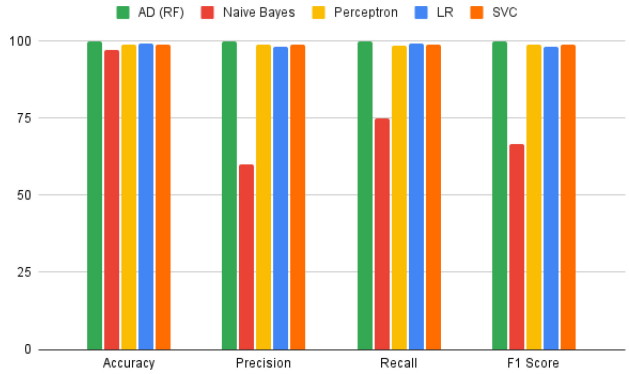

Fig. 5. Comparison of various ML models for AD.

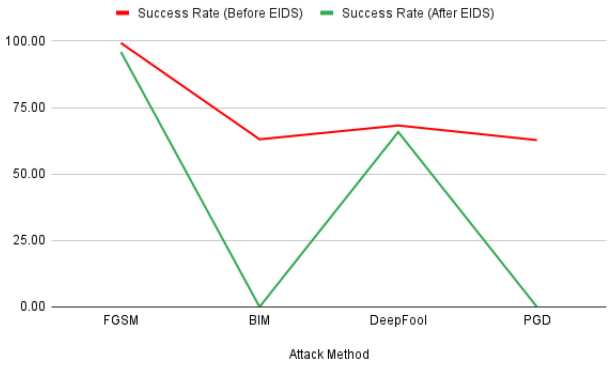

Fig. 6. Analysing success rate based on adversarial attack.

(99.57%) under normal conditions, suffers the most under adversarial attacks, with an accuracy of just 4.89% and an F1-score of 8.9075. This highlights the varying degrees of vulnerability among different model architectures when subjected to adversarial inputs.

Additionally, Fig. 4 highlights the success rates of adversarial attacks. For the MLP model, the adversarial attack success rate is 75.05%, indicating that a significant majority of adversarial attempts successfully compromised the model's performance. This aligns with the earlier observation of the MLP's substantial drop in accuracy and F1 score under adversarial conditions, showcasing its high vulnerability to such attacks.

Conversely, CNN demonstrates a better resilience to adversarial attacks with a success rate of 49.88%. While still significant, this rate is notably lower than that of the MLP, reflecting the CNN's comparatively stronger robustness. The LSTM model, however, exhibits the highest susceptibility with a success rate of 95.11%, indicating that nearly all adversarial attempts succeed in disrupting its performance. This extreme vulnerability underscores the need for enhanced defensive mechanisms for LSTM models in adversarial settings.

2) Analyzing the Performance of Adversarial Discriminator (AD) Against Adversarial Attacks: Fig. 5 presents the performance metrics for different classifiers in context to adversarial attacks.

The AD using RF achieved perfect performance, with accuracy, precision, recall, and F1 score all at 100%. This indicates that the RF model correctly identifies all samples, both benign and adversarial, without any false positives or false negatives. Such a high level of performance suggests that the RF model is highly effective in distinguishing adversarial examples from normal traffic.[1]

The Naive Bayes classifier achieved 97.26% accuracy but struggles with precision (59.83%) and recall (74.79%), resulting in an F1 score of 66.49%. This suggests it correctly classifies many samples but produces more false positives than other models. In contrast, the Perceptron model excels, with 98.98% accuracy, 98.86% precision, 98.63% recall, and a 98.75% F1 score. These metrics indicate the Perceptron's effectiveness in balancing true positive and negative identifications, making it a superior classifier for adversarial detection

The LR classifier also shows strong performance with an accuracy of 99.04%, precision of 98.02%, recall of 99.05%, and an F1 score of 98.13%. These results demonstrate that LR is particularly good at recall, indicating it can identify almost all actual positive adversarial examples, with slightly lower precision compared to Perceptron and SVC.

Lastly, the SVC exhibits high accuracy at 98.99%, with precision, recall, and F1 scores all around 98.85% to 98.94%. This consistency across all metrics signifies that the SVC is highly effective and reliable in classifying adversarial examples, comparable to the Perceptron and LR models.

