Constraining Adversarial Attacks on Network Intrusion Detection Systems: Transferability and Defense Analysis

Nour Alhussien, Ahmed Aleroud, Abdullah Melhem, and Samer Y. Khamaiseh, *Member, IEEE*

***Abstract*—Adversarial attacks have been extensively studied in the domain of deep image classification, but their impacts on other domains such as Machine and Deep Learning-based Network Intrusion Detection Systems (NIDSs) have received limited attention. While adversarial attacks on images are generally more straightforward due to fewer constraints in the input domain, generating adversarial examples in the network domain poses greater challenges due to the diverse types of network traffic and the need to maintain its validity. Prior research has introduced constraints to generate adversarial examples against NIDSs, but their effectiveness across different attack settings, including transferability, targetability, defenses, and the overall attack success have not been thoroughly examined. In this paper, we 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. Our constraints are categorized into four types: feature mutability constraints, feature value constraints, feature dependency constraints and distribution preserving constraints. We evaluated the impacts of these constraints on white box and black box attacks using two intrusion detection datasets. Our results demonstrated that the introduced constraints have a significant impact on the success of white box attacks. Our research revealed that transferability of adversarial examples depends on the similarity between the targeted models and the models to which the examples are transferred, regardless of the attack type or the presence of constraints. We also observed that adversarial training enhanced the robustness of the majority of machine learning and deep learning-based NIDSs against unconstrained attacks, while providing some resilience against constrained attacks. In practice, this suggests the potential use of pre-existing signatures of constrained attacks to combat new variations or zero-day adversarial attacks in real-world NIDSs.**

***Index Terms*—Adversarial attacks, network intrusion detection systems, artificial intelligence, neural networks, computer networks.**

Manuscript received 20 June 2023; revised 10 November 2023; accepted 2 January 2024. Date of publication 22 January 2024; date of current version 12 July 2024. This material is based upon work supported by the National Science Foundation under grant No. 2131538 and grant No. 2306109. The associate editor coordinating the review of this article and approving it for publication was T. Inoue. *(Corresponding author: Ahmed Aleroud.)*

Nour Alhussien, Ahmed Aleroud, and Abdullah Melhem are with the School of Computer and Cyber Sciences, Augusta University, Augusta, GA 30912 USA (e-mail: noalhussien@augusta.edu; aaleroud@augusta.edu; abanimelhem@augusta.edu).

Samer Y. Khamaiseh is with the Computer Science and Software Engineering Department, Miami University, Oxford, OH 45056 USA (e-mail: khamaisy@miamioh.edu).

Digital Object Identifier 10.1109/TNSM.2024.3357316

I. Introduction

DEEP learning (DL) is revolutionizing almost every business sector by enabling the creation of intelligent systems and providing solutions to complex problems, which include defenses against network intrusion. Tasks such as intrusion detection are becoming increasingly complex in today's modern networks due to the ever-changing methods and techniques employed by cyber attackers to circumvent traditional security measures [1].

There is an increasing number of NIDSs that employ DL algorithms for detecting suspicious traffic in network flows [2]. This recent trend of using DL as opposed to traditional ML algorithms is attributed to the limitations of traditional ML algorithms. ML-based intrusion detection approaches are typically rule-based. Even when statistical or feature engineering methods are used for analyzing traffic, they can still lead to a significant percentage of false positives [3]. This also requires a significant monitoring effort by system administrators and security teams to adapt to emerging threats.

However, the increased use of DL algorithms in NIDSs is also associated with the recent threats of adversarial attacks. While DL methods are capable of detecting malicious activities [4], existing research shows that DL approaches are vulnerable to adversarial attacks. In these attacks attackers deliberately craft adversarial examples (AEs) with the goal of misleading IDSs and causing incorrect classification of the incoming traffic [5]. Most studies have been investigating adversarial attacks against DL models in the context of image classification, leading to findings such as the capability of adversaries to arbitrarily perturb image features [6], [7]. Existing works in this area rely on the assumption that image features can be updated without adhering to specific constraints. This is due to the domain itself where images can be perturbed easily [8]. This is not the case for network data where changes are usually bounded by constraints related to the data and the domain itself, where few changes in network features can lead to invalid network flows [7], [8], [9]. The majority of existing research on adversarial attacks on network data have focused on crafting and mitigating adversarial examples without taking into consideration a need of the generated examples to be valid network traffic [10], [11]. It can be easily detected by IDSs otherwise.

Applying constraints when crafting adversarial attacks means that the adversary should consider the generated AEs still meeting the characteristics of the network traffic [12].

Generating AEs using conventional adversarial attacks may lead to invalid network traffic such as generating negative port numbers, a source TCP window advertisement value with a value greater than 255 or setting a DNS service to a TCP protocol. Although most of the existing works have reported high attack success rates [13], [14], they do not consider such domain constraints. However, with the recent trend of systems detecting AI-generated data, attackers are expected to adapt to such changes to avoid detection of AEs. Therefore, it is quite significant to study the behavior of network Adversarial attacks under the contextual constraints of a valid traffic. Such a contextual analysis can lead to create new defense approaches to mitigate those attacks.

While there is a recent growing body of work that explores the impact of constrained adversarial attacks on NIDSs [15] [16] [8], there are many limitations in the existing methods:

1) Most of the existing research does not comprehensively investigate the impact of constraints under different threat models, attack classifications, and characteristics. There are different classifications of attacks which include targeted and untargeted attacks, black box (BB) and white box (WB) attacks. It is then necessary to investigate whether applying generally crafted constrained AEs will lead to similar/different decisiveness behavior when applied to different attacking methods. While existing research indicates that network constraints may not inherently improve robustness against AEs [8], a comprehensive examination is essential, considering the varieties of adversarial attacks, the models they target, and the complexity of the DL architectures involved. It's also critical to assess whether implementing constraints as part of an adversarial training defense mechanism can improve resilience against AEs [17], and if such defenses are transferable between DL and ML) models.
2) While constraints may enhance the robustness of some adversarial attacks against a specific model, their effectiveness and transferability across different models or architectures need to be studied. Constrained adversarial attacks on a specific DL model may not be necessarily effective against other models, making them less practical in the real-world NIDSs.
3) Existing works that develop constraints do not consider relationships/dependencies between traffic features. Some of those dependencies are statistical and others are semantic-based dependencies. By overlooking the relationships between traffic features when creating AEs, the generated traffic may not accurately reflect the complex interdependencies and interactions among the features in the original data.

This paper presents an approach for crafting and implementing constraints to generate AEs against NIDSs. We adopted a new set of network constraints while generating AEs to produce valid network traffic posing a real threat to NIDSs. We conducted evaluations to assess the impact of the network-constrained AEs on the success rates of both BB and WB attacks and transferability of AEs. Additionally, we tested the application of our constraints in enhancing adversarial training to create defense mechanisms against sophisticated adversarial attacks targeting NIDSs.

The contributions of the paper can be summarized under the following three research areas:

1) *Adversarial domain constraints:* We proposed a new set of network domain constraints that can be applied to network data. Not only those constraints preserve characteristics such as feature ranges, but also the semantic-based and statistical relationship between features after the generation of AEs. We then tested the validity of the generated AEs within the network domain both before and after applying the created constraints.
2) *Transferability analysis:* In addition to evaluating the transferability of constrained AEs, we extended this analysis to include transferability of defenses. This comprehensive evaluation provides valuable insights into the generalization capabilities of both attacks and defense mechanisms across different datasets and models.
3) *Analysis of transferability and defense using similar and different attack signatures:* The paper investigates the effects of the proposed adversarial domain constraints on transferability of AEs between similar/different source and target DL/ML models. Using the observed variations in the signatures of the generated AEs, we investigated whether training a model on AEs from a specific attack could be effective in detecting AEs from a different attack. By considering a range of model complexities, including both basic and intricate architectures, we provide a comprehensive analysis of the constraints' applicability and the transferability of attacks and defenses across various model setups.

The rest of this paper is organized as follows: Section II discusses the existing research that is relevant to our work. Section III delves into the threat models investigated in our research. Section IV introduces our methodology. Section V discusses the results of our experiments. The research is concluded in section VI. Table I summarizes the symbols and abbreviations used in the related work and methodology sections.

