Modeling Realistic Adversarial Traffic Against Deep-Learning-Based Intrusion Detection System in Industrial IoT

Wei Yao, Haixia Peng, *Member, IEEE*, Qihao Li, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—The widely deployment of infrastructure and wireless interfaces increases industrial IoT (IIoT) vulnerability to network intrusions, highlighting the requirements for robust network intrusion detection systems (NIDSs). Although deep learning (DL) provides a promising solution for NIDSs, it remains susceptible to adversarial attacks as minor input perturbations can lead to major misclassifications. In this article, we propose a packet-level adversarial traffic generation (PATG) approach for attacking NIDSs in IIoT, which not only aligns with domain constraints but also evades various DL-based NIDSs. Particularly, we introduce a reversible abstract traffic representation to ensure that the original traffic can be effectively modified while preserving its functionality. We propose a packet-level generative adversarial networks to craft adversarial traffic by learning benign data distribution in feature space and simulating evasion behaviors, which escapes the DL-based NIDSs. We further design two defense schemes to enhance system resilience against proposed adversarial attacks. We evaluate PATG on nine state-of-the-art DL-based NIDSs in the Kitsune and CICIoT23 datasets. Experimental results demonstrate that PATG can achieve a maximum evasion increase rate of 99% with cost-effective execution, while the defense methods significantly mitigate the impact of the adversarial attacks.**

***Index Terms*—Adversarial traffic attacks, deep learning (DL), generative adversarial network, Industrial Internet of Things (IIoT), intrusion detection systems (NIDSs).**

I. Introduction

As INDUSTRY 4.0 era rapidly advances, the adoption of Internet of Things (IoT) across industries such as automotive, aerospace, and logistics has become prevalent [1]. The swift expansion of industrial IoT (IIoT) devices has significantly boosted production efficiency and fault tolerance. However, this rapid expansion has triggered a surge in network infrastructure threats, ranging from ransomware to IoT botnets [2]. The shifting IIoT security landscape necessitates the development of resilient and robust network intrusion detection systems (NIDSs) to effectively detect potential threats. To mitigate increasingly sophisticated attacks, various machine learning (ML) and deep-learning (DL)-based NIDSs have been developed in recent years [3], [4], [5], including multilayer perceptron (MLP) [6], convolutional neural networks (CNNs) [7], long short-term memory (LSTM) networks [7], deep autoencoders [8], and gated recurrent Units (GRUs) [9]. For example, Zhang et al. [7] developed a packet-level NIDS that achieves a high detection rate by combining CNN and LSTM models to capture essential spatial features. At the core of these methods is the extraction of high-dimensional feature vectors from network traffic, which are then utilized by ML or DL models to train and classify as benign and malicious. Despite their considerable promise and performance, ML/DL models are particularly susceptible to adversarial attacks, where minor and carefully designed alterations in the input can result in significant shifts to the model's output [10], [11], [12], [13]. To comprehensively understand the upper limits of the generalization ability of ML/DL-based systems, constructing adversarial attacks becomes a necessity for evaluating the robustness of the systems [10], [13]. For these attacks to be applicable, they must authentically reflect an adversary's capabilities in real-world scenarios. Otherwise, their relevance is limited.

The approaches of adversarial attacks on ML/DL systems have been well-studied in fields like computer vision [12], [14], natural language processing [15], and malware detection [16]. However, these approaches cannot be directly applied to network intrusion detection for two reasons. First, unlike in nonsecurity domains, modified malicious traffic in NIDS must adhere to strict communication protocol rules, with fixed headers and payloads. Second, these attacks generate adversarial examples in feature space rather than raw traffic space. Different from images where features can be easily mapped back to original data, traffic features like flow length or bytes per second are neither reversed nor differentiated, making those attacks impractical for NIDSs.

Currently, several methods of adversarial attacks (also known as evasion attacks), targeting either feature space or traffic space have been developed against the ML/DL-based NIDSs [17], [18], [19], [20], [21], [22]. Although most

Received 26 January 2025; revised 10 April 2025; accepted 30 April 2025. Date of publication 9 May 2025; date of current version 25 July 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62301411; in part by the Dongguan Strategic Scientist Teams Project under Grant 20231900700022; and in part by the Research Initiation Special Fund Project of Dongguan University of Technology under Grant 221110333. *(Corresponding author: Haixia Peng.)*

Wei Yao is with the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, Dongguan 523808, China (e-mail: yaow.neu@gmail.com).

Haixia Peng is with the School of Information and Communications Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: haixia.peng@xjtu.edu.cn).

Qihao Li is with the College of Communication Engineering, Jilin University, Changchun 130015, China (e-mail: qihaol@jlu.edu.cn).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/JIOT.2025.3568503

existing methods demonstrate promising attack effectiveness, we identify three limitations when applying them to real-world scenarios. First, various techniques have been introduced to alter raw traffic or features for bypassing detection based on traffic analysis [17], [18], [23]. In [23], features are randomly changed to evade botnet detectors. Nevertheless, these methods for modifying traffic or features generally focus on imitating benign traffic or introducing random alterations to malicious traffic, rather than leveraging adversarial methods to exploit the weaknesses of ML/DL models. This could result in breaches of protocol compliance or loss of malicious functionality. Consequently, such attacks tend to be costly and ineffective, conflicting with attackers' common concern about minimizing attack overhead such as timing. Second, most evasion attack methods perturb the traffic features based on white- or gray-box settings [11], [19], [20], [21], [24]. Hashemi et al. [11] proposed a white-box attack based on legal traffic manipulation to evade ML-based detection. In [21], an adversarial attack framework based on generative adversarial networks is proposed, which utilizes minimal prior knowledge to generate realistic adversarial traffic for deceiving ML-based detection. In these scenarios, the attackers possess complete knowledge of the targeted NIDS, including the utilized features, architecture, and parameters of the ML models, as well as the feedback from the target models (e.g., predicted labels or confidence). In practice, however, attackers rarely have access to such detailed information. Finally, adversarial attacks on the ML/DL-based NIDSs in feature space and on the ML-based NIDSs in traffic space have been well investigated [22], [25], [26], [27]. Sharon et al. [22] introduced an end-to-end evasion attack method, which leverages an LSTM network to learn interpacket delays of normal traffic and predict the timing of subsequent attack traffic. However, a thorough investigation of adversarial attacks in traffic space targeting DL-based NIDSs remains insufficient.

In this article, we propose a novel method called packet-level adversarial traffic generation (PATG) for generating practical adversarial traffic attacks. It generates malicious packet-level traffic with limited prior knowledge to effectively evade the DL-based NIDSs in IIoT. We propose a reversible traffic vectorization to transform the crafted adversarial traffic information within valid ranges, ensuring compliance with communication protocol rules while preserving malicious functionality. To generate effective adversarial traffic, the wasserstein generative adversarial networks (WGANs) is exploited to learn the data distribution of benign data in feature space and mimic evasion behaviors [28]. Additionally, we explore two defense schemes to strengthen the system's robustness against these attacks. In Table I, we compare our proposed method with most existing adversarial attacks and outline their unrealistic requirements. In summary, the main contributions of this article are as follows.

1) We propose a new practical adversarial attack method PATG to evade DL-based malicious traffic detection in both gray- and closed-box contexts. Compared to previous studies, our method adheres to protocol standards and retains malicious intent, all while requiring minimal prior knowledge and incurring low execution overhead.

TABLE I
COMPARISON BETWEEN OUR METHOD AND EXISTING STUDIES

Requirements	Study	Our method
Access to small training data	[22], ✗	✓
Legal traffic modification	[17], [18], ✗	✓
Target for DL-based NIDS	[23], [26], ✗	✓
Low execution cost	[11], [24], [26], ✗	✓
Knowledge on feature extractor	[23], [25], [26], ✓	✗
Knowledge on NIDS	[11], [21], [27], ✓	✗
Alter extracted features	[23], [24], [26], ✓	✗

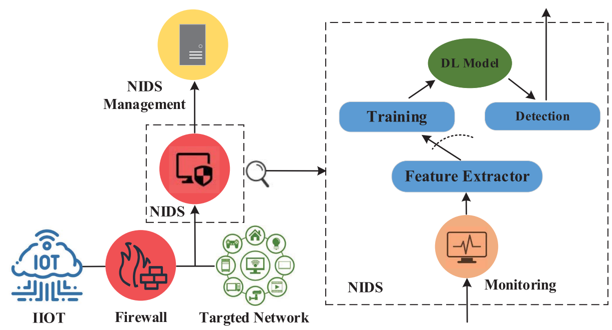


Fig. 1. System model of the DL-based NIDS.

2) We design two defense schemes to enhance the system's robustness against the proposed attacks. The adversarial training (AT) approach, in particular, effectively mitigates these attacks by retraining on adversarial features.
3) We conduct an extensive evaluation of our attack method on the state-of-the-art DL-based NIDSs (namely nine representative deep neural networks (DNNs)) and defense methods. The experimental findings demonstrate PATG's effectiveness in evading detection and cost efficiency in attacking.

The remainder of this article is as follows. In Section II, the system model, threat model, and design goals are presented. Section III details the PATG method and its two core modules. Section IV presents the experimental results, key findings, and defense schemes. Section V further discusses the possibility of adversarial attacks. Finally, Section VI concludes this article.

II. Problem Formulation

In this section, we formulate the system model, threat model, and the design goals of this article.

A. System Model

As shown in Fig. 1, we consider a system model for a generic detection scenario in IIoT. An NIDS is hosted on a central server with strong computational resources, deployed at the core of the target network. The NIDS sniffs the firewall's internal interface in read-only mode and analyzes the ongoing traffic from IIoT devices entering the core network via the firewall. Initially, traffic is monitored and captured from the targeted network. Features are then extracted into predefined feature vectors from the raw traffic, and ultimately input into a DL classifier for training or detection. Unlike previous works, our study focuses on closed box traffic space attacks to evade DL-based NIDS. Once malicious traffic is detected by the trained DL model, an alert will be sent to an NIDS management server to prevent this traffic.