3) Analyzing the Performance of EIDS Against Adversarial Attacks[2]*:* Fig. 6 showcases the effectiveness of an EIDS in mitigating adversarial attacks. It compares the success rates of various attack methods before and after the implementation of EIDS.

For the FGSM, the success rate slightly decreases from 99.30% to 95.96% after EIDS, indicating that while the defense mechanism reduces the attack's effectiveness, it remains highly successful. The BIM shows a dramatic decrease in success rate from 63.06% to 0.01% after EIDS is applied. This nearly complete mitigation highlights the robustness of EIDS against iterative attacks like BIM.

Similarly, PGD attacks experience a significant reduction in success rate from 62.76% to 0.01% post-EIDS implementation, demonstrating the defense's efficacy against this robust

[1]It should noted that the AD is only trained on the adversarial samples and identifies adversarial samples only. There is a possibility that the intrusion attacks me surpass the AD systems. This is where EIDS comes into picture and acts as both NIDS and adversarial attack detector.

[2]The efficacy of the EIDS in case of NIDS attacks can already be established using Table III as it's an ensemble of these models. However, more detail is also discussed in the next sub-section.

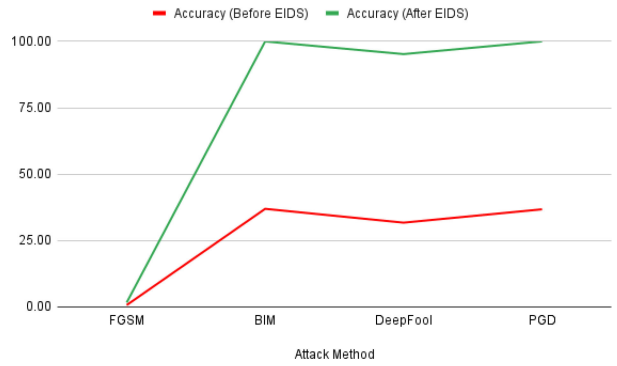

Fig. 7. Comparing the accuracy of EIDS with respect to different adversarial attacks.

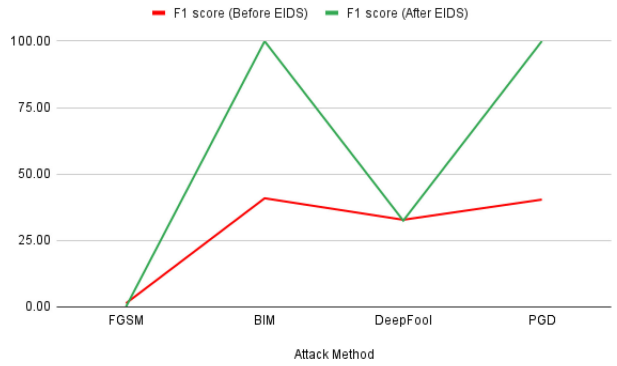

Fig. 8. Comparing the F1 score of EIDS for different adversarial attacks.

attack method. In contrast, the DeepFool attack's success rate decreases only slightly from 68.27% to 65.82% with EIDS. This suggests that while EIDS provides some level of defense, it is individually effective against DeepFool compared to BIM and PGD.

Fig. 7 provides insight into the effectiveness of an EIDS by comparing model accuracies before and after its implementation under various adversarial attack methods.

For the FGSM, the accuracy improves from 0.70% to 1.00% after applying EIDS. Although this is an improvement, the accuracy remains quite low, indicating that while EIDS offers some defense, FGSM attacks still significantly affect the model's performance. This can also be pre-countered using the AD, which achieved 100% on adversarial attack detection. The BIM shows a remarkable improvement in accuracy from 36.94% to 99.98% with EIDS. This dramatic increase demonstrates the high effectiveness of EIDS in defending against BIM attacks, essentially restoring the model's performance to its original accuracy level.

The DeepFool attack detection also sees a significant increase in accuracy from 31.73% to 95.20% after EIDS is applied. While not as complete as the defense against BIM, this substantial improvement indicates that EIDS is highly effective in mitigating the impact of DeepFool attacks. The PGD method experiences a perfect restoration of accuracy from 36.76% to 100.00% post-EIDS implementation. This complete recovery highlights EIDS's robust defense capabilities against PGD attacks, effectively neutralizing their impact.