II. Related Work

General Background: Adversarial attacks can be classified into two main categories: BB and WB attacks. This categorization depends on the extent of the attacker's knowledge of the targeted DL/ML model and data [29]. In BB adversarial attacks, the attacker does not have access to the internal workings of the model. She/he may only have partial knowledge about the input features or feature vectors and output, which correspond to the predictions made by the model [30]. WB adversarial attacks occur when the attacker has full access to the internal workings of the model, including the model architecture and parameters [12]. WB attacks tend to be more effective compared to BB attacks. This is because in WB attacks the attacker has more information about the model. With this increased knowledge, the attacker can craft fine-tuned AEs [31].

TABLE I
LIST OF ABBREVIATIONS

Acronym	Meaning	Ref
FGSM	Fast Gradient Sign Method	[18]
MI-FGSM	Momentum Iterative Fast Gradient Sign Method	[19]
JSMA	Jacobian-based Saliency Map Attack	[20]
EAD	Elastic-Net Attack with Directional Gradients	[21]
BIM	Basic Iterative Method,	[22]
PGD	Projected Gradient Descent	[23]
MLP	Multilayer Perceptron	[24]
ResNet	Residual networks	[25]
PSO	Particle Swarm Optimization	[26]
GA	Genetic Algorithm	[27]
STTL	Source to Destination Time to Live	[28]
TCPRTT	Sum of synack and ackdat	[28]
SYNACK	The time between the SYN and the SYNACK packets	[28]
SINPKT	Inter-packet arrival time from source to destination	[28]
SLOAD	Source to destination bits/s	[28]

The effectiveness of existing adversarial attacks can be determined by the attack success rate and magnitude of perturbations introduced [32]. The attack success rate refers to the percentage of AEs that are successfully misclassified by the model. The magnitude of perturbations indicates the extent of noise introduced to the original input to ensure they are not easily detectable by the targeted DL/ML models [33]. Success of AEs can be also determined based on their transferability [34]. Transferability measures the capacity of an attack to successfully fool other models differing from the one specifically designed to evade.

Compared to WB attacks, the success of BB attacks can be also determined by their computational efficiency [35]. Efficiency refers to methods that minimize the number of queries or interactions with the model to craft AEs [36]. Therefore, researchers employed several query optimization techniques for enhancing the efficiency of BB adversarial attacks, including: zeroth-order optimization [37], BB gradient estimation [38], and evolutionary algorithms [15].

Since the seminal work of Szegedy et al. [24] who discovered that DL/ML models are vulnerable to adversarial attacks, it became apparent that even the most sophisticated DL models can be deceived when targeted by a well-crafted noise to the inputs. The field of adversarial attacks applied to IDSs has experienced an increase in the number of works exploring their applicability. The following subsections provide an overview of the recent advancements and contributions in this field.

BB Attacks and Substitute Models: Guo et al. [9] proposed a BB attack method against ML-based anomaly network flow detection models. The method consists of training another model to substitute for the target ML model. Based on the overall understanding of the substitute model and the migration of the AE, they used the substitute model to craft adversarial examples. The authors conducted experiments on typical classification models, including Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), k-nearest Neighbor (kNN), Multilayer Perception (MLP), and Residual Networks (ResNet), which were all trained and evaluated using the KDD99 and the IDS2018 datasets. A key success factor in their work was mirroring the target model and the substitute model. However, the proposed work didn't consider adding constraints to craft AEs. The authors limited the number of modifiable features and concentrated on introducing perturbations to specific features of network flow data.

GAN-Based Approaches: Abdelaty et al. [39] presented the GADoT framework, which is a novel GAN-based approach for adversarial training. GADoT was proposed to leverage and augment the training set with adversarial traffic samples and increase the robustness of ML models against DDoS attacks. The approach generates fake-benign samples and adversarial training data that can be fed into LUCID (i.e., a lightweight DL solution for DDoS attack detection). GADoT successfully reduced the percentage of undetected malicious flows to 1.5% compared to 60% without it. The approach focused solely on adversarial training to mitigate the impact of adversarial attacks. However, the evaluation of adversarial training did not utilize constrained adversarial network flows for retraining the GAN model, which limits the applicability of the approach to real-world IDSs. In [40] a related IDSGAN approach was proposed. IDSGAN consists of a generator transforming the original malicious traffic records into adversarial versions and a discriminator dynamically learning the real-time black-box detection system by classifying traffic examples. The framework incorporates a restricted modification mechanism to ensure the adversarial generation preserves the original functionalities of the traffic records, but traffic constraints were not explicitly applied to perturbed traffic.

Adversarial Domain Constraints: Several studies examined domain constraints to generate valid adversarial network data. The authors in [15] used evolutionary algorithms and generative adversarial networks to create AEs that bypass ML-based NIDSs. The authors introduced the concept of mutability within domain constraints, where certain network traffic features can be perturbed while others are immutable. Mutable feature perturbations were limited to incrementing feature values to generate valid AEs. Our method proposes a broader set of constraints allowing various modifications while adhering to the semantics and statistical relationships between the features of network data.

The authors in [41] employed the Carlini and Wagner (C&W) method to create an adversarial attack known as the Restricted Traffic Distribution Attack (RTDA). The attack was used to target ML models used for classifying network traffic. The approach relied on two types of constraints: increasing the values of packet sizes and ensuring that the distribution of the manipulated packet size maintained a monotonically non-decreasing property. The authors, however, solely focused on maintaining their constraints on packet sizes without considering other features of network traffic.

In [42], the authors conducted an evaluation of an anomaly-based NIDS in terms of its effectiveness against AEs created through legitimate traffic transformations. The proposed constraints pertained to how adversaries interact with the victim system rather than perturbing a traffic to deceive DL models.

They focused on dividing the payload across multiple packets, changing IP addresses and the timing between packets while keeping their content intact. They also created fake packets with predefined attributes and subsequently transmitted such packets alongside the attack traffic.

Evaluating DL/ML Models' Resilience to Adversarial Attacks: The authors in [43] conducted a comparative study on various adversarial machine learning techniques applied to NIDSs, which included Fast Gradient Sign Method (FGSM), Basic Iterative Method (BIM), DeepFool, C&W, EAD (Elastic-Net Attack with Directional Gradients), JSMA (Jacobian-based Saliency Map Attack), and CW-L2 (C&W with L2 Norm) on two public datasets, NSL-KDD and CICIDS2017. FGSM, BIM, and DeepFool were the most effective methods. Following these top-performing methods, the study identified additional effective techniques, namely C&W, EAD, and JSMA, and C&W-L2. The approach involved generating adversarial examples (AEs) without enforcing network domain constraints. In a similar vein, Zhang et al. [44] presented a framework called TIKI-TAKA to generate and detect adversarial attacks on well-known IDSs datasets in one-to-one and one-to-all and classification tasks. Their framework targeted attacks that occur due to time-based features perturbations. They proposed three defense mechanisms: ensemble voting, adversarial training through ensemble techniques, and query detection. The authors assumed that many features should be left unchanged when creating AEs as they directly affect the behavior of traffic flow. However, leaving too many features unchanged limits the scope for adversarial attacks.

Analysis of Transferability and Adversarial Training: The authors in [16] investigated how AEs transfer between different ML models, including transferability from feed-forward neural networks to logistic regression and random forests models. Key findings indicated that adversarial attacks were less successful when applied to models different from the target model. The study also found that transferability of AEs tended to decrease when ML models were trained on datasets with imbalanced class distributions. Our research extends this category of studies by examining the transferability of constrained AEs and the effectiveness of adversarial training across various ML and DL models, including both similar and different attack signatures.

The authors in [8] focused on whether network domain constraints can enhance defense against adversarial attacks. They modified two adversarial algorithms: Adaptive Jacobian-based Saliency Map (AJSMA) and Histogram Sketch Generation (HSG) and integrated constraints in creating valid AEs. Additionally, they conducted an analysis of transferability, utilizing two metrics: inter-transferability (transferability between different models) and intra-transferability (transferability within the same model). We discussed our comparison with this research in the experiments section.

Problem Space Attacks: Recent research has placed emphasis on adversarial ML attacks in the problem space. These attacks deceive ML models by modifying the input data in a way that is imperceptible to humans. The authors in [45] argued that problem-space attacks pose greater challenges to the defense mechanism compared to traditional feature-space attacks. Our constraints partially addressed the problem space of network data to generate valid AEs.