B. Threat Model

Attackers' Knowledge: Unlike previous white- or gray-box attacks, the attackers in this context are not required to possess knowledge of the target classifier's architecture, output labels, or confidence levels. Assume that an adversary can intercept system communications to obtain training data with the similar distribution. In practice, attackers often gain access to network traffic through techniques, such as packet sniffing and traffic hijacking. Additionally, it is assumed that the attacker possesses a higher level of prior knowledge regarding the meaning of features. Namely, the attackers can leverage publicly statistical features (e.g., flow duration and packet sizes) that are commonly adopted in industrial NIDS implementations (e.g., Snort [29], Zeek [30], and Kitsune) to approximate a proxy feature extractor. For instance, attackers may require only general domain knowledge such as common protocol fields like interarrival timing, acquired through domain-general tools, rather than detailed information about the specific feature engineering process. Moreover, our empirical evaluation (see Fig. 9 in Section IV-D) on testing datasets demonstrates that the proposed method can still achieve a 30% increase in evasion increase rate (EIR) against closed-box DL-based NIDSs, even when the attacker lacks knowledge of the exact feature space. This transferability stems from the semantic overlap in high-level traffic patterns (e.g., temporal regularity) that are widely exploited by most NIDSs. For example, simply modifying interpacket timing enables attackers to manipulate temporal features such as flow duration across a broad range of systems, regardless of their specific implementation details. This assumption is well-supported by previous studies, which have shown that adversarial attackers can reverse-engineer learning problems to extract sufficient information about ML models, thereby facilitating the construction of effective adversarial attacks [31]. Based on the attacker's understanding of the targeted features, the following two types of attacks.

1) *Gray-Box Realistic Attack (GRA):* This assumes that the attackers possess knowledge of the features used by the targeted NIDS, enabling them to construct an identical feature extractor for feature extraction. Although this may appear to be a challenging requirement, attackers can still approximate the feature extraction through domain-specific insights, as many of the features commonly utilized in ML-based NIDSs are publicly documented and accessible in the public domain.
2) *Closed-Box Realistic Attack (BRA):* A more realistic scenario involves the attacker having limited or no awareness of the features utilized by the targeted NIDS. Namely, the attackers can only rely on the domain knowledge (e.g., exploiting transferability principles of DL models [32]) to construct a proxy feature extractor.

Attackers' Goal: We assume the attackers can penetrate the firewall by using tools (e.g., port scanning) and gain access to a small portion of the target network's traffic. Different from existing feature-space attacks, the attackers aim to mislead the NIDS model by altering the raw traffic of controlled devices (i.e., traffic-space attacks) with affordable costs. Formally, attackers attempt to deceive the model F by introducing minimal perturbations δ. Thus, the objective is set to

$$\arg\min_{\delta} F(E(R(x+\delta))) \neq F(E(x)) \tag{1}$$

where x is the input malicious traffic, E represents feature extraction, and R denotes the constraints on adversarial traffic.

Attackers' Capabilities: In practice, it is impossible for attackers to arbitrarily modify the malicious traffic without concerning the execution overhead. Therefore, we assume that attackers can modify a few raw packet fields while ensuring the modified samples can remain convertible to compliant network packets. Specifically, certain fields of the malicious samples, including packet length, timestamps, and protocol layers (PLs), must be correctly formatted; otherwise, the malicious packets cannot be properly parsed by the target victim.

C. Design Goals

Under the threat assumptions, the PATG should achieve the following goals.

1) *Legitimacy:* Adversarial traffic must adhere to protocol rules and be communicable through the channel, such as the TCP headers specifying source ports, sequence numbers, and control flags. Even a single-bit wrong modification may render the entire packet invalid.
2) *Maliciousness:* Adversarial traffic must inflict damage or engage in malicious activities. For example, if denial of service (DoS) traffic fails to deliver a significant volume of packets to the target in a short period, it loses its malicious property.
3) *High Evasion Rate and Efficiency:* Adversarial traffic should be capable of successfully evading various DL-based NIDSs while maintaining attack execution cost as low as possible.

III. PATG Methodology

In this section, we first give an overview of the PATG method, and then provide the details of its two primary modules individually.

A. Overview of PATG

The main idea of PATG is to generate realistic packet-level adversarial traffic to evade various DL-based NIDSs. To achieve this, we must determine how altering specific raw traffic can effectively influence feature values after feature extraction and develop a method to link raw traffic information with feature behavior. With this understanding, we can construct a traffic generator that adds small perturbations into vectorized raw traffic. This generator, when processed through feature extraction, enables a discriminator and a surrogate classifier to distinguish between modified malicious traffic and normal traffic in feature space, which can solve the challenge of feature-to-traffic mapping within the low overhead of

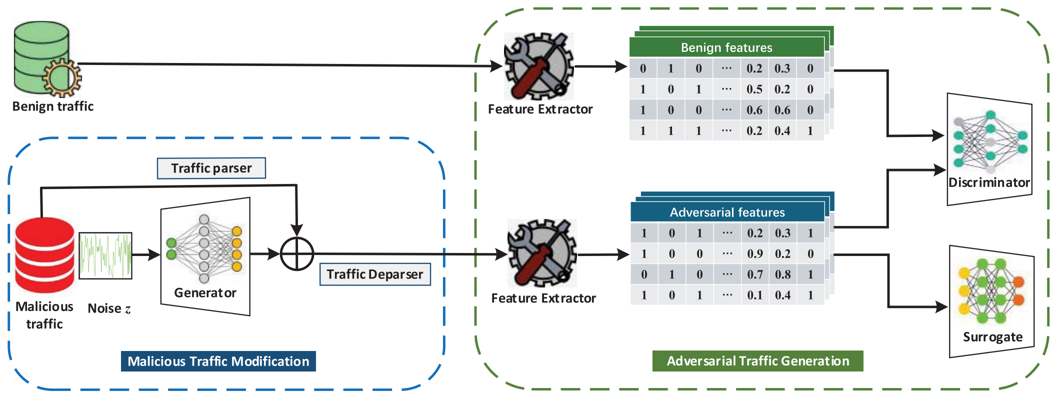


Fig. 2. Overview of proposed PATG methodology.

crafting traffic. To this end, two core modules, forming an end-to-end pipeline from real-world malicious traffic modification to downstream adversarial traffic generation are considered in PATG, as shown in Fig. 2.

Malicious Traffic Modification: This module aims to modify and vectorize network malicious traffic while preserving its legitimacy and maliciousness. Specifically, it involves altering the specific attributes of the original packets to influence the feature space. We propose a vectorization and recovery approach to convert traffic into a vector containing metadata from the original traffic, and this conversion operation is reversible. The modified traffic will be subjected to the adversarial traffic generation module for training.

Adversarial Traffic Generation: The purpose of this module is to make the carefully generated malicious traffic match the data distribution of benign samples in feature space and evade the surrogate classifier of the simulated targeted NIDS. Unlike existing methods that randomly modify malicious traffic, this module utilizes a generator based on collected malicious traffic from the target network. The feature extractor extracts features from both benign and crafted malicious traffic, which are then fed into the packet sequence WGAN (PS-WGAN) model, as shown in Fig. 5. The PS-WGAN consists of a generator, a discriminator, and a surrogate classifier. The discriminator and surrogate classifier differentiate between these traffic in feature space. After training, the generator can produce minimal perturbations to modify the malicious traffic.

The details of the above two modules are described in the next two Sections III-B and III-C, respectively.

	No.	Time	Source(Attacker)	Destination(Target)	Protocol	Length(Payload)
$\hat{P}_1$ ←	1.	0.1	192.168.1.116:55723	166.111.22.33:80	ICMP	139(95)
P_1 ←	2.	0.14	192.168.1.116:55723	166.111.22.33:80	TCP	64(10)
	3.	0.35	192.168.1.116:55723	166.111.22.33:8080	UDP	68(110)
P_2 ←	4.	0.45	192.168.1.116:55723	166.111.22.33:8080	UDP	46(98)
	...	...	...	...	...	...

Fig. 3. Format of malicious data, where each raw denotes an IP packet and each column represents a packet field. The crafted packet and the raw packet are, respectively, highlighted in red and black.

B. Malicious Traffic Modification

In this section, we begin by detailing the modifications made to malicious traffic, followed by the explanation of vectorizing raw traffic into metadata vectors and its recovery.

Primary Traffic Manipulation: To enable crafted adversarial traffic to effectively evade detection by DL-based NIDSs, we first design traffic modification operations. These modifications aim to affect as many feature types as possible with minimal changes, while maintaining both maliciousness and functionality. In practice, the attackers have little or no knowledge of the target system's features (i.e., GRA or BRA). To address this challenge, we employ a summarization method for features used in NIDSs [8], and categorize them into temporal and spatial features. Temporal features relate to aspects such as flow duration, and spatial features are separated into global and local ones. Global spatial features reflect overall traffic characteristics (e.g., volume in bytes or packet count), whereas local spatial features pertain to packet content. Given our focus on packet headers and specific protocols, local spatial features are somewhat constrained.

Then, we present modification operations that affect all summarized high-level features while maintaining functionality. Given that altering traffic headers is complex, crafted packets must align with nearby original packets in fields like IP/MAC/port to influence feature extraction. A prior study [11] indicates that modifying the time to live field requires topology information of the target network, which is unfeasible in practice. Hence, we consider alternative methods for various types of traffic without requiring additional knowledge.

1) *Adjusting Interarrival Time (IAT) of Packets:* This indicates that the interpacket arrival time should not exceed the maximum IAT between any two original malicious and normal packets.
2) *Adjusting PL of Packets:* The modified PL must adhere to established network standards (e.g., TCP and UDP at layer 3).
3) *Adjusting Payload Size (PS) of Packets:* The PS should not exceed the maximum value allowed by the PL. For example, the maximum TCP packet size theoretically should not surpass the network's maximum transmission unit (MTU), which is set as 1500 bytes.

To facilitate clarity, Fig. 3 illustrates the data format of malicious packets and highlights the fields subject to modification.