Apart from attack success rate and accuracy, Fig. 8 highlights the impact of the EIDS on the F1 scores. For the FGSM, the F1 score decreases from 1.37 before EIDS to 0.10 after EIDS, indicating that while EIDS changes the attack's impact, it does not improve the model's performance under FGSM attacks. This suggests that the defense mechanism, in this case, might be causing excessive caution, reducing the model's ability to correctly classify even non-adversarial examples.

Conversely, for the BIM and PGD, the F1 scores improved dramatically after implementing EIDS. The F1 score for BIM rises from 40.88 to 99.95, and for PGD, it increases from 40.38 to 99.99. These near-perfect post-EIDS F1 scores highlight the efficacy of EIDS in defending against these iterative attack methods. In contrast, the F1 score for DeepFool remains relatively unchanged, decreasing slightly from 32.79 to 32.46, indicating that EIDS does not significantly affect the model's resilience to DeepFool attacks. This underscores that while EIDS is highly effective against certain attacks, its effectiveness can vary depending on the attack method.

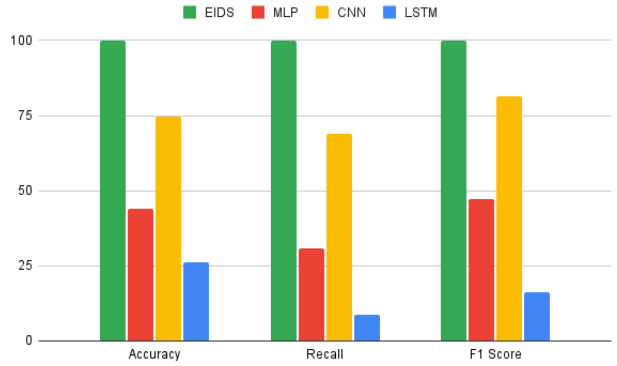

Fig. 9. Comparison of EIDS with other NIDS against intrusion and adversarial attacks.

In summary, EIDS significantly enhances model robustness against various adversarial attacks, with especially notable improvements for iterative methods like BIM and PGD, while also providing substantial defenses against FGSM and DeepFool attacks.

4) Analyzing the Performance of EIDS Against Adversarial and Intrusion Attacks: Fig. 9 compares the performance of the EIDS against individual models (MLP, CNN, LSTM) in a comprehensive scenario that includes both intrusion attacks, adversarial attacks, and benign traffic. EIDS demonstrates superior performance with an accuracy of 99.96%, a recall of 99.96%, and an F1 score of 99.98%. This indicates that EIDS is highly effective in correctly identifying and classifying both benign and malicious traffic, maintaining a balance between precision and recall, and achieving near-perfect performance across all metrics.

In contrast, the MLP model shows significantly lower performance with an accuracy of 44.18%, a recall of 30.98%, and an F1 score of 47.3%. This suggests that MLP struggles considerably in this mixed scenario, particularly with recall, indicating a high rate of false negatives where actual threats are missed.

The CNN model performs better than MLP but still falls short compared to EIDS, with an accuracy of 74.81%, recall of 68.87%, and an F1 score of 81.56%. While CNN shows

a more balanced performance, it still has considerable room for improvement, especially in recall, reflecting challenges in detecting all intrusions and adversarial attacks accurately.

The LSTM model exhibits the lowest performance, with an accuracy of 26.18%, a recall of 8.75%, and an F1 score of 16.09%. These metrics highlight the LSTM's difficulty in handling the combined scenario, particularly in detecting actual threats, as indicated by its extremely low recall and F1 score.

Overall, it underscores the robustness of EIDS and highlights its effectiveness in providing a resilient and reliable defense mechanism against a wide range of cyber threats.

Overall, the combined power of the validator, AD, and EIDS presents a robust and comprehensive solution to the challenges posed by adversarial attacks. The validator acts as the first line of defense, filtering out potential adversarial inputs before they reach the core detection system. The AD ensures that subtle adversarial manipulations are effectively detected and classified. Finally, the EIDS, employing an ensemble of DL models further enhances the robustness by leveraging the strengths of each model to accurately identify and mitigate a wide range of threats. This multi-layered approach not only improves detection accuracy and reduces false positives but also significantly enhances the overall resilience of the network security system against sophisticated adversarial attacks, thereby providing a highly effective defense mechanism.