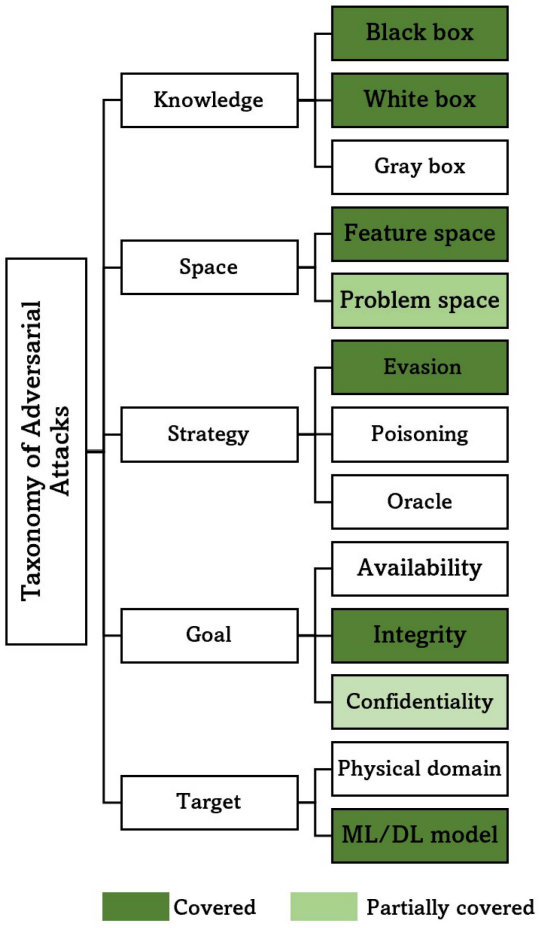


Fig. 1. A taxonomy of adversarial attacks and dimensions covered in our threat models.

Robust and Explainable Attacks: Recent research has focused on evaluating the robustness of WB adversarial-trained models using explainable AI and the SHAP (SHapley Additive exPlanations) method [46]. By leveraging SHAP method, the authors were able to measure the contribution of different features for a deeper understanding of the model's behavior and identify vulnerabilities or areas where the NIDSs is more susceptible to adversarial attacks.

III. Assumptions

Our methodology were developed based on the taxonomy presented in prior research within the area of adversarial attacks as summarized in Figure 1. We adapted the threat models within the context of adversarial attacks on NIDSs based on the 1) attacker's knowledge, 2) attack space, 3) attacker's strategy, 4) attacker's goal, and 5) attack target.

Attacker knowledge describes the amount of knowledge the attacker know about the targeting ML/DL model. Based on that, adversarial attacks are classified into WB, BB and grey box. Under WB attacks setting, we assumed that the attacker has full access to the internal details of the targeted model. This includes knowledge of the model architecture, parameters, and potentially the training data. In BB attacks

setting, we assumed that the attacker has no access to the internal details of the targeted model, such as its architecture, parameters, or training data. The attacker can only observe the input-output behavior of the model and has limited knowledge about specific features or patterns that influence its classification process. Gray-box attacks assume that the attackers have partial knowledge of the targeted ML/DL models. Under those attacks, the attackers may have a partial knowledge about the model either by querying it or by having some level of access to its training data.

Attack space refers to possible perturbations applied to the data used in the training and evaluation of ML/DL models. In the domain of network traffic, perturbations can be applied at two levels: the feature space and the raw network data space. The feature space represents a high-level aggregation of network traffic, which includes various features extracted from network packets such as protocol types, services, port numbers, and other features that characterize the traffic. The raw network data space involves the unprocessed network traffic data, typically captured in pcap files containing the actual network packets that were transmitted over the network. As a result, attackers have two principal approaches for launching their attacks. They can operate within the feature space, where they manipulate existing feature values to deceive ML/DL NIDSs models. Alternatively, attackers may opt to operate within the problem space, generating entirely new instances of network traffic that the ML/DL model has never encountered. While the primary focus of this paper is feature space attacks, the constraints we developed maintain the semantic and statistical characteristics of network data. Therefore, our constraints are applicable to problem space attacks as well.

Attacker's strategy pertains to the stage within the ML process when the attacker carries out their attacks, which can be categorized into evasion, poisoning, or oracle attacks. Evasion attacks are executed during the inference or testing phase, after the model has been trained. Poisoning attacks take place during the model's training phase, with the intent of corrupting the model learning process. Oracle attacks involve creating substitute models that mimic the behavior and functionality of the target model to induce similar responses. Our attacks were only conducted at testing time; therefore, we applied the evasion strategy.

Attacker's goal refers to the specific objectives that she/he seeks to accomplish. These objectives can be broadly categorized into three main areas: integrity, confidentiality, and availability. If the attacker's intention is to tamper with or corrupt the training data, this falls under the category of integrity-related goals. If the attacker is looking to gain access to the characteristics of the ML/DL Model or the network data it handles, this aligns with confidentiality-related goals. On the other hand, when the attacker aims to disrupt the normal functioning of ML-based NIDSs Systems, their goal is to compromise the model's availability. In our study, we conducted both targeted and untargeted attacks with the primary aim of compromising the model's integrity and, to some extent, its confidentiality.

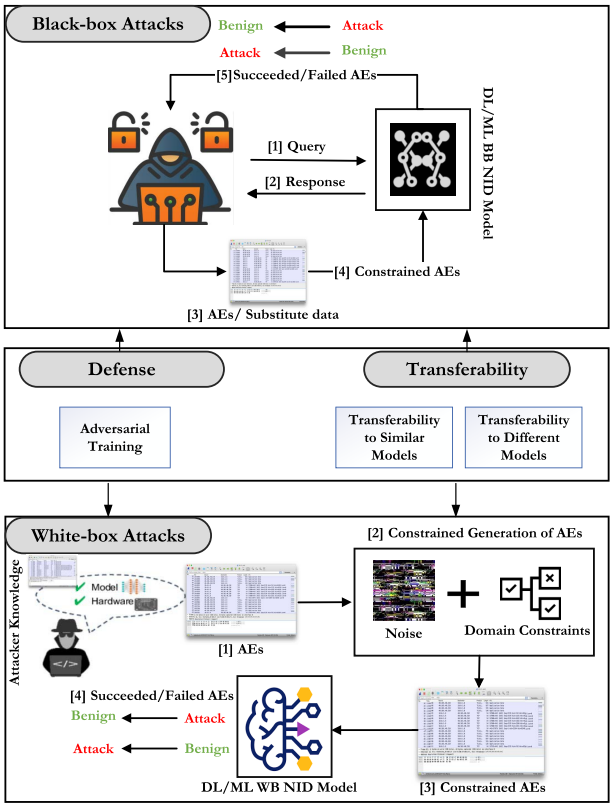


Fig. 2. A framework for generating valid network adversarial traffic.

Attacker's target can be the physical domains such as the cyber-physical systems or the applications of ML/DL. In this research our target is to undermine the effectiveness of DL and ML models' in accurately classifying network traffic as either an attack or a benign activity.

IV. Methodology

The paper employs the methodological framework summarized in Figure 2 to generate AEs under targeted and untargeted attack modes. Under the targeted mode, the goal of the attacker is to misclassify an AEs as a benign activity. In the untargeted mode, the attacker's goal is to cause misclassification of attacks and benign activities without specifying a particular target class. The approach allows for the evaluation of the techniques' evasiveness on both attacks and benign activities before and after applying the developed network domain constraints.

Threat Models: Our constraints work for both WB and BB attacks. We assumed that BB attacks mimic real-world scenarios where attackers possess limited knowledge of the internal workings and architecture of the targeted ML/DL NIDS. In addition, we aimed to assess the vulnerability of ML/DL NIDSs under WB attacks, where the attacker has full knowledge about the targeted models.

We considered Zeroth Order Optimization BB attack (ZOO) [37], DeepFool WB attack [47], and Carlini & Wagner WB Attack (C&W) [48]. Those attacks were commonly adopted as benchmark attacks within the domain of adversarial

ML. Moreover, prior research extensively validates their efficacy [49], [50], [51], [52]. Therefore, they represent a suitable choice for evaluating the resilience of NIDSs under various attacking strategies. Under BB attacks, the attack on a DL/ML model involves a series of steps as shown in Fig. 2. The attacker begins by querying the targeted BB NID model with an input network example such as a netFlow x to obtain a corresponding prediction $y = f(x)$. Input examples and query results are used to construct a substitute dataset D', consisting of input-output pairs (x', y'). Those pairs represent the initial AEs without any constraints. Given our interest in binary classification of network traffic as attacks or benign activities, our primary objective is to develop a BB hard-label attack. This attack involves iterative updates of input vectors using gradient estimates obtained from the targeted trained DL/ML NID model [53]. A hard-label attack directly assigns a discrete label to each input instance without providing probabilities or confidence scores. The attacker then generates AEs $\hat{x}_p$ while satisfying the set of network domain constraints *C*. These constraints encompass restrictions on: 1) the number of modified features, 2) the magnitude of perturbation, 3) statistical and semantics restrictions, and 4) feature range restrictions.