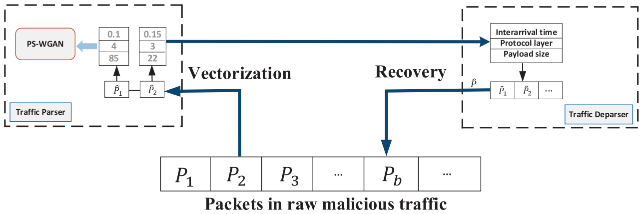


Fig. 4. Vectorization and recovery between vectors and raw traffic.

Each row essentially contains the 5-tuples of a flow. After applying the modifications, each packet is reconstructed with all fields (e.g., length and checksum) correctly updated. Based on the modified packet fields, we present the details of traffic operation in the next part.

Traffic Vectorization and Recovery: To enable numerical operation on structured traffic data, we convert the traffic into metadata vectors that encapsulate the original traffic information. In contrast to feature extraction, this vectorization method is reversible, allowing for the easy recovery of traffic from the metadata vectors. The previously described modification fields are also represented within the vectors. Combining with Fig. 3, the details of the metadata vector and examples of vectorization and recovery are illustrated in Fig. 4.

1) Traffic Parser: The traffic parser, $\mathbf{V}$, converts raw malicious IP packets $\mathbf{D}$ into a set of sequential data samples, serving as training data for the subsequent generator model. The following vectorization process is included.

Vectorization: For each packet $x \in \mathbf{D}$, three sequential fields, IAT, PS, and PL, are extracted to form a vector P. A packet sequence is denoted by $\mathbf{P} = [P_1, P_2, \ldots, P_b] \in \mathbb{R}^{3b}$, where b is the batch size. Hence, the vectors $\mathbf{P} = \mathbf{V}(\mathbf{D})$ stores the three statistical values of original traffic. The IAT for the first packet in the sequence is set to zero.

2) Traffic Deparser: It is responsible for recovering the ordinal fields to produce continuous packet-level data. Let $\mathbf{R}$ denotes the traffic deparser. For the ordinal field sequence $\widehat{\mathbf{P}}$ generated by the generator G in PS-WGAN, the recovery process involves the following steps.

1) *Truncation:* For the IAT, PS, and PL fields of the vectors in $\widehat{\mathbf{P}}$, generated values are truncated to $[0, 1]$ to match the space constraints.
2) *Denormalization:* This is the inverse process of rescaling the generated value back to its original range. For each dimension $d \in \{1, 2, 3\}$ of the vectors in $\widehat{\mathbf{P}}$, it is achieved by
$$\widetilde{\mathbf{P}}^d = \widehat{\mathbf{P}}^d\left(\max_{\mathbf{D}}^d - \min_{\mathbf{D}}^d\right) + \min_{\mathbf{D}}^d \tag{2}$$
where $\min_{\mathbf{D}}$ and $\max_{\mathbf{D}} \in \mathbb{R}^3$ represent the minimum and maximum values of network settings, respectively.
3) *Restoration:* Except for the IAT field, the denormalized values $\widetilde{\mathbf{P}}$ for PS and PL fields are approximated to integers, as seen in real packets.
4) *Fabrication:* For each restored value, it is then restructured as the nominal field of the corresponding packet. This entire process is denoted as $\mathbf{R}(\widehat{\mathbf{P}})$.

C. Adversarial Traffic Generation

After modifying malicious packets, the resulting traffic is passed through an adversarial traffic generation module to create features that evade DL-based NIDSs. This module includes a feature extractor and the PS-WGAN model. The extractor extracts relevant features, while the PS-WGAN generates adversarial packet-level traffic by learning benign data distribution in feature space. As the feature extractor is not the main focus, we next primarily introduce the PS-WGAN.

The PS-WGAN is designed to generate packet-level adversarial traffic with a WGAN model tailored for packet fields. The WGAN is involved to address the training instability issues in vanilla GAN. WGAN-div [28], the most advanced variant in the WGAN family, has demonstrated an enhanced stability during training. Based on WGAN-div, we then propose the PS-WGAN, as shown in Fig. 5, which consists of a surrogate classifier, a generator, and a discriminator. Essentially, the training process of a PS-WGAN model represents Nash equilibrium, where the generator attempts to create realistic packet sequences from a noise distribution and integrate them into the metadata vectors. The surrogate classifier assists in determining whether the generated samples are adversarial, while the discriminator differentiates between generated and benign data in feature space. Through this iterative competition, the generator refines its ability to match noise distribution with benign data distribution, ultimately producing high-quality adversarial traffic sequences. The details of the three components of our PS-WGAN, along with the training and execution processes, are given as follows.

Surrogate Classifier: To bridge the knowledge gap between the attackers and the closed-box DL-based NIDSs, we first construct a surrogate classifier C with transferable characteristic [32], [33]. This classifier serves two primary purposes: 1) to approximate the decision boundary of the target closed-box model via transferable AT and 2) to provide differentiable gradient signals for generating adversarial samples. Model the surrogate classifier as a K-layer DNN parameterized by $\Theta = [\boldsymbol{W}^{\{[k]\}}, \boldsymbol{b}^{\{[k]\}}]$, where each layer $\boldsymbol{L}^{[k]}$, $k \in \{1, \ldots, K\}$, consists of a weight matrix $\boldsymbol{W}^{[k]} \in \mathbb{R}^{d_k \times d_{k-1}}$ with d_k hidden units, a bias vector $\boldsymbol{b}^{[k]} \in \mathbb{R}^{d_k}$, and an activation function δ, such as the Sigmoid or rectified linear unit (ReLU). Given an input $\boldsymbol{X}$, the transformation at the kth layer is defined as
$$\boldsymbol{L}^{[k]} = \delta\left(\boldsymbol{W}^{[k]} \cdot \boldsymbol{X}^{[k]} + \boldsymbol{b}^{[k]}\right). \tag{3}$$
For $\boldsymbol{L}^{[k]}$, $k \in \{1, \ldots, K\}$, $\boldsymbol{L}^{[1]}$ is the input layer and $\boldsymbol{L}^{[K]}$ is the output layer. By using (3), the output of a DNN model F can be expressed as
$$F(\boldsymbol{X}) = \boldsymbol{L}^{[K]}\left(\boldsymbol{L}^{[K-1]}(\cdots \boldsymbol{L}^{[1]}(\boldsymbol{X}^{[1]}))\right) \tag{4}$$
where $\boldsymbol{X}^{[1]}$ is the initial input to $\boldsymbol{X}$. To achieve the above first purpose, the surrogate classifier is initially trained on a mixed dataset comprising both original malicious and benign samples in order to learn its classification model F_C, as defined in (4). Accordingly, the loss function for training examples and their corresponding labels $(\boldsymbol{X}, \boldsymbol{y})$ is formulated as
$$\mathcal{L}(\Theta) = \mathbb{E}_{x,y \in (\boldsymbol{X},\boldsymbol{y})}\left[\ell((F_C(x), y))\right] \tag{5}$$

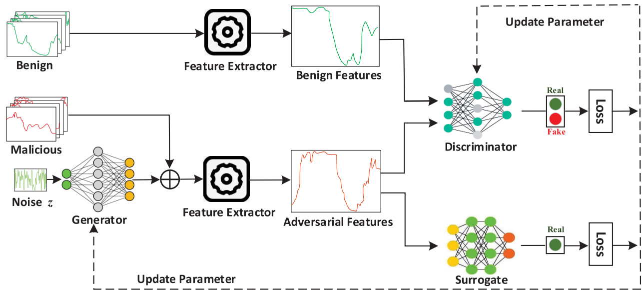


Fig. 5. Architecture of PS-WGAN.

where ℓ is the cross-entropy function that measures the discrepancy between the classifier's predictions and desired labels. Through backpropagation, the surrogate classifier C learns its model F_C by minimizing the loss function defined in (5). Subsequently, to fulfill the second purpose, the trained model F_C processes adversarial feature vectors f^*, which are generated by generator G (as described in the next part). This allows the model to assess whether the adversarial features are classified as malicious, thereby facilitating the iterative refinement of evasion examples. Formally, we denote $\mathcal{L}_C$ as the classification loss, representing the discrepancy between the predicted label of f^* and the target label, which is defined by

$$\mathcal{L}_C = \mathbb{E}_{f^* \in f_{\text{adv}}}\left[\ell\left(F_C(f^*), y^*\right)\right] \tag{6}$$

where f_{adv} is the adversarial feature set extracted by the extractor and y^* represents the target label of the generated samples. As we aim to produce adversarial malicious samples that evade detection and are classified as normal, y^* is set to 0 by default. Hence, the surrogate classifier C provides gradient loss information for training the generator.

Generator: We also use a DNN to describe the generator. The generator G aims to generate noise vectors containing meta-information (i.e., IAT, PS, and PL), which are further processed to produce adversarial feature vectors namely crafted adversarial traffic in feature space. It takes noise vectors, z from the Gaussian distribution $p_z(z) \sim \mathcal{N}(0, 1)$ as input and then outputs perturbation vectors $\widehat{\mathbf{P}}$, where $\widehat{\mathbf{P}} = G(z)$. These vectors are recovered by traffic deparser $\mathbf{R}$ and added to the metadata vectors $\mathbf{P}$ to generate adversarial traffic $\mathbf{P}^*$. The vectors $\mathbf{P}^*$ are then fed into a feature extractor E, ultimately generating adversarial feature vectors f^*. This process can be formulated as

$$f^* = E\left(\mathbf{R}(\widehat{\mathbf{P}}) \oplus \mathbf{P}\right) = E\left(\mathbf{P}^*\right) \tag{7}$$

where $\mathbf{P}^* = \mathbf{R}(\widehat{\mathbf{P}}) \oplus \mathbf{P}$. Here, symbol $\oplus$ represents the adding operation. The loss function of the generator then can be formally expressed as

$$\begin{aligned}\mathcal{L}_G &= -\mathbb{E}_{f^* \in f_{\text{adv}}}\left[D(f^*)\right] + \mathcal{L}_C \\ &= -\mathbb{E}_{f^* \in f_{\text{adv}}}\left[D(f^*)\right] + \mathbb{E}_{f^* \in f_{\text{adv}}}\left[\ell\left(F_C(f^*), y^*\right)\right].\end{aligned} \tag{8}$$

The second term in (8) ensures that the generated adversarial data is identified as the target label (i.e., benign) by the surrogate classifier C. Minimizing (8) guarantees that the generated traffic not only mimics the distribution of benign traffic but also closely aligns with adversarial features in feature space.