VI. Conclusion and Future Work

In summary, this paper addresses the critical vulnerability of ML and DL-based Intrusion Detection Systems (IDS) to white-box adversarial attacks, with a focus on IoT environments. While standalone models like MLP, CNN, and LSTM achieved higher performance under normal conditions, their accuracy and F1 scores drop drastically when exposed to adversarial inputs; highlighting a significant security gap. To counter this, we introduced the Adversarial Attack Detector (AAD) framework, a multi-layered defense system comprising a request validator, an Adversarial Discriminator (AD), and an Enhanced IDS (EIDS). Empirical results show that AAD substantially improves resilience against a range of white-box attacks, restoring detection performance even under adversarial pressure.

Although the framework shows strong potential in experimental settings, further research is needed to assess its real-world applicability and generalizability. Future work will focus on deploying AAD in practical environments to validate its effectiveness under real-world network conditions.

Appendix

A complete list of abbreviations is included in the Appendix (Section A) for the reader's convenience.

- Intrusion Detection Systems (IDS)
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Network (CNN)
- Long Short-Term Memory (LSTM)
- Internet of Things (IoT)
- Machine Learning (ML)
- Deep Learning (DL)
- Deep Neural Networks (DNNs)
- Network Intrusion Detection Systems (NIDS)
- Generative Adversarial Network (GAN)
- Fast Gradient Sign Method (FGSM)
- Projected Gradient Descent (PGD)
- Carlini & Wagner (CW)
- Feature-Level Attack (FLA)
- Membership Inference Attack (MIA)
- Adversarial Robustness Toolbox (ART)
- Nearest Class Mean (NCM)
- Adversarial Discriminator (AD)
- Enhanced Intrusion Detection System (EIDS)
- Enhanced IDS (EIDS)
- Adversarial Attack Detector (AAD)

Acknowledgment

For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

[1] A. Paya, S. Arroni, V. García-Díaz, and A. Gómez, "Apollon: A robust defense system against adversarial machine learning attacks in intrusion detection systems," *Comput. Security*, vol. 136, Jan. 2024, Art. no. 103546.

[2] Z. A. El Houda, H. Moudoud, B. Brik, and M. Adil, "A privacy-preserving framework for efficient network intrusion detection in consumer network using quantum federated learning," *IEEE Trans. Consum. Electron.*, vol. 70, no. 4, pp. 7121–7128, Nov. 2024.

[3] M. Obaidat and N. Boudriga, *Security of E-Systems and Computer Networks*.Cambridge, U.K.: Cambridge Univ. Press, 2007.

[4] "2023 official cybercrime report," eSentire, 2023. [Online]. Available: https://www.esentire.com/resources/library/2023-official-cybercrime-report

[5] D. Javeed, T. Gao, P. Kumar, and A. Jolfaei, "An explainable and resilient intrusion detection system for industry 5.0," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 1342–1350, Feb. 2024.

[6] L. Yang and A. Shami, "IDS-ML: An open source code for intrusion detection system development using machine learning," *Softw. Impacts*, vol. 14, Dec. 2022, Art. no. 100446.

[7] P. Parkar and A. Bilimoria, "A survey on cyber security IDS using ML methods," in *Proc. 5th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, 2021, pp. 352–360.

[8] H. Babbar and S. Rani, "FRHIDS: Federated learning recommender hybrid intrusion detection system model in software-defined networking for consumer devices," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2492–2499, Feb. 2024.

[9] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: Recent developments and challenges," *Soft Comput.*, vol. 25, no. 15, pp. 9731–9763, 2021.

[10] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, 2022, Art. no. e3803.

[11] X. Yuan, S. Han, W. Huang, H. Ye, X. Kong, and F. Zhang, "A simple framework to enhance the adversarial robustness of deep learning-based intrusion detection system," *Comput. Security*, vol. 137, Feb. 2024, Art. no. 103644.

[12] Z. Bao, Y. Lin, S. Zhang, Z. Li, and S. Mao, "Threat of adversarial attacks on DL-based IoT device identification," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 9012–9024, Jun. 2022.

[13] A. Aldahdooh, W. Hamidouche, S. A. Fezza, and O. Déforges, "Adversarial example detection for DNN models: A review and experimental comparison," *Artif. Intell. Rev.*, vol. 55, no. 6, pp. 4403–4462, 2022.