The approach to create AEs for WB attacks is relatively different since the attacker is assumed to have knowledge about the target model and data. The goal is to perturb the input network data by adding noise while considering domain constraints to ensure the validity of the perturbed examples. Both categories of attacks were tested under targeted and untargeted attack modes. The created adversarial examples were tested under many experimental scenarios, which include transferability of those examples DL/ML models similar to initially targeted model and to models with different or more complex architectures. The next subsection discusses the role of constraints to create valid network traffic and outlines our formal model to define constraints.

A. Defining Network Constraints in NIDSs

NIDSs rely on network traffic features to establish a baseline profile of typical network behavior. The created profiles detect deviations in the incoming traffic to identify attacks [54]. However, the incoming traffic needs to maintain network protocols and standards. Network traffic that fails to conform to the established protocols can be identified as an anomaly, suggesting it may be invalid or suspicious. Such a traffic could be the result of an attacker's deliberate perturbation, leading to the creation of AEs. Therefore, adversarial attacks against security systems such as NIDSs may also lead to a non-functional network traffic. A common example of generating invalid AEs is through perturbing the "Source to Destination Time to Live" (*STTL*) feature. STTL is used to ensure that packets have a large enough TTL value to reach their intended destination even if there are many routers in the path. In some cases, anomalies or deviations in the STTL values can be indicative of potential network attacks or abnormal network behavior. If the value of STTL is manipulated to be 257, the perturbation applied is considered invalid because the maximum value allowed for the STTL field in an IP packet is 255. Another example can be an invalid tcp roundtrip time *tcprrt*, which potentially affects the sum of the *synack* and *ackdat* of the TCP packet. This can happen if an attacker modifies the values of the *synack* and *ackdat* features in a way that can result in a different checksum calculation compared to the original values. By bypassing the network functionality of the checksum, attackers aim to create AEs passing the integrity checks performed by the network devices and NIDSs. However, they may still need to consider network constraints into their account when crafting AEs to evade threat detection algorithms.

TABLE II
TABLE OF NOTATIONS

Variable	Definition
x, $\hat{x}_p$	Clean example, adversarial example
y	Label of a network flow $\in$ {benign, attack}
y^t	Targeted label
$TCA(x)$	Traffic classification algorithm
I, MU	Immutable Feature, Mutable feature
B_j, R_j and C_j	Binary, continuous and categorical features
$R(f,k)$	Linear relationship between features f and k
μ, σ	Mean and standard deviation
D'	Substitute dataset
WB, BB	White box and black box
B	Perturbation budget
f	Feature before perturbation
$\hat{f}_p$	Feature after perturbation
s_s	Step size
m_i	Maximum iteration
c_f	Attack confidence
VAR	Valid adversarial example

Table II demonstrates the formal notations used to create, apply, and experiment our constraints. Formally, given a benign sample $x \in X$, with the corresponding correct label $y \in Y$, where *y* belongs to either an attack or a benign activity, the attacker goal is to craft an AE $\hat{x}_p = x + \eta$ such that $TCA(\hat{x}_p) \neq y$ in an untargeted attack, $TCA(\hat{x}_p) = y^t$ in a targeted attack, η denotes the applied perturbation, and *TCA* is a traffic classification algorithm.

In this paper, y^t represents the target chosen by the attacker, with the intention of classifying AEs as benign network flows. It is important to note that our constraints are not mutually exclusive, meaning that multiple types of constraints can be applied to the same feature. For each of the attack modes, our constraints belong to one or more of following categories:

- *Features Mutability Constraints:* Network traffic includes mutable and immutable features. Mutable features are those attributes of network traffic that can be changed, while immutable features are network traffic attributes that cannot be changed. Examples of mutable features are *srcbytes*, *dstbytes*, *duration* and *port numbers*. The examples of immutable features are *protocol*, which is the protocol used to transmit data, *state*, which denotes the state of the network connection, whether it is established, closed, or in the listening state, and *service*, which is the type of service or application associated with the network communication, such as HTTP or FTP.
  Formally, let an immutable feature denoted by $I \in \{protocol, state, service\}$ and a mutable feature denoted by $MU \subseteq \{1, 2, \ldots, r\}$. The attacker goal is to craft AEs

that satisfy the following constraint:

$$\forall i \in I, \hat{f}_p = f \tag{1}$$

$$\forall m_u \in MU, \hat{f}_p = f + \eta \tag{2}$$

where the perturbation η is applied to the original mutable feature to create an *AE* $\hat{x}_p$

- *Feature value constraints:* Network traffic features can adopt different forms, such as binary, discrete, continuous, and categorical. Binary traffic features are those that can only have two possible values 0 or 1. The examples of binary features in network traffic are *IP options*, *Fragmentation features*, and *Error features*. Continuous network features can take a value within a numerical range. The examples of continuous features in network traffic include *ackdat*, which is the time duration between the sending of a SYN packet and the receiving of an ACK packet in a TCP connection, *sintpkt*, which is the inter-packet arrival time between consecutive packets sent from the source to the destination, and *sload*, which is the average traffic load (in bits per second) from the source to the destination. Categorical network features are those that can take discrete values without any inherent order or numerical relationship between the categories. The examples of categorical features in a network traffic include: the protocol field in the IP header, which take the values such as *TCP*, *UDP*, or *ICMP*, service field, which can take values such as *HTTP*, *FTP*, and *DNS*, state field which can values such as *FIN*, *INT*, *CON*, *URH* or others, and Flag feature which indicates various control flags in the TCP header, such as *SYN* (synchronize), *ACK* (acknowledge), *RST*, and (reset).
  Let the set of binary features in the network traffic is denoted by $B_j \subseteq \{1, 2, \ldots, n\}$, the set of continuous features is denoted by $R_j \subseteq \{1, 2, \ldots, m\}$, and the set of categorical features is denoted by $C_j \subseteq \{1, 2, \ldots, t\}$. The attacker goal is to craft *AEs* that satisfy the following constraints:

$$\forall b_j \in B_j,\ \hat{f}_p \in [0, 1] \tag{3}$$

$$\forall j \in R_j,\ \min(f) \leq \hat{f}_p \leq \max(f) \tag{4}$$

$$\forall c_j \in C_j,\ \hat{f}_p \equiv \sum_{j=1}^{t} C_j = 1 \tag{5}$$

  where *t* in formula 5 represents the total number of values for a categorical feature c_j. Since categorical features are mapped using one hot encoding, the last constraint holds. In one-hot encoding, each category is represented by a binary feature that takes the value of 1 if the category is present and 0 otherwise.
- *Feature dependency constraints:* Some features of network traffic are dependent on other features. For instance, the source IP address and source port of a packet are often dependent on each other because they are associated with the device sending the packet. Similarly, the destination IP address and destination port number have a dependency relation because they are both associated with the device receiving the packet. Another example is the flow average inter-arrival time which is the average time between packets in a flow. It is calculated by dividing the total time of the flow (*duration*) by the number of packets in the flow. If *R* denotes a relationship between the features f and k in the original flow, then the introduced constraint must preserve the relationship between the features of the perturbed flows $\hat{f}_p$ and $\hat{k}_p$ as follows:

$$\forall R(f, k) \exists R\left(\hat{f}_p, \hat{k}_p\right) | R\left(\hat{f}_p, \hat{k}_p\right) = R(f, k) \tag{6}$$

  Dependency relationships can be also regression-based, example of those relationships is the linear regression. In BB models those relationships can be approximated from the substitute dataset D'. Suppose that there is a linear relationship between features z and h. Let us assume that D' contains (m) number of features. Let us also assume that z can be predicted based on h. We are interested in preserving the relationship between z and h in the corresponding linear regression model:

$$z = h\beta + \varepsilon \tag{7}$$

  Suppose that $n \times m$ data matrix contains observations of both features, β is the coefficient of the model that we are trying to find, and ε is the error. If we use ordinary least square method for estimation, then, $\hat{\beta} = (h^T h)^{-1} h^T z$. Here $h^T h$ is the covariance matrix of the data if the mean of each feature is set to zero. $h^T z$ is the covariance between every feature vector of h and z. Our constraints ensure the preservation of the linear regression relationship between those features.
- *Distribution-preserving constraints:* The statistical distribution of feature values in the original network data should be taken into account by attackers when crafting *AEs*. Specifically, an attacker should try to craft AEs that follow the same statistical distribution of the benign examples to make them look legitimate if his target is classifying them as benign. For instance, if a NIDS is trained on a dataset where the majority of packets have a small size, an attacker may try to craft AEs with small size too. Hence, the adversary should follow the original statistical distribution of data to avoid outliers AEs through considering the following constraints:

$$\forall \hat{f}_p | \hat{f}_p = f \pm \epsilon \tag{8}$$

$$\mu_{\hat{f}_p} \approx \mu_f \tag{9}$$

$$\sigma_{\hat{f}_p} \approx \sigma_f \tag{10}$$

  Due to the absence of access to the original data in BB attacks, we utilize substitute data D' to uphold this constraint. In order to perturb numerical features, we employ a noise parameter (e.g., ϵ) as indicated in formula 8. This perturbation process ensures that the mean and standard deviation of the perturbed feature closely match those of the original feature.