Algorithm 1 Training of PS-WGAN

Require: Malicious training dataset $\mathbf{D}$, batch size b, training epoch T, random noises z, the number of discriminator iterations per generator iteration n_D, surrogate model C

Ensure: Generator G

- $\mathbf{P} \leftarrow$ TrafficParser $\mathbf{V}(\mathbf{D})$;
- Initialize a generator G and a discriminator D;
- **for** $t = 1$ to T **do**
  - **for** $j = 1$ to n_D **do**
    - Sample a batch of noise samples $\{z_i\}_{i=1}^b$;
    - Obatin a batch of adversarial feature samples $\left\{f_i^*\right\}_{i=1}^b$ by computing $\mathbf{P}^*$;
    - Sample a batch of normal feature samples $\{f_i\}_{i=1}^b$;
    - Update the weights ω_D of D by descending the loss function $\mathcal{L}_D$: $\omega_D \leftarrow$ Adam $(\nabla_{\omega_D} \frac{1}{b} \sum_{i=1}^b \mathcal{L}_D)$;
  - **end for**
  - Sample a batch of noise samples $\{z_i\}_{i=1}^b$;
  - Update the weights ω_G of G by descending the loss function $\mathcal{L}_G$: $\omega_G \leftarrow$ Adam $(\nabla_{\omega_G} \frac{1}{b} \sum_{i=1}^b \mathcal{L}_G)$;
- **end for**
- **return** Generator G.

Discriminator: Similarly, describe the discriminator D as a DNN neural network, which serves the purpose of assessing the authenticity of the generated adversarial features. The corresponding DNN takes both benign vectors f and adversarial vectors f^* produced by the extractor and generator as inputs, and outputs the probability of whether the input vectors are malicious. Through iterative learning, D can improve its ability to discriminate generated samples from G. Hence, the loss function of discriminator $\mathcal{L}_D$ is formalized as

$$\begin{aligned}\mathcal{L}_D = &-\mathbb{E}_{f \in f_{\text{ben}}}[D(f)] + \mathbb{E}_{f^* \in f_{\text{adv}}}\left[D\left(f^*\right)\right] \\ &+ k\mathbb{E}_{\tilde{f} \sim p(u)}\left[\left\|\nabla_{\tilde{f}} D\left(\tilde{f}\right)\right\|^m\right]\end{aligned} \tag{9}$$

where f_{ben} represents the feature set extracted from previously collected benign traffic, and f_{adv} refers to the adversarial feature set extracted by the extractor. $\tilde{f}$ is the linear interpolation between benign and adversarial samples (i.e., the corresponding distribution, $p(u)$). The third term in (9) acts as a regularization term to enforce the Lipschitz constraint on D. Minimizing (9) ensures that the generated samples more closely resemble normal benign data in feature space. Refer to [28], we set that $k = 2$ and $m = 6$ to yield the optimal results.

Training Phase: The detailed training of PS-WGAN is illustrated in Algorithm 1, which primarily relies on the iterative training of discriminator D and generator G. The Adam optimizer is used to update parameters. To enhance the quality of generated traffic, we iterate D n_D times for each iteration of G. The malicious packets are initially collected and vectorized. Then, the perturbation noise generated by G is added to the malicious vectors and recovered to the corresponding packets. In line 6, the generated adversarial feature vectors f^* are calculated by (7). In line 8, D tries to distinguish between normal and adversarial vectors in feature space by using (9). In line 11, G tries to confuse discriminator D and surrogate model C according to (8). At the end of training, only the generator is retained to produce adversarial traffic during execution.

Execution Phase: In this process, the well-trained generator can be utilized to generate data samples with a batch number of malicious data. Namely, the PS-WGAN generator produces packet-level data while maintaining the original sequence of packets, thereby enhancing the fidelity of adversarial traffic and improving its applicability across various feature types and DL models. The generated adversarial traffic will be sent to the targeted network to evade detection by DL-based NIDSs.

IV. Experiment, Evaluation, and Defense

In this section, we present the experiments results to demonstrate the performances of the proposed PATG and defense schemes. All experiments are conducted on a PC equipped with an NVIDIA RTX 4060 GPU, an Intel Core i7@2.40GHz CPU, 16GB of RAM, and the 64-bit operating system. All methods are implemented by Python 3.8, Scikit-learn 1.5.2, and PyTorch 1.12.

A. Experimental Settings

Datasets Description: For the evaluation and analysis of the proposed PATG, we use two up-to-date representative public real-world traffic datasets, namely Kitsune [8] and CICIoT23 [34]. Both benchmarks have been utilized in related studies [13], [35], [36]. The Kitsune dataset is used to evaluate the Kitsune (i.e., the autoencoder-based model) by actively executing a series of attacks within its video surveillance network. The CICIoT23 dataset, on the other hand, is a real-time and publicly available collection of traffic data from 105 IoT devices (e.g., doorbell, monitor, water sensor, and camera) subjected to common attacks in real IIoT networks. The traffic data is provided in PCAP file format and can be retrieved in isolated network environments. Table II summarizes the six well-known attack datasets used in this study. Notably, to better reflect the attacker's capabilities, the size of the selected training dataset is intentionally kept significantly smaller than the test dataset size.

DL-Based NIDSs: With an advanced feature extraction method, PATG is extensively evaluated on various widely-used DL classifiers, including CNN, GRU, LSTM, CNN-GRU (C-GRU), CNN-LSTM (C-LSTM), Bidirectional GRU (BiGRU), Bidirectional LSTM (BiLSTM), and Kitsune. These models and extractor have been widely employed in previous flow-level anomaly/intrusion detection studies.

Feature Extractor: To achieve finer granularity in predictions, we employ AfterImage [8], a packet-based extractor in Kitsune, as feature extraction in the target model. AfterImage processes nine key attributes for each packet (e.g., MAC, IP, and protocol attributes for both source and destination, as well as IP type, packet size, and timestamp) to extract features across five different time windows. Modifying a single attribute in traffic space impacts multiple features in feature space. PATG leverages the fact that NIDSs are trained exclusively on benign traffic and adjusts malicious traffic in traffic space accordingly. In the following experiments, we evaluate this weakness for each DL-based NIDS individually.

TABLE II
Datastet Used in Experiments

Dataset	Attacks	Testing (Malicious)	Training
Kitsune Dataset	SSL DoS	10,000 (7,500)	4,000 (2,000 benign & 2,000 malicious)
	Botnet	14,000 (10,000)	
	Fuzzing	20,000 (14,000)	
CICIoT23	OS Scan	10,000 (3,000)	
	Backdoor	12,000 (6,000)	
	DDoS	10,000 (9,000)	

Implementation Details: The proposed PATG is trained utilizing Adam optimizer with the learning rate 0.0002, $\beta_1 = 0.5$, and $\beta_2 = 0.999$. The dimension of noise space z, batch size b, training epoch T, and the number of discriminator iterations per generator iteration n_D are set to 64, 256, 200, and 5, respectively. For PS-WGAN, the discriminator is a three-layer DNN with 128, 64, and 32 units in each layer, using the LeakyReLU function. The generator is a four-layer DNN with 64, 128, 512, and 1024 units, incorporating BatchNorm and ReLU function (except for the last layer), while the surrogate model is a three-layer DNN with 128, 64, and 32 units utilizing ReLU and dropout rate 0.5. By default, we use C-LSTM model as the targeted NIDS due to its strong generalization capability.

Evaluation Metrics: For the evaluation of usability, we consider three types of indicators for assessing the proposed PATG and DL-based NIDSs.

1) *Evasive Effectiveness:* To evaluate the impact of the proposed attack method, we employ two metrics introduced by IDSGAN [26], namely, adversarial detection rate (ADR) and EIR. The *ADR* represents the proportion of adversarial malicious samples correctly classified by the targted NIDS among all adversarial attack samples. The *EIR* quantifies the increase in the detection rate of adversarial samples ADR, relative to the detection rate of original malicious samples $\mathrm{DR}_{\mathrm{ori}}$. ADR and EIR metrics are defined as

$$\mathrm{ADR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{10}$$

$$\mathrm{EIR} = 1 - \frac{\mathrm{ADR}}{\mathrm{DR}_{ori}} \tag{11}$$

where *TP* and *FN* denote true positive and false negative, respectively. Hence, the goal of PATG is to achieve a lower *ADR* and a higher *EIR* against the NIDSs.

2) *Field Similarity:* Field similarity evaluates the overall distributional similarity between corresponding packet fields in real and generated adversarial data. For numerical fields, such as packet length, the Kolmogorov–Smirnov statistic is employed to quantify the maximum deviation between the cumulative distribution functions (CDFs) of the real data and generated data.