[14] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[15] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2484–2493.
[16] L. Ye and S. M. Hamidi, "Thundernna: A white box adversarial attack," 2021, *arXiv:2111.12305*.
[17] Y. Xu, X. Zhong, A. J. Yepes, and J. H. Lau, "Grey-box adversarial attack and defence for sentiment classification," 2021, *arXiv:2103.11576*.
[18] X. Wang and W. H. Wang, "GCL-Leak: Link membership inference attacks against graph contrastive learning," in *Proc. Privacy Enhanc. Technol.*, 2024, pp. 1–21.
[19] 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. Security Privacy (SP)*, 2016, pp. 582–597.
[20] X. Jia, Y. Zhang, B. Wu, K. Ma, J. Wang, and X. Cao, "LAS-AT: Adversarial training with learnable attack strategy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13398–13408.
[21] K. G. Vamvoudakis, J. P. Hespanha, B. Sinopoli, and Y. Mo, "Detection in adversarial environments," *IEEE Trans. Autom. Control*, vol. 59, no. 12, pp. 3209–3223, Dec. 2014.
[22] S. Zhang, H. Gao, and Q. Rao, "Defense against adversarial attacks by reconstructing images," *IEEE Trans. Image Process.*, vol. 30, pp. 6117–6129, 2021.
[23] T. Anastasiou et al., "Towards robustifying image classifiers against the perils of adversarial attacks on artificial intelligence systems," *Sensors*, vol. 22, no. 18, p. 6905, 2022.
[24] Z. Zhao, Y. Zhang, G. Chen, F. Song, T. Chen, and J. Liu, "CLEVEREST: Accelerating CEGAR-based neural network verification via adversarial attacks," in *Proc. Int. Static Anal. Symp.*, 2022, pp. 449–473.
[25] H. Lee, H. Bae, and S. Yoon, "Gradient masking of label smoothing in adversarial robustness," *IEEE Access*, vol. 9, pp. 6453–6464, 2020.
[26] S. Shan, W. Ding, E. Wenger, H. Zheng, and B. Y. Zhao, "Post-breach recovery: Protection against white-box adversarial examples for leaked DNN models," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2022, pp. 2611–2625. [Online]. Available: https://doi.org/10.1145/3548606.3560561
[27] O. Gungor, T. Rosing, and B. Aksanli, "STEWART: Stacking ensemble for white-box adversarial attacks towards more resilient data-driven predictive maintenance," *Comput. Ind.*, vol. 140, Sep. 2022, Art. no. 103660.
[28] D. Wu et al., "Understanding and defending against white-box membership inference attack in deep learning," *Knowl.-Based Syst.*, vol. 259, Jan. 2023, Art. no. 110014.
[29] N. Alhussien, A. Aleroud, A. Melhem, and S. Y. Khamaiseh, "Constraining adversarial attacks on network intrusion detection systems: Transferability and defense analysis," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 3, pp. 2751–2772, Jun. 2024.
[30] X. Zhou, P. Qi, W. Zhang, S. Zheng, N. Zhang, and Z. Li, "GAN-based Siamese neuron network for modulation classification against white-box adversarial attacks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 10, no. 1, pp. 122–137, Feb. 2024.
[31] F. Zhang, P. P. Chan, B. Biggio, D. S. Yeung, and F. Roli, "Adversarial feature selection against evasion attacks," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 766–777, Mar. 2016.
[32] D. Bhattacharyya, R. Ranjan, F. Alisherov, and M. Choi, "Biometric authentication: A review," *Int. J. u-e-Service, Sci. Technol.*, vol. 2, no. 3, pp. 13–28, 2009.
[33] D. Parekh et al., "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, p. 2162, 2022.
[34] N. Chaurasia, M. Ram, P. Verma, N. Mehta, and N. Bharot, "A federated learning approach to network intrusion detection using residual networks in industrial IoT networks," *J. Supercomput.*, vol. 80, pp. 18325–18346, May 2024.
[35] M.-I. Nicolae et al., "Adversarial robustness toolbox V1.0.0," 2018, *arXiv:1807.01069*.
[36] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, vol. 1, 2018, pp. 108–116.