Algorithm 1 Constrained Generation of Adversarial Examples

Require: Dataset $\mathcal{D}$ or $\mathcal{D}' = (x_p, y_p)_{p=1}^{n}$, Adversarial perturbation budget $B = \{\epsilon,\ c_f,\ m_i,\ s_s\}$, Attack set $\mathcal{A}$, Constraints set $\mathcal{C}$, Feature set $\mathcal{M}$
Ensure: Adversarial examples $\tilde{x}_p$ for $p = 1, ..., n$
1: **for** $p = 1$ to n **do**
2: $x_p \leftarrow$ Preprocess(x_i) {Preprocess input flow}
3: $y_p \leftarrow$ One-Hot(y_p) {Encode true label as one-hot vector}
4: $\tilde{x}_{p,0} \leftarrow x_p$ {Initialize adversarial example as original input}
5: **for all** $a \in \mathcal{A}$ **do**
6: $\tilde{x}_{p,a} \leftarrow$ ApplyAttack$(a, \tilde{x}_{p,a-1}, y'_p, B)$ {Apply adversarial attack}
7: **for all** $c \in \mathcal{C}$ **do**
8: $\tilde{x}_p \leftarrow$ Project$(\tilde{x}_p, c)$ {Projection of constraints set on $\mathcal{M}$}
9: **end for**
10: **end for**
11: **end for**
12: **return** $AE = \{\tilde{x}_{p,a}, y_p^{(t)})\}_{p=1}^{n}, a \in \mathcal{A}$

B. Generating Constrained AEs for NIDSs

Our AEs generation procedure is depicted in Algorithm 1. The algorithm requires a set of constraints $\mathcal{C}$, which encompasses a broad spectrum of constraints discussed earlier. The parameters of perturbation budget B are selected depending on the type of the attack used to generate *AEs*. It can be a WB or a BB attack. B includes parameters such as attack confidence c_f, maximum iterations m_i, step size s_s or the noise parameter ϵ. Preprocessing of flows includes removal of any missing feature values and label binarization. The algorithm returns, for each type of attacks, the set of $AEs = \{\tilde{x}_{p,a}, y_p^{(t,u)})\}_{p=1}^{n}, a \in \mathcal{A}$, and the targeted labels of AEs $y_p^{(t)}$. The attacks used in the generation of AEs are WB or BB. They can be targeted or untargeted. The following subsections discuss each of those attacks in detail.

In our study, we employed three adversarial attack generation algorithms presented in the subsequent subsections:

1) Zeroth Order Optimization BB Attack (ZOO): ZOO attack was proposed by the authors in [37]. It operates under the constraint that the attacker lacks direct access to the gradients of the target model and can only observe model's output for a given input. Unlike gradient-based adversarial attacks, which utilize model gradients to create AEs, ZOO generates such examples through optimization of the following regularization function:

$$\min_{\hat{x}_p} \|\hat{x}_p - x\|_p + c \cdot g(\hat{x}_p) \tag{11}$$

Such that $\|.\|_p$ is the regularization used to enforce the similarity between AEs and the benign examples in terms of the Euclidean distance (if the attack is targeted), p is the norm, and c is regularization parameter. $g(\hat{x}_p)$ is the BB hinge loss function which depends on model output as follow:

$$g(\hat{x}_p) = \max\big(\max i \neq t \log[f(x)_i] - \log[f(x)_t], -c_f\big) \tag{12}$$

where $f(x)$ represents the logarithm of the probability of the model's output for class i given the input flow x, t is the label of the target class (i.e., the benign label), and c_f is used to control the attack confidence. For estimating the gradient information, symmetric difference quotient is used to optimize the model and approximate gradients in BB setting without any knowledge about model architecture and parameters as follow:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + s_{s\,i}) - f(x - s_{s\,i})}{2s_s} \tag{13}$$

The partial derivative above approximates $f(x)$ with respect to x_i by measuring the difference in the model's output for perturbed inputs $x + s_{s\,i}$ and $x - s_{s\,i}$. The step size s_s is a small constant used in the approximation.

2) DeepFool WB Attack: DeepFool WB adversarial attack is based on the idea of finding the minimum perturbation to an input that will cause the model to misclassify it [47]. The attack works with an initial input, then it iteratively computes the gradient of the model's output with respect to the input. At each iteration, the attack adds a small perturbation to the input in the direction of the gradient that will increase the model's output for the least likely class. The process is repeated until the model's output for the least likely class exceeds that of the true class. The process of generating AEs is formalized as follows:

$$\hat{x}_p \leftarrow x; \tag{14}$$
$$\bar{y} \leftarrow 1 \quad \text{to} \quad T; \tag{15}$$
$$cf\left(\hat{x}_{p,\bar{y}}\right) \leftarrow cf\left(\hat{x}_p\right); \tag{16}$$
$$w_{\bar{y}} \leftarrow \nabla cf_y\left(\hat{x}_{p,\bar{y}}\right) - \nabla cf_{\bar{y}}\left(\hat{x}_{p,\bar{y}}\right); \tag{17}$$
$$\Delta x_{\bar{y}} \leftarrow \epsilon \frac{w_{\bar{y}}}{|w_{\bar{y}}|_2}; \hat{x}_{p,\bar{y}+1} \leftarrow \left(\hat{x}_{p,\bar{y}} + \Delta x_{\bar{y}}\right). \tag{18}$$

DeepFool takes as input the traffic flow x to be perturbed, a classifier cf and a perturbation parameter ϵ. The attack will run for T number of iterations. It starts by initializing the adversarial example $\hat{x}_p$ to be the same as the original example. A minimum perturbation is required to move the input across the decision boundary of the neural network. At each iteration, the gradients output of the neural network with respect to the original input example are calculated, $w_{\bar{y}}$ is the direction of the decision boundary between the true class y and the $\bar{y}_{th}$ class. It is obtained by linearizing the decision boundary. The direction of the minimum perturbation to change the classification of the input example is then computed. DeepFool then checks whether the input example has been successfully classified as benign if the attack is targeted. The process is continued until that example is classified as benign.

3) Carlini & Wagner WB Attack: The Carlini and Wagner (C&W) attack is a WB adversarial attack [48]. The generation of adversarial examples is formulated as the minimization of the distance between the adversarial example and the original input, while also maximizing the classifier's confidence in the target class. C&W attack finds an initial AE, which is an input that is close to the original example. It continues refining it until it generates an AE based on the following optimization functions:

$$\min_{\hat{x}_p} \|\hat{x}_p - x\|_2 + c \cdot f(x, t) \tag{19}$$
$$f(x, t) = max_{i \neq t}[Z(x)]_i - [Z(x)]_t \tag{20}$$

TABLE III
DATASETS

Criteria	UNSW-NB15	IoT-23
Attack sample	17814	5537
Benign sample	542133	4462
Number of flows	559947	9999
Number of features	49	21

where $\hat{x}_p$ is the perturbed input, x is the original input or the clean sample to be perturbed. $\|\cdot\|_2$ denotes the Euclidean norm or the L2 norm to measures the distance between $\hat{x}_p$ and x, c is a scalar constant that controls the trade-off between the perturbation size and the confidence of the attack, $f(x, t)$, is the objective function which includes both the distance ($\|\hat{x}_p - x\|_2$) and confidence ($c \cdot f(x, t)$), and $Z(x)_i$ is the confidence score of the logit layer. $Z(x)_i$ is computed using the SoftMax classification rule to predict the class label of an input. The AEs are passed through the logit layer and the SoftMax function to maximize the difference between the true class and the target class.