TABLE III
EVASIVE EFFECT OF DL-BASED NIDSS AGAINST DOS ATTACKS IN KITSUNE DATASET(%)

DL Classifier	Original Detection A	P	R	F1	Evasive Metric ADR	EIR
CNN	99.14	99.97	98.88	99.42	31.64	68.00
GRU	98.59	99.97	98.15	99.05	25.04	74.49
LSTM	98.59	99.97	98.15	99.05	25.47	74.05
C-GRU	99.91	99.95	99.93	99.94	18.17	81.81
C-LSTM	99.88	99.96	99.88	99.92	12.45	87.53
BiGRU	98.60	99.96	98.17	99.06	26.41	73.10
BiLSTM	98.61	99.97	98.17	99.06	25.71	73.82
Kitsune	97.01	99.97	97.36	98.65	17.44	82.09

TABLE IV
EVASIVE EFFECT OF DL-BASED NIDSS AGAINST FUZZING ATTACKS IN KITSUNE DATASET(%)

DL Classifier	Original Detection A	P	R	F1	Evasive Metric ADR	EIR
CNN	76.64	78.08	92.61	84.73	0.53	99.43
GRU	72.70	74.69	92.25	82.55	0.53	99.43
LSTM	72.81	74.59	92.74	82.68	0.51	99.45
C-GRU	72.84	75.35	90.96	82.42	0.54	99.41
C-LSTM	73.16	76.96	87.99	82.11	3.92	95.54
BiGRU	72.87	75.23	91.30	82.49	0.53	99.42
BiLSTM	73.02	75.10	91.94	82.67	0.54	99.42
Kitsune	74.19	79.98	87.42	85.06	3.76	95.70

TABLE V
EVASIVE EFFECT OF DL-BASED NIDSS AGAINST BACKDOOR ATTACKS IN CICIOT23 DATASET(%)

DL Classifier	Original Detection A	P	R	F1	Evasive Metric ADR	EIR
CNN	82.26	76.93	92.15	83.86	54.87	40.45
GRU	74.68	68.66	90.82	78.20	58.45	37.03
LSTM	77.44	70.88	93.17	80.85	54.88	41.09
C-GRU	79.72	75.65	88.65	81.64	51.92	41.43
C-LSTM	80.84	75.58	91.12	82.63	50.10	45.02
BiGRU	73.48	66.99	92.60	77.74	54.74	40.89
BiLSTM	77.14	70.59	93.07	80.28	52.78	43.29
Kitsune	78.54	73.38	91.32	81.37	51.64	43.45

TABLE VI
EVASIVE EFFECT OF DL-BASED NIDSS AGAINST DDOS ATTACKS IN CICIOT23 DATASET (%)

DL Classifier	Original Detection A	P	R	F1	Evasive Metric ADR	EIR
CNN	93.54	98.27	99.49	96.34	21.42	77.35
GRU	94.08	99.78	93.63	96.61	20.41	78.20
LSTM	94.94	99.79	94.58	97.11	20.07	78.78
C-GRU	93.81	99.64	93.46	96.45	18.32	80.39
C-LSTM	93.62	99.73	93.17	96.34	10.76	88.46
BiGRU	94.66	99.77	94.51	97.07	23.66	74.97
BiLSTM	94.81	99.70	94.52	97.04	23.52	75.11
Kitsune	94.03	99.57	93.74	96.56	21.16	77.43

3) *Detection Performance:* We additionally utilize four standard metrics, accuracy (A), recall (R), precision (P), and F1-score (F1), to comprehensively measure the original detection performance of the DL-based NIDSs.

Baseline Methods: To highlight the improvements of PATG, we compare it with three closed-box adversarial traffic generation methods from prior studies as baselines. These include two approaches for modifying packet-level data (i.e., independent packets within the data) and one method for sequential data modification.

1) *Random Extending Interval Timing (RET) [18]:* RET randomly extends the interval time between packets. Noted that, random traffic modifications are not weak or meaningless attacks [17], [18].
2) *Random Duplicating Packtes (RDup) [18]:* RDup randomly duplicates portions of the original traffic (i.e., 5% of original packets). Other methods, such as deleting or reordering packets, are not considered, as we find that these methods can disrupt the functionality.
3) *Timing-Based Adversarial Network Traffic Reshaping Attack (TANTRA) [22]:* TANTRA is the state-of-the-art closed box method against NIDSs. It predicts and reshapes the timestamps of subsequent attack packets by exploiting an LSTM neural network to learn the IAT between normal packets.

B. Evasive Effect of Different Attacks

We evaluate the evasive effects of the proposed adversarial traffic on nine DL-based NIDSs under a GRA scenario (i.e., with full feature knowledge) across different traffic sets. The results for DoS, Fuzzing, Backdoor, and DDoS are presented in Tables III–VI,[1] respectively. The results indicate that after applying the adversarial traffic generated by our method to each malicious dataset, the network traffic successfully evades detection by the DL-based NIDSs. Specifically, the proposed attack achieves an EIR exceeding than 50% in four out of six traffic sets. In the absence of PATG, various DL models effectively detect most malicious attacks. However, after generating adversarial traffic, PATG nearly conceals the attacks entirely, with the highest EIR exceeding 99% in the case of Fuzzing. Although the ADR for most attacks remains below 40%, the Backdoor attack is a notable exception. In this case, the original detection rate decreases significantly from 91.12% to 45.75%, which is two to ten times higher than the ADR observed for other attacks. A possible explanation for this outcome is that the malicious features of Backdoor attacks are inherently distant from the benign space, limiting the attacker's capability to modify them. Additionally, Backdoor often exhibits more fixed traffic fields (e.g., specific port numbers or packet lengths), which are frequently present in the dataset and thus more easily identifiable by DL-based NIDSs.

In addition, the detection performance of different DL models on adversarial traffic varies significantly. Among the nine evaluated DL models, C-LSTM exhibits strong original detection performance across most traffic sets. However, the proposed adversarial traffic exhibits a high escape ability against the C-LSTM model. This can be attributed to the complexity of the C-LSTM architecture, making it susceptible to overfitting specific feature patterns (e.g., spatial and temporal) present in the training data. When adversarial traffic is generated using PATG, these features can be carefully

[1]The analysis of the remaining two traffic attacks are omitted due to space constraints, as the analysis of the remaining four presented attacks sufficiently supports the same conclusions.

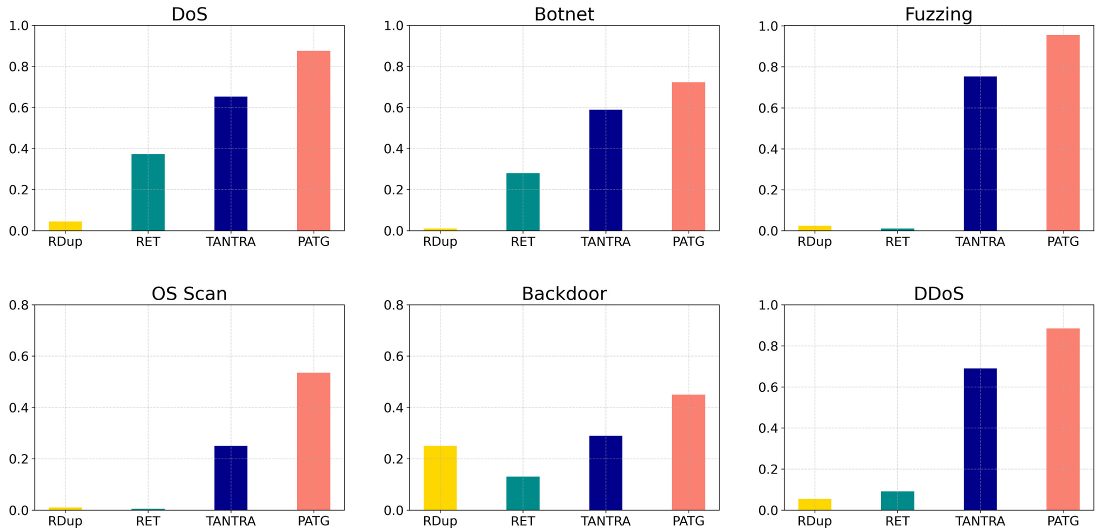
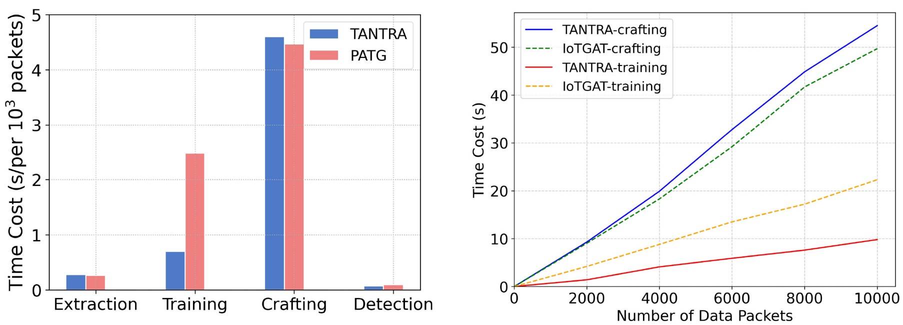


Fig. 6. EIR results of the proposed PATG in comparison to baseline methods.



(a) (b)

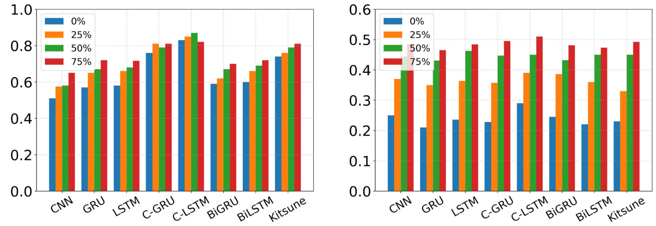
Fig. 7. Comparison of time costs. (a) Execution time on different tasks. (b) Execution time on different numbers of packets.

impacted by PATG, thereby undermining the C-LSTM's ability to effectively detect adversarial traffic. In contrast, other DL models such as CNN and GRU, exhibit weaker detection performance but achieve lower EIR values. This is likely due to their relatively simpler architectures, which provide better generalization capabilities for detecting adversarial traffic compared to C-LSTM.

To validate the malicious functionality of the generated traffic, we utilize Docker as the testing platform and employ Tcpliverreplay to replay both the original and generated attack traffic. Our results show that the adversarial traffic basically retains its malicious functionality. For example, in the generated Fuzzing traffic, we successfully simulate the size of packets including payload, with only a slight decrease in attack time (i.e., from 23.95 to 22.56 s). However, the rate of change (i.e., 5%) in such attack time remains in a controllable range.

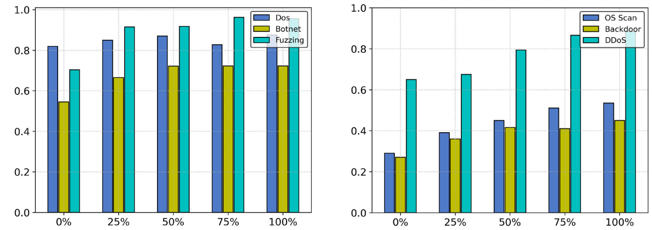
C. Comparison of Different Attack Methods

We perform experiments to compare the evasive effectiveness of the proposed PATG with three baseline approaches. Fig. 6 presents the EIR results (higher is better) for different attack methods across three types of malicious traffic, respectively. The results indicate the proposed PATG achieves remarkable evasion performance compared to other methods. Specifically, the performance of RDup is consistently poor, while RET exhibits evasive effects only in a few cases. This is because random modification methods only work on specific types of malicious traffic, leading to highly unstable evasive capabilities. TANTRA typically requires learning the history timestamp information of a large amount of normal traffic for modification. However, it is difficult for attackers to do this in practice. Moreover, to assess the execution cost of the PATG, particularly for resource-constrained attackers, we utilize the state-of-the-art TANTRA as a baseline method to represent the lower bound of time cost. Fig. 7 shows the comparison of the time costs for different execution tasks. Specifically, Fig. 7(a) presents the execution time on different execution tasks during one epoch for processing 1000 packets. In Fig. 7(a), extraction means the process of feature extraction, crafting denotes the process of generating adversarial traffic, and detection means the process of intrusion detection. It is shown that except for the training process, our PATG method can perform various tasks at a fast speed. PATG involves feature extraction in each training process, so the time consumption of training is relatively high. We further analyze the impact of variations in the total number of network packets on execution time. As depicted in Fig. 7(b), the execution time for both methods



(a) (b)

Fig. 8. EIR results of adversarial traffic under different BRA (i.e., partial feature knowledge) for different DL models. (a) DoS. (b) OS scan.



(a) (b)

Fig. 9. EIR results of different BRA (i.e., partial feature knowledge) compared to GRA (i.e., full feature knowledge) across two datasets. (a) Kitsune. (b) CICIoT23.

increases with the network packets growth in volume. Notably, the proposed PATG retains an affordable overhead cost compared to TANTRA. This observation validates the capability of the proposed PATG to maintain high efficiency in attacking NIDSs.

D. Adversarial Attacks With Limited Feature Knowledge

In this section, we consider the GRA scenario, where the attacker has full feature knowledge of the targeted NIDSs. Next, we extend our analysis to the BRA scenario, which assumes limited knowledge of the targeted NIDSs. Specifically, we examine four types of attackers with varying levels of feature knowledge: 0%, 25%, 50%, and 75% of the features utilized by NIDSs. As noted in Section II-B, the primary distinction between GRA and BRA (100%) lies in the substitute feature extractor employed by the attacker. In the BRA scenario, attackers with partial feature knowledge rely on extracting commonly used features, while those with no knowledge (BRA 0%) simulate the feature extractor using unrelated features.

Fig. 8 presents the EIR results of proposed adversarial traffic on two traffic datasets across different DL models under conditions of incomplete feature extraction. Fig. 8 reveals that as the proportion of feature knowledge declines, the EIR of DL models correspondingly decreases. Notably, the reduction in EIR from 25% to 0% is most significant, primarily because PATG struggles to accurately learn the true feature distribution when in the absence of any feature knowledge. Moreover, across various BRA conditions, the C-LSTM consistently exhibits a relatively higher EIR compared to other DL models. This trend aligns closely with observations from the GRA scenario (i.e., full feature knowledge), indicating that C-LSTM demonstrates less robustness than the other models.

Moreover, we utilize C-LSTM as the targeted model and evaluate the EIR of four GRA and BRA scenarios across two datasets, as illustrated in Fig. 9. Notably, BRA (50%) and BRA (75%) achieve EIR levels comparable to those observed under GRA. Remarkably, even attackers with no prior knowledge of the target system's features (BRA 0%), the proposed PATG still demonstrates notable evasion capability. A critical observation is that, despite the absence of precise information regarding the features utilized by the NIDSs, attackers can effectively generate evasion variants using simulated feature computations. This finding underscores a significant concern for DL-based NIDSs, indicating that even attackers with minimal information can successfully render a considerable proportion of malicious traffic evasive with relative ease.

TABLE VII
EVASIVE EFFECT OF PATG AGAINST RECONNAISSANCE AND RANSOMWARE ATTACKS IN TON_IOT DATASET (%)

Attack Type	DL Classifier	Original Detection (F1)	ADR	EIR
Reconnaissance	CNN	98.22	21.73	78.31
	GRU	97.83	19.46	80.53
	LSTM	98.54	18.16	81.78
	C-LSTM	99.12	12.48	87.64
	Kitsune	96.38	24.19	75.85
Ransoware	CNN	95.61	33.86	66.08
	GRU	94.75	29.68	70.32
	LSTM	95.03	28.53	71.46
	C-LSTM	96.86	26.78	74.24
	Kitsune	93.54	36.20	63.79

E. Fidelity of Data Distribution

To verify the validity of the data generated by the PATG traffic generator (i.e., PS-WGAN), we evaluate the field values of real traffic and adversarial traffic across different attack types. The CDFs of these field values are then analyzed qualitatively to assess their similarities and differences. Our analysis focuses on two critical packet fields: 1) payload length (PL) and 2) IAT, which are widely utilized by traffic analysis systems [37]. Figs. 10 and 11 present the CDF results for the Kitsune and CICIoT23 datasets, respectively. The adversarial traffic generated by PATG (represented by the light green curve) exhibits a trend closely aligned with that of real traffic (represented by the blue curve). For example, the CDF curves for Fuzzing and DDoS attacks show significantly similarity to those of real traffic. Notably, a recurring pattern in the real traffic data reveals alternating steep and flat segments in the CDF curves, indicating that a substantial proportion of field values are concentrated within distinct narrow ranges. Our findings demonstrate that PATG effectively replicates this pattern, largely due to its generator's ability to simplify the learning process of traffic patterns by modifying the traffic space to induce corresponding changes in the feature space. These results highlight the high fidelity of the adversarial traffic generated by PATG.

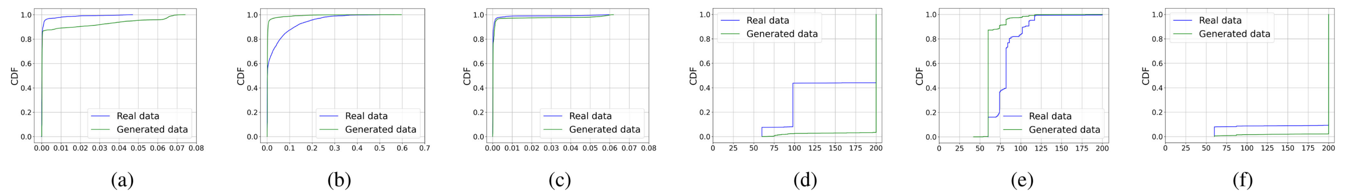


Fig. 10. Distribution of real and generated IAT and PL in Kitsune dataset. (a) DoS-IAT. (b) Botnet-IAT. (c) Fuzzing-IAT. (d) DoS-PL. (e) Botnet-PL. (f) Fuzzing-PL.

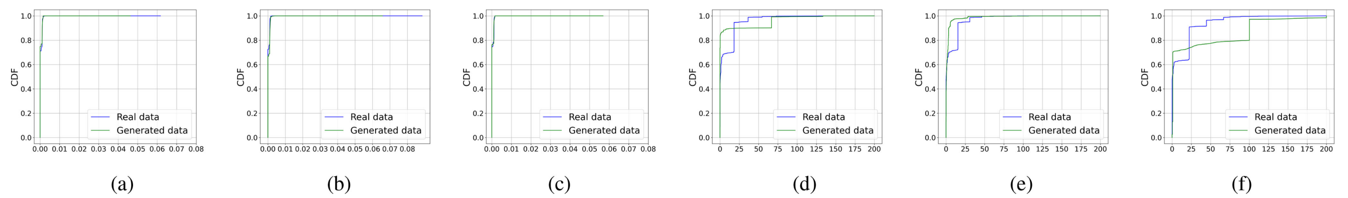


Fig. 11. Distribution of real and generated IAT and PL in CICIoT23 dataset. (a) OS Scan-IAT. (b) Backdoor-IAT. (c) DDoS-IAT. (d) OS Scan-PL. (e) Backdoor-PL. (f) DDoS-PL.

F. Generalizability Analysis of PATG

To rigorously assess the generalizability of PATG across diverse attack scenarios and IIoT environments beyond the initially tested six attack types (e.g., DDoS and Botnet), we conduct supplementary experiments using an additional publicly available IIoT dataset, namely ToN_IoT [38]. These experiments focus on Reconnaissance (e.g., port scanning) and Ransomware attacks. The selected attack categories are intended to evaluate PATG's adaptability to distinct traffic characteristics: Reconnaissance attacks are characterized by systematic probing using high-frequency, low-payload packets, whereas Ransomware traffic typically involves encrypted communication patterns with strict protocol compliance. The evasive performance of PATG against these attacks is summarized in Table VII.

As illustrated in Table VII, PATG demonstrates strong evasion capabilities against Reconnaissance attacks, achieving an average EIR of 80.82% across DL-based NIDSs. Notably, the C-LSTM classifier exhibits the highest EIR degradation (from 99.12% to 12.48%), as PATG's ability to disrupt the periodicity of scanning patterns by perturbing IATs and protocol headers. For Ransomware attacks, PATG attains an average EIR of 69.18% by exploiting protocol-header modifications (e.g., TCP/UDP alternation) and adjusting PSs within MTU constraints to mask encrypted flow. This is exemplified by a reduction in the ADR of the C-LSTM classifier from 96.19% to 26.78%. Although the presence of encryption imposes limitations on spatial feature perturbations, PATG's effectiveness varies across different model architectures, with simpler models like CNN showing lower EIR due to their reliance on relatively static spatial features (e.g., packet length). These results demonstrate PATG's adaptability to attacks with distinct traffic characteristics.