V. Experimental Results

A. Data Preprocessing

We employed the UNSW-NB15 and IoT-23 datasets to assess the effectiveness of our methodology. Both datasets comprise network flows specifically designed to evaluate NIDSs based on DL/ML methods. Table III demonstrates the details of both datasets.

UNSW-NB15 dataset [55] is a public dataset that was collected at the Cyber Range Lab of the Australian Centre for Cyber Security using IXIA PerfectStorm devices. This dataset includes both legitimate and malicious network traffic represented in both packet and flow-based formats. The attacks in this dataset are classified into backdoors, DoS, Reconnaissance activities, Worms, and some other generic attacks. From this dataset, we generated a balanced sample containing 17,814 benign flows and 17,814 attack flows.

The IoT-23 dataset contains IoT network flows that were captured by the Avast AIC lab between 2018 and 2019 [56]. IoT-23 includes both malicious and benign network traffic from 20 infected IoT devices and 3 non-infected devices. The dataset was created to examine how benign IoT devices behave within computer networks. IoT-23 includes the original network capture (i.e., pcap file) and the log files. The log files have been analyzed and labeled manually. Flows are labeled as benign or attacks under different categories including C&C, DDoS, Filedownload, Heartbeat, Mirai, Okiru, PartOfHorizontalPortScan, or Torii.

We applied several data preprocessing and cleaning steps. We removed records with missing values from both datasets. We then binarized the class labels by converting the target classes into a binary format representing attacks or benign activities. To prepare the data for ML and DL algorithms, we converted categorical features into numerical values using the One-Hot Encoding technique [57]. The most influential features were selected using Information Gain leading to excluding some features such as timestamps, locally originated connections, and locally originated responses [58]. We also utilized the Min-Max approach to normalize the data before the training step to ensure that all features are on the same scale and to prevent some features from dominating the learning process based on their magnitudes. We addressed the class imbalance in the UNSW-NB15 dataset since imbalanced datasets can result in a model that exhibits a bias towards the majority class, making the models more vulnerable to adversarial attacks [16]. We utilized random sampling without replacement to generate representative and unbiased samples from both classes such that every instance has an equal chance of being selected [59].

B. Training of Adversarial Attacks

For the ZOO attack, we employed the Decision Tree (DT) model in the training and testing phases. The DT was configured with the Gini impurity criterion and employed a best split strategy with no imposed restrictions on the tree's maximum depth, allowing it to expand until reaching a minimum split of 2 in our configuration. Additionally, we constrained the maximum number of nodes to 50 as a measure to control the tree's growth. DeepFool and C&W were trained on a DNN model with 3 hidden layers, each comprising 256, 256, and 128 neurons. The model employed Rectified Linear Unit (ReLU) activation function and utilized Adam optimizer. We used Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and a learning-rate of 0.01 to control the parameter updates during each DNN training iteration. In our experiments on DeepFool and C&W attacks, we systematically explored a range of hyperparameter values to identify the optimal configuration for our datasets and threat model. We adopted this parameter tuning approach using the methodology presented in [48]. Our main objective was to preserve the balance between ASR and attack stealthiness. After a comprehensive evaluation, we have determined that the following hyperparameter settings yielded the best results: for the DeepFool attack: $m_i = 100$, $\epsilon = 1e{-}6$, and the number of class gradients $= 10$. For the C&W attack: $c_f = 0.0$, learning-rate $= 1e - 2$, and the binary search step $= 10$. All experiments run on a NVIDIA A100 GPU with 80 GBs of VRAM. Details about our experimental setup and the source code are available in the source code repository.[1]

C. Evaluation Metrics

In our experimental analysis, we employed various evaluation metrics as outlined below.

1) Attack success rate (*ASR*): For untargeted attacks the *ASR* is calculated as: $ASR_{\text{untargeted}} = \frac{S}{N} \times 100$ where *S* represents the number of successful *AEs* (i.e., correctly misclassified), and *N* represents the total number of the tested *AEs*. Attack success rate for targeted attacks is calculated as: $ASR_{\text{targeted}} = \frac{S_{\text{targeted}}}{N} \times 100$ where ASR_{targeted} represents the *ASR* for targeted attacks, S_{targeted} represents the number of successful targeted *AEs* that deceived the model, and *N* represents the total number of attempted targeted *AEs*.

[1] https://github.com/Nour-Alhussien/ConstrainedAdversarialAttacks

2) The percentage of valid *AEs* based on the measure (*VAR*): *VAR* provides insight on the effectiveness of the constrained attacks. It measures the proportion of adversarial examples that mimic a valid network traffic. VAR is calculated as: $VAR = \frac{V}{T} \times 100$ where V represents the number of valid *AEs* and T represents the total number of *AEs*.
3) Accuracy (*Acc*): Model accuracy is calculated as: $(Acc) = \frac{CP}{N} \times 100$ where *Acc* represents the effectiveness of model in terms of classifying attacks as attacks and benign activities as benign activities, and *CP* represents the total number of correctly classified flows at the testing time and N is the total number of tested flows. Since we are interested in testing the model's resistance to intentional manipulation and analyzing the model's vulnerability to adversarial attacks, we solely focus on testing the model's performance against these crafted inputs.

The measures above were evaluated under both targeted and untargeted attacking modes. Targeted attacks involve training the model on both benign and malicious activities but generating *AEs* specifically crafted to misclassify attacks as benign activities. On the other hand, untargeted attacks involve the same training process, but the attacker targets the testing phase using *AEs* generated for both the benign and malicious activities. During the testing phase, untargeted and targeted attacks were evaluated on the UNSW-NB15 dataset using two sets of *AEs*. Specifically, 8916 adversarial network flows were used for testing untargeted attacks, while 4502 adversarial network flows were employed for testing targeted attacks. For the IoT-23 dataset, 2000 and 1121 adversarial network flows were generated to test untargeted and targeted attacks respectively.

D. Results and Discussions

This section discusses the results of 1) effectiveness of adversarial attacks under constraints, 2) transferability of attacks, and 3) attack defense using adversarial training.

1) Effectiveness of Attacks:

Results of ZOO BB Attack: To examine the impact of parameter settings on the *ASR* of the ZOO adversarial attack, we tuned two parameters m_i and c_f, then analyzed their impacts on both *Acc* and *ASR*. Tables IV and V show the results of this experiment before and after applying network constraints on the two datasets. Our observations on the UNSW-NB15 dataset indicate that increasing m_i and c_f results in a higher *ASR*. Specifically, varying the m_i between 20 to 80 and the c_f between 0.0 to 0.5 increased the *ASR* from 0.061% to 20.4% under untargeted attack mode. However, increasing m_i and c_f in the targeted attack resulted in a small increase in *ASR* from 0.039 to 0.088. It is observed that applying constraints do not lead to significant changes in *ASR* and *Acc* as noticed in Table IV. We observed a small impact on *ASR* when applying such constraints to targeted attacks. In untargeted attacks, the objective is to find any perturbation that leads to misclassification regardless of the target class. As a result, the attacker has a broader range of perturbation magnitudes that meet the constraints and lead to misclassification across the two classes.

The results for employing ZOO attack on the IoT-23 dataset which has a smaller number of features is shown in Table V. Changing the m_i and *cf* does not significantly affect the values of *ASR* and *Acc*. However, the performance of ZOO attack is affected by varying the values of the step size s_s. Increasing the s_s from 0.2 to 0.8 resulted in a significant increase in the *ASR* from 0.045 to 27.84% under untargeted modes and before applying any constraints. This substantial increase can be attributed to the s_s numerical estimation of the derivatives. The larger s_s enables the algorithm to navigate and explore more perturbation magnitude.

Applying network constraints to the IoT-23 dataset does not have a significant impact on *ASR*. It is also noticed that the *ASR* for the targeted/untargeted ZOO attacks was higher on IoT-23 compared to UNSW-NB15. This is because in the case of the IoT-23 dataset, the model was trained on smaller number of attack signatures compared to the UNSW-NB15 dataset.

Results of DeepFool and C&W WB Attacks: The results of our experiments on DeepFool and C&W attacks are shown in Table VI. Before conducting both attacks, we assessed the accuracy of the DNN models, which resulted in a high *Acc* value of 95%. However, after executing both attacks, the accuracy has significantly decreased before applying any constraints. It is noticed that applying constraints leads to a significant impact on the *ASR* of WB both attacks. Results in Table VI demonstrate the following observations under different experimental settings:

- There is no significant difference observed between targeted and untargeted attack modes on the UNSW dataset when utilizing the DeepFool attack. Applying constraints leads to very low *ASR* of $x \leq 0.001\%$ (Table VI). It is observed that the introduced constraints interfere with the effectiveness of the adversarial perturbations. The constraints may alter the behavior of the model and the introduced BB perturbations in a way that weakens their ability to deceive it, leading to a significantly lower *ASR*. DeepFool attacks approximate the model's decision boundary linearly, therefore, when network constraints were applied after generating DeepFool AEs, the linear approximation between the features of the UNSW dataset is disrupted, resulting in a significant drop in *ASR*. The decline in the *ASR* when experimenting on the IoT-23 dataset, which has less number of features was less significant under both targeted and untargeted attack modes. The *ASR* dropped from 95.15% to 34.36%, and from 96.38% to 39.20% for untargeted and targeted modes respectively.
- The findings for C&W attack were relatively similar to DeepFool. Under the untargeted model, the *ASR* was higher before applying our constraints and reaches 99.30%. It drops to 42.20% on the UNSW dataset after enforcing network domain constraints. *ASR* was comparable for C&W attacks under both untargeted and targeted attack modes.

TABLE IV
RESULTS FOR ZOO ATTACK ON UNSW-NB15 DATASET

Before constraints		**Untargeted attack**				**Targeted attack**			
Attack parameters	m_i	20	40	60	80	20	40	60	80
	c_f	0.0	0.2	0.4	0.5	0.0	0.2	0.4	0.5
Evaluation metrics	asr	0.061	10.7%	15.4%	20.4%	0.039	0.059	0.079	0.088
	acc	93.9%	89.3%	84.6%	79.6%	96.1%	94.1%	92.1%	91.2%
After constraints		**Untargeted attack**				**Targeted attack**			
Attack parameters	m_i	20	40	60	80	20	40	60	80
	c_f	0.0	0.2	0.4	0.5	0.0	0.2	0.4	0.5
Evaluation metrics	asr	0.06	10.7%	15.4%	20.3%	0.039	0.059	0.081	0.092
	acc	94%	89.3%	84.6%	79.7%	96.1%	94.1%	91.9%	90.8%

TABLE V
RESULTS FOR ZOO ATTACK ON IOT-23 DATASET

Before constraints		**Untargeted attack**				**Targeted attack**			
	m_i	20	40	20	40	20	40	40	80
Attack parameters	c_f	0.0	0.2	0.0	0.2	0.0	0.0	0.2	0.5
	s_s	0.2	0.2	0.8	0.8	0.2	0.8	0.6	0.8
Evaluation metrics	asr	0.045	0.051	12.7%	27.84%	0.003	0.09	21.49%	31.8%
	acc	95.5%	94.95%	87.3%	72.16%	99.73%	91%	78.51%	68.2%
After constraints		**Untargeted attack**				**Targeted attack**			
	m_i	20	40	20	40	20	40	40	80
Attack parameters	c_f	0.0	0.2	0.0	0.2	0.0	0.2	0.4	0.5
	s_s	0.2	0.2	0.8	0.8	0.2	0.8	0.6	0.8
Evaluation metrics	asr	0.043	0.05	12.8%	27.17%	0.0019	0.08	20.7%	30.36%
	acc	95.57%	94.96%	87.2%	72.83%	99.81%	92.3%	79.3%	69.46%

TABLE VI
COMPARISON OF ATTACK PERFORMANCE ON TARGETED AND UNTARGETED ATTACKS FOR DEEPFOOL AND C&W ATTACKS

Datasets	**Attack**	**Evaluation metrics**	**Untargeted attack**		**Targeted attack**	
			Before constraints	After constraints	Before constraints	After constraints
UNSW-NB15	DeepFool	asr	99.99%	0.0044	99.78%	0.001
		acc	0.0029	99.56%	0.0002	99.97%
	C&W	asr	99.30%	42.20%	99.54%	51%
		acc	0.007	57.80%	0.005	49%
IoT-23	DeepFool	asr	95.15%	34.36%	96.38%	39.20%
		acc	0.0485	65.64%	0.0362	68.14%
	C&W	asr	94.9%	43.74%	95.72%	44.18%
		acc	0.051	56.26%	0.0428	55.82%

- We observed that C&W was less impacted by constraints compared to DeepFool. Its optimization function encourages minimizing the distance between the adversarial example and the original input while maximizing the classifier's confidence in the target class. Particularly, C&W attack aims both AEs leading to misclassification and have a minimal distance between the original input and the perturbed example. DeepFool attack takes a different approach since it considers perturbation magnitude. such a difference contributes to variations in both attack responses to constraints. The C&W attack, with its explicit consideration of distances has partially handled constraints that affect perturbation direction and the confidence in that direction. Not only on the features where constraints were applied but also on other features that contribute to the model prediction.

While the existing BB attacks are usually tested on ML models, we conducted additional experiment where the ZOO attack targets a DNN model. The AEs were also tested on the same DNN model used to generate and test DeepFool and C&W attacks. We then reported the constraint impacts under both targeted and untargeted modes. The results on both datasets are shown in Figures 3 and 4. The figures illustrate

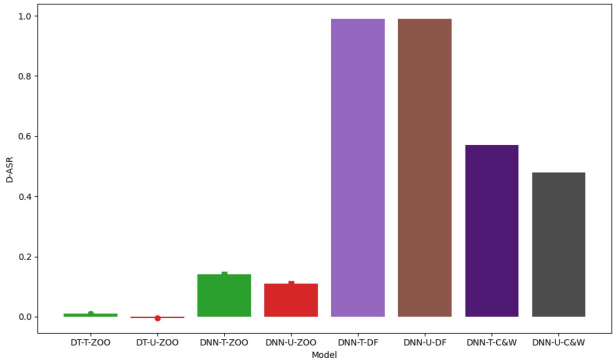


Fig. 3. Impact of constraints on different models-UNSW dataset, **DT**:Decision Tree, **T**: Targeted Attack, **U**: Untargeted Attack.

the difference in ΔASR before applying constraints compared to after applying them. The calculation of ΔASR is done as follows:

$$\Delta ASR = ASR_{beforeC} - ASR_{afterC}$$

The applied constraints had a substantial effect on DNN models when applied to WB attacks, surpassing the impact of

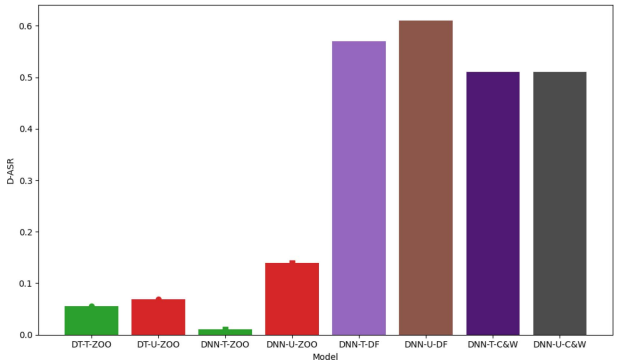


Fig. 4. Impact of constraints on different models-UNSW dataset, **DT**:Decision Tree, **T**: Targeted Attack, **U**: Untargeted Attack.

the ZOO BB attack. In addition, our observations revealed that the constrained ZOO attack had a greater impact on the DNN model compared to the ML model. Nonetheless, this impact was notably lower compared to DeepFool and C&W attacks.

Validity of AEs: We analyzed the differences between adversarial attacks in terms of generating valid *AEs* before compared to after applying constraints. The results are shown in Table VII. *VAR* was lower in the unconstrained domain for the two WB attacks compared to the BB attack. In the constrained domain, *VAR* has improved significantly to 80.88% for DeepFool and 75.85% for C&W attacks. For the ZOO attack, the *VAR* increased from 75.34% to 100%. Notably, *VAR* was higher for the ZOO compared to DeepFool and C&W attacks before enforcing network constraints. The effectiveness of the ZOO attack in generating valid *AEs*, despite being a BB attack can be attributed to its reliance on a surrogate model to generate adversarial *AEs*. This approach enables the generated *AEs* to be closer to the original valid examples, as they are based on the output of the model rather than solely the input itself. However, direct perturbation in WB attacks resulted in noticeable changes to the input, making most of the crafted *AEs* invalid. Consequently, *VAR* does not seem to have much dependency on the threat model or having access to model parameters and data. In our experiments, the BB attack yields higher *VAR* compared to WB attacks. The results were consistent across both datasets. We observed no significant differences when generating Valid *AEs* under targeted and untargeted attack modes.

It is important to note that for DeepFool and C&W attacks, the number of valid *AEs* do not reach 100% as for ZOO attacks. We believe that the existence of some outliers in the datasets leads to examples that are still not a valid network traffic. Sometimes modifying network features can also change the relationships between them, which potentially results in an invalid traffic.

2) Transferability Analysis: This series of experiments aims to evaluate the transferability of constrained and unconstrained *AEs* under both targeted and untargeted attack modes. Our main focus is to answer the following question: *To what extent do constrained/unconstrained AEs maintain their deceptive behavior when transferred between DL/ML models with similar or different structures/architectures?* Our experiments involve examining the resilience of these examples in successfully fooling models, even when they deviate from the specific model they were originally designed for. In the previous experiments, we conducted extensive analysis and evaluations on both datasets. However, since we observed similar result patterns and to maintain clarity in our discussion, we decided to limit the presentation of our findings to the results obtained from the UNSW-NB15 dataset.

Table VIII shows the architectures of the various models used in this series experiments. The table highlights the structures of the DL models used in our transferability and/or adversarial training experiments. For transferability analysis, we mainly used Deep Neural Network 1 (DNN1), Deep Neural Network 2 (DNN2), and Deep Belief Network (DBN). DNN1 and DNN2 have similar structure, since both consist of three layers with 256, 256, and 2 neurons in each layer. Both models employ *ReLU* activation function. However, DNN1 utilizes the Adam optimizer, while DNN2 employs the ASGD optimizer.

Additionally, we conducted experiments using a DBN model with a more complex structure consisting of multiple layers of restricted Boltzmann Machines (RBMs). With the Stochastic Gradient Descent (*SGD*) optimizer, the DBN optimizes the model parameters by iteratively updating them based on mini batches of training data.

Two ML models were used in our experiments, mainly the Decision Tree (DT) and Gradient Boost (GB). The DT model consists of a hierarchical structure of nodes and branches. The GB model is an ensemble ML model that combines multiple weak learners through a boosting mechanism. Since Gradient Boosting commonly employs DT as weak learners in its ensemble, both models have some similarities in the classification process.

Our experiments evaluate the transferability of *AEs* generated by WB and BB attacks between models with similar structures and different structures. The evaluation includes transferability of constrained/unconstrained *AEs* under both targeted/untargeted attacks modes as follows:

1) The transferability of *AEs* between models with similar structures:
   a) WB attacks: In this scenario, we tested the transferability of *AEs* from DNN1 to DNN2 model. Specifically, the *AEs* created using the DNN1 model are assessed against the DNN2 model.
   b) BB attacks: In this scenario, we tested the transferability of *AEs* from the DT to GB model. The *AEs* generated on the DT model were evaluated on the Gradient GB model.
2) The transferability of *AEs* between models with different structures:
   a) WB attacks: In this scenario, tested the transferability of *AEs* from DNN1 to DBN model. Particularly, the *AEs* generated by the DNN1 model were evaluated on the DBN model.
   b) BB attacks: In this scenario, we tested the transferability of *AEs* from the DT to DNN1 model. Particularly, the *AEs* generated by the DT model were evaluated on the DNN1 model.

TABLE VII
VAR RESULTS FOR BOTH DATASETS

Dataset	Attack	Untargeted		Targeted	
		Before Constraints	After Constraints	Before Constraints	After Constraints
UNSW-NB15	ZOO	75.34%	100%	78.28%	99.62%
	DeepFool	0	80.88%	0	72.68%
	C&W	0.006	75.85%	0.023	75.85%
IoT-23	ZOO	85.35%	100%	80.5%	100%
	DeepFool	0	99.35%	0	99.82%
	C&W	0	98.95%	0	98.24%

TABLE VIII
STRUCTURES OF ML/DL MODELS

Models	Name	Architecture	Optimizer
Deep learning models	Deep Neural Network 1	FNN with 3 layers: 256,256,2	Adam
	Deep Neural Network 2	FNN with 3 layers: 256,256,2	ASGD
	Deep Neural Network 3	FNN with 6 layers: 256,256,256,256,256,2	AdaMax
	Deep Belief Network	Multiple layers of stacked Restricted Boltzmann Machines	SGD
Machine learning models	Decision Tree	Hierarchical structure of nodes and branches	
	Gradient Boosting	Ensemble of decision trees with boosting mechanism	

* **FNN**: Feedforward Neural Network

Transferability Scenario 1a: We first assessed the transferability of *AEs* between DNN1 and DNN2 models. We implemented the untargeted DeepFool attack against a DNN1 model and measured its *ASR* on that model. Subsequently, we conducted a test by applying the generated DeepFool *AEs* on the DNN2 model and measured the *ASR*. Transferability results that correspond to this experiment are shown in Table IX. Most of unconstrained and untargeted DeepFool *AEs* are transferable as indicated by their ability to achieve an *ASR* of 99.80% on DNN2, which is close to the *ASR* of 99.99% achieved on DNN1. Constrained DeepFool *AEs* showed limited transferability likely due to their inability to fool the original DNN1 model.

The transferability of unconstrained DeepFool *AEs* under the targeted attack mode is lower compared to the untargeted attacks. However, the constrained DeepFool *AEs* exhibit similar behavior in both targeted and untargeted settings. It is noticed that under the targeted attack mode, the transferability of the DeepFool attack is relatively lower. The targeted attack might have been successful in exploiting the decision boundary of the original examples of the targeted class, but it might not generalize well to other models with a minor variation in the decision boundaries. The unconstrained C&W *AEs* showed similar transferability behavior under both targeted and untargeted modes. Likewise, it is observed that the transferability of the constrained C&W *AEs* is comparable to that of the unconstrained ones under both targeted and untargeted attack modes.

Transferability Scenario 1b: We implemented a ZOO attack against a DT model, we then used the resulting *AEs* against a GB model. Table XI summarizes the findings of this experiment. We observed a limited transferability of the untargeted constrained ZOO attack. The original model yielded an *ASR* of 19.5%. When these *AEs* were tested on the GB model, the *ASR* significantly drops to 0.0174. There are a few possible explanations for this. In general GB models are less sensitive to perturbations given their ensemble prediction model, which makes them more robust to adversarial perturbation. The transferability of the untargeted unconstrained ZOO attack mode demonstrated no significant variations when compared to the targeted unconstrained attack modes. The findings suggest that ZOO *AEs* do not exhibit notable transferability among the examined models that share similar structures.

Transferability Scenario 2a: Our objective in this scenario is to investigate the transferability of *AEs* between DL models with different structures. To accomplish this, we generated DeepFool *AEs* on DNN1 model and measured the *ASR* on the same model. Subsequently, we transferred these *AEs* to a different and more complex DBN model. The results of this experiment are summarized in Table X. It is noticed that a significant portion of DeepFool *AEs*, regardless of whether they were generated under constrained or unconstrained settings or in targeted or untargeted attack modes, did not exhibit transferability to the DBN model. It appears that when *AEs* are transferred from simple model structures to complex and fundamentally different model structures, they lose their evasiveness. A relatively similar results were observed when transferring the constrained and unconstrained C&W *AEs*. The transferability of such *AEs* was limited before constraints. However, transferring C&W *AEs* under the untargeted attack mode was higher compared to the targeted mode.

Transferability Scenario 2b: We explored the transferability of ZOO *AEs* across different model architectures. Specifically, the ZOO *AEs* generated by attacking the DT model were transferred to the DNN1 model. The *ASR* of the DNN1 model was evaluated on the clean model when exposed to the transferred ZOO *AEs*. The experiments were conducted considering both constrained and unconstrained ZOO *AEs* under both targeted/untargeted attack modes. Table XII demonstrates our results. The transferability of ZOO *AEs* to a completely different model structure is notably limited. The *ASR* experienced a significant decline to 0.004 when the DNN1 model was exposed to the transferred ZOO *AEs*. We noticed that the ZOO *AEs* did not achieve a high *ASR* on the original ML model. Therefore, transferring these *AEs* also had no significant impact on the accuracy of the target model. One

TABLE IX
TRANSFERABILITY OF WB AES AMONG MODELS WITH SIMILAR STRUCTURES (SCENARIO 