In summary, PATG demonstrates superior evasion effectiveness against volume-based attacks (e.g., DoS and DDoS) and iterative attacks (e.g., reconnaissance and botnet), where prominent and modifiable statistical artifacts such as flow burstiness and packet periodicity are present. Conversely, its performance is limited in the context of low-volume, context-dependent attacks (e.g., SQL injection) and attacks requiring strict protocol adherence (e.g., ransomware and backdoor), where perturbations risk invalidating attack semantics. These findings underscore PATG's suitability for IIoT environments characterized by volumetric and temporal anomalies, while also emphasizing the necessity for adversarial methods to effectively address encrypted or context-sensitive threats.

G. Defending Against Adversarial Attacks

Defensive schemes against adversarial attacks aim to strengthen the robustness of the DL-based models to adversarial samples, making them less susceptible to compromise and reducing the success evasive rate of various attacks. We introduce two defense schemes to mitigate the proposed attack.

AT: Following the study [14], we aim to build a robust classifier by incorporating adversarial information during the training phase. It is widely used to retrain a classifier with correctly labeled adversarial examples to defend against adversarial attacks in the image domain. However, in our traffic space attacks, its effectiveness is limited to reducing the impact of adversarial feature generation. Specifically, we retain the DL-based NIDS model on benign features and malicious features generated by PS-WGAN.

Feature Selection (FS): This critical method in feature engineering involves removing redundant or irrelevant dimensions from the features utilized by the ML/DL models, thereby significantly enhancing detection performance and robustness. To achieve this, we employ the gradient boosting decision Tree (GBDT) model in sklearn library to retain the top 70% most important features for training. Given any input $\mathbf{x}$, the objective of the GBDT is to minimize the following regression function $F(\mathbf{x})$ by an additive expansion way:

$$F(\mathbf{x}) = \sum_{i=0}^{m} \tau_i h(\mathbf{x}; \mathbf{a}_i) \tag{12}$$

where $h(\mathbf{x}; \mathbf{a}_i)$ is the base tree learner with parameters $\mathbf{a}_i$ and τ_i represents its expansion coefficients.

TABLE VIII
CROSS-DATASET EVALUATION ON AT AND FS AGAINST OVERFITTING USING DIFFERENT MITIGATION STRATEGIES

Defense Strategy	Known Attacks (EIR%)	Unknown Attacks (EIR%)	Detetction Rate (DR%)	Inference Time (ms/batch)
Baseline (No Defense)	89.34	73.78	28.24	–
AT	9.17	18.91	84.41	13.84
AT + Regularization	7.82	15.29	86.15	14.28
AT + Data Augmentation	8.06	16.75	85.54	14.51
FS	24.53	32.37	72.12	6.29
FS + Regularization	23.86	30.74	73.52	6.55
FS + Data Augmentation	21.24	28.92	75.26	6.86

We evaluate the proposed two defense methods with no defense scenario. The results of the two defense schemes against the three types of attacks are shown in Fig. 12. Note that, if the ADR exceeds the original value after employing the defense scheme, the EIR is set to 0 (e.g., for Botnet). Clearly, FS provides a limited and unstable defense effect for Botnet, Fuzzing, OS scan, and Backdoor attacks. This is because the selected important features may have little correlation with specific attack fields. In contrast, AT shows better defensive performance, indicating that generating adversarial features can increase the difficulty for attackers to succeed. Since the neural network architecture remains unaltered, both AT and FS can be considered as foundational defense against adversarial traffic in the IIoT environment. These approaches significantly enhance the robustness of DL-based NIDSs against the proposed adversarial traffic. However, AT requires powerful attacks to encompass all potential adversarial sample types, necessitating extensive retraining with additional data—a process that is both time-consuming and computationally expensive. Despite fulfilling these requirements, DL classifiers in NIDSs may still struggle to generalize effectively to novel, unseen adversarial traffic, increasing the risk of overfitting Similarly, FS, while potentially effective, may lead to the loss of crucial information, thereby impairing the ability of NIDSs to detect complex attack patterns.

To mitigate the risk of overfitting in defense schemes, particularly within AT, we implement both regularization techniques and data augmentation as key mitigation strategies. For regularization, we employ early stopping with spectral normalization to prevent the model from overfitting to adversarial traffic patterns. Early stopping is implemented by monitoring validation loss (e.g., using a 15% validation subset) and terminating training once the loss plateaus or increase over a defined threshold (e.g., no improvement for five consecutive epochs) [39]. Concurrently, spectral normalization is applied to the PS-WGAN discriminator to constrain its Lipschitz constant, thereby stabilizing training dynamics and reducing overfitting risk [40]. For data augmentation [41], [42], we simulate natural network variability by introducing temporal jitter (e.g., perturbing IATs by $\pm 10\%$) and feature-level noise (e.g., injecting Gaussian noise with $\sigma = 0.01$ into packet lengths within MTU limits) into benign traffic samples. To evaluate the effectiveness of these mitigation strategies, we conduct cross-dataset evaluation experiments (Kitsune $\rightarrow$ ToN_IoT). Specifically, the defense schemes are trained on adversarial samples generated by PATG targeting known attack types (i.e., Botnet and DoS) and tested on two unseen attack types (i.e., DDoS and Ransomware) from ToN_IoT [38].

TABLE IX
DEFENSE PERFORMANCE UNDER VARYING TRAFFIC LOADS (%)

Defense Scheme	Traffic load	Botnet (EIR Reduction)	DDoS (EIR Reduction)
AT	Low	72.36 → 28.84	89.57 → 34.75
	Medium	72.36 → 31.53	89.57 → 38.21
	High	72.36 → 35.63	89.57 → 42.69
FS	Low	72.36 → 45.59	89.57 → 50.16
	Medium	72.36 → 49.82	89.57 → 54.48
	High	72.36 → 55.56	89.57 → 61.36

Table VIII presents a comparative analysis of overfitting mitigation strategies, where baseline refers to the PATG method without defense, DR denotes the detection rate against previously unseen attacks by the target NIDS model, and inference time indicates the computational cost of the defense methods measured in milliseconds per batch. As shown in Table VIII, AT enhanced with either regularization or data augmentation significantly reduces the EIR for unknown attacks, outperforming both the baseline AT and FS methods. However, AT introduces substantial computational overhead due to the processes involved in generating adversarial examples and retraining the model. As such, it is more appropriate for high-security IIoT scenarios (e.g., healthcare) where robustness is prioritized over latency constraints. In contrast, the integration of FS with data augmentation enables near-real-time inference by selecting important and relevant features. However, this approach may lead to a reduced detection rate, making it be a viable compromise for latency-sensitive scenario (e.g., Internet of Vehicles). These findings highlight a critical tradeoff: while AT emphasizes robustness at higher computational costs, FS prioritizes efficiency with moderate security guarantees.

H. Defense Scheme Evaluation Under Dynamic Environments

To assess the resilience of defense schemes in dynamic IIoT environments, we evaluate the performance of AT and FS schemes under varying network traffic loads (low: 100–500 packets/s, moderate: 500–1k packets/s, and high: 1k–5k packets/s) on the Kitsune (Botnet) and CIC-IoT23 (DDoS) datasets. For each traffic condition, we measure EIR reduction (i.e., variation in EIR before and after applying defenses). The results, presented in Table IX, demonstrate that AT consistently achieves the highest EIR reduction (up to 54% for DDoS) across all traffic conditions, albeit at the expense of increased higher computational overhead due to retraining

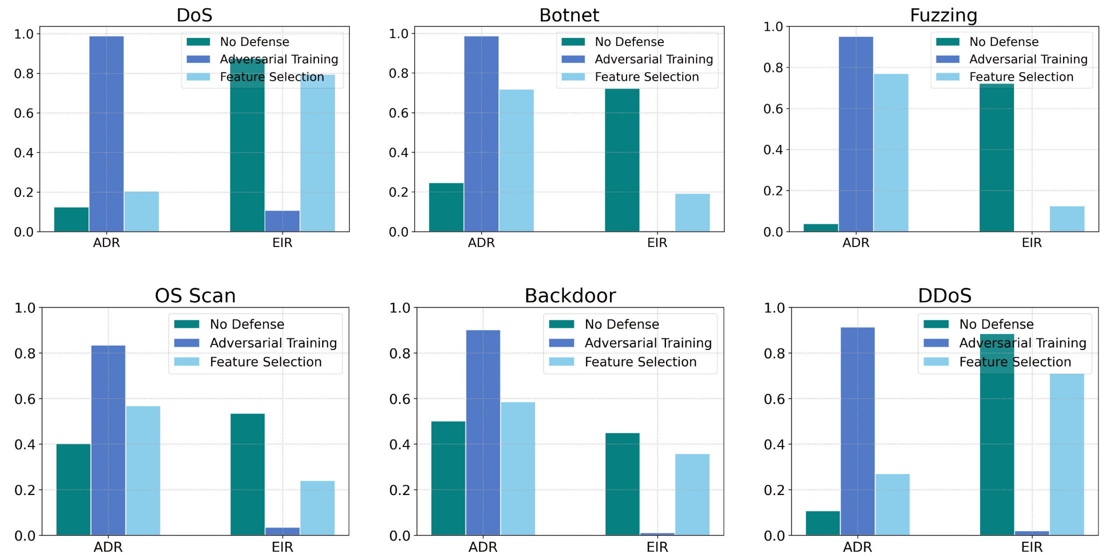


Fig. 12. Defense performance results against proposed adversarial attacks.

with adversarial samples. In contrast, FS significantly reduces training and inference costs by discarding redundant features (e.g., protocol-specific flags), but exhibits a 17%–27% decline in EIR reduction under high traffic loads, primarily due to the loss of critical features. While both defense mechanisms improve the NIDS robustness, their performance is adversely affected under dynamic traffic conditions. These findings underscore the tradeoff between detection effectiveness and computational scalability, emphasizing the need for adaptive thresholding or hybrid methods in resource-constrained IIoT networks.

Furthermore, to comprehensively evaluate defense robustness beyond the PATG, we also extend our validation to include white-box attacks (e.g., fast gradient sign method (FGSM) [14], [43]) on Botnet. FGSM is a single-step white-box attack that generates adversarial examples by perturbing inputs along the gradient direction of the model's loss function. These adversarial examples are generated using the same feature extractor. As illustrated in Fig. 13, AT demonstrates strong cross-attack generalization, achieving a EIR by 51% reduction in the EIR against FGSM and a 42% for closed-box PATG attack. However, the FS proves inadequate against gradient-based attack due to preserved high-dimensional correlations. These findings emphasize that context-aware defense selection is critical for attack-specific resilience.

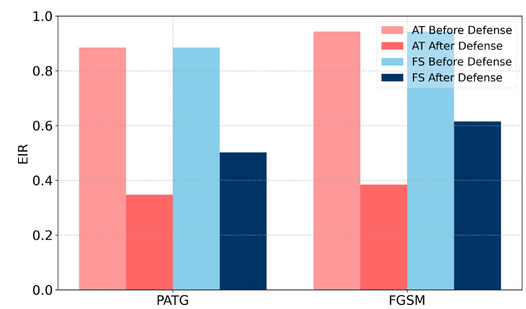


Fig. 13. Defense robustness against different adversarial attack methods.

V. Further Discussion

In this section, we delve deeper into the considerations and potential enhancements of our adversarial traffic attacks.

Impact of Anomaly Detection: The adversarial traffic in PATG is meticulously crafted to evade detection by DL-based NIDSs. However, when deployed in real-world environments, subtle changes introduced to evade detection could lead to inconsistencies in feature distributions or behavioral patterns, which may flag the traffic as anomalous. Anomaly-based detection systems, particularly those leveraging time-series modeling or statistical analysis, might identify semantic or contextual discrepancies between the generated and actual traffic. To address this challenge, advanced traffic generation techniques incorporating large language models, such as transformers [44], can be employed to enhance semantic and contextual alignment with legitimate traffic. By ensuring consistency across network traffic characteristics, these techniques could effectively mitigate the risk of detection.

Attacking Rule-Based NIDSs: While DL-based NIDSs are often regarded as the next generation of intrusion detection tools in commercial settings, traditional rule-based systems, such as Snort and Zeek [29], [30], remain prevalent. These systems function by matching network traffic against predefined rule sets rather than leveraging learned patterns. To generate traffic capable of evading these systems, attackers can exploit publicly available rule sets or infer rules through iterative probing, wherein test traffic is injected, and detection outcomes are analyzed. To enhance the robustness of rule-based systems against such adversarial traffic, the integration of adaptive rule generation mechanisms or hybrid detection approaches that combine DL-based models with rule-based frameworks can be explored.

Stability of Modeling: The effectiveness of adversarial traffic often hinges on its alignment with specific traffic distributions. However, variations in network topology, traffic loads, or the deployment of novel protocols can disrupt this alignment, potentially making adversarial traffic more detectable. Furthermore, changes in protocol constraints or network environments may reduce the compatibility of the generated traffic. To address these challenges, the development of adaptive traffic generation model is essential. Such models should account for evolving traffic distributions and protocol constraints, thereby ensuring sustained compatibility and effective evasion in rapidly changing network environments.

VI. Conclusion

In this article, we have proposed a practical adversarial attack method called PATG, which can modify various malicious traffic to evade DL-based detection with minimal execution cost and limited prior knowledge. For each malicious attack, PATG builds a WGAN-based generator to generate adversarial traffic with domain constraints. We have also conducted attacks on nine DL-based NIDSs using the Kitsune and CICIoT23 datasets. The evaluation demonstrates that adversarial attacks on nine DL-based NIDSs achieve significant evasion rates (>50%) with minimal execution costs and high fidelity, posing serious threats to system integrity in applications such as smart cities and vehicular networks. This study underscores the critical severity of these attacks and investigates AT and FS as potential defense schemes. Future work will address the generation of adversarial attacks in dynamic environments and the development of novel defense strategies for constraint-compliant attacks, utilizing uncertainty-handling techniques.

References

[1] L. Qi, Y. Yang, X. Zhou, W. Rafique, and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6503–6511, Sep. 2022.

[2] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.

[3] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2671–2701, 3rd Quart., 2019.

[4] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3540–3552, 2020.

[5] D. Han et al., "DeepAID: Interpreting and improving deep learning-based anomaly detection in security applications," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Nov. 2021, pp. 3197–3217.

[6] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in *Proc. ACM Southeast Conf.*, Apr. 2019, pp. 86–93.

[7] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37004–37016, 2019.

[8] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Feb. 2018, pp. 1–15.

[9] P. B. Chaluvaraj, R. Vasanthi, G. Sugitha, and S. A. Lakshmi, "Intrusion detection and secure data storage in the cloud were recommend by a multiscale deep bidirectional gated recurrent neural network," *Expert Syst. Appl.*, vol. 255, pp. 1–12, Dec. 2024.

[10] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, May 2017, pp. 39–57.

[11] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *Proc. ACM CoNEXT Workshop Big DAta, Mach. Learn. Artif. Intell. Data Commun. Netw. (Big-DAMA)*, 2019, pp. 14–21.

[12] C. Szegedy et al., "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Represent (ICLR)*, Apr. 2014, pp. 1–10.

[13] M. A. M. Carrasco, C. Wu, and W. Fuertes, "Adversarial examples: A survey of attacks and defenses in deep learning-enabled cybersecurity systems," *Expert Syst. Appl.*, vol. 238, Mar. 2024, Art. no. 122223.

[14] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–11.

[15] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, Feb. 2019, pp. 1–15.

[16] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust PDF malware classifiers," in *Proc. 29th USENIX Security. Symp. (USENIX Securi)*, Aug. 2020, pp. 2343–2360.

[17] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods," in *Proc. WOOT*, 2008, pp. 1–9.

[18] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanácek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach," *EAI Endorsed Trans. Security Safety*, vol. 5, no. 17, pp. 1–15, 2019.

[19] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1962–1976, Jun. 2021.

[20] A. Chernikova and A. Oprea, "FENCE: Feasible evasion attacks on neural networks in constrained environments," *ACM Trans. Priv. Security*, vol. 25, no. 4, pp. 1–34, 2022.

[21] P. Sun, S. Li, J. Xie, H. Xu, Z. Cheng, and R. Yang, "GPMT: Generating practical malicious traffic based on adversarial attacks with little prior knowledge," *Comput. Security*, vol. 130, Jul. 2023, Art. no. 103257.

[22] Y. Sharon, D. Berend, Y. Liu, A. Shabtai, and Y. Elovici, "TANTRA: Timing-based adversarial network traffic reshaping attack," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3225–3237, 2022.

[23] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *Proc. IEEE Int. Symp. Netw. Comput. Appl. (NCA)*, Sep. 2019, pp. 1–8.

[24] B. Zolbayar et al., "Generating practical adversarial network traffic flows using NIDSGAN," 2022, *arXiv:2203.06694*.

[25] O. Ibitoye, M. O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[26] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," in *Proc. Adv. Knowl. Discov. Data Min. PAKDD*, 2022, pp. 79–91.

[27] C. Zhang, X. Costa-Pérez, and P. Patras, "Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms," *IEEE/ACM Trans. Netw.*, vol. 30, no. 3, pp. 1294–1311, Jun. 2022.

[28] J. Wu, Z. Huang, J. Thoma, D. Acharya, and L. V. Gool, "Wasserstein divergence for GANs," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 673–688.

[29] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source IDS? Snort, suricata or zeek," *Comput. Netw.*, vol. 213, Aug. 2022, Art. no. 109116.

[30] A. Muyideen et al., "Machine learning assisted snort and zeek in detecting DDoS attacks in software-defined networking," *Int. J. Inf. Tecnol.*, vol. 16, no. 3, pp. 1627–1643, 2024.

[31] D. Lowd and C. Meek, "Adversarial learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Aug. 2005, pp. 641–647.

[32] K. Roshan and A. Zafar, "Black-box adversarial transferability: An empirical study in cybersecurity perspective," *Comput. Security*, vol. 141, Jun. 2024, Art. no. 103853.

[33] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proc. Int. Conf. Learn. Represen. (ICLR)*, Apr. 2017, pp. 1–24.

[34] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CicIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[35] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 538–566, 1st Quart., 2023.
[36] S. Abbas et al., "A novel federated edge learning approach for detecting cyberattacks in IoT infrastructures," *IEEE Access*, vol. 11, pp. 112189–112198, 2023.
[37] R. Li, Q. Li, Y. Huang, W. Zhang, P. Zhu, and Y. Jiang, "Iotensemble: Detection of botnet attacks on Internet of Things," in *Proc. Eur. Symp. Res. Comput. Security (ESORICS)*, Sep. 2022, pp. 569–588.
[38] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. den Hartog, "Ton_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 485–496, Jan. 2022.
[39] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proc. Mach. Learn. Res.*, Jul. 2020, pp. 8093–8104.
[40] F. Farnia, J. M. Zhang, and D. Tse, "Generalizable adversarial training via spectral normalization," in *Proc. Mach. Learn. Res.*, May 2019, pp. 1–13.
[41] Y. Ma, M. Dong, and C. Xu, "Adversarial robustness through random weight sampling," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2023, pp. 1–13.
[42] X. Jiang et al., "Netdiffusion: Network data augmentation through protocol-constrained traffic generation," *ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 1, pp. 1–32, 2024.
[43] M. L. Naseem, "Trans-IFFT-FGSM: A novel fast gradient sign method for adversarial attacks," *Multim. Tools Appl.*, vol. 83, no. 29, pp. 72279–72299, 2024.
[44] S. Islam et al., "A comprehensive survey on applications of transformers for deep learning tasks," *Expert Syst. Appl.*, vol. 241, May 2024, Art. no. 122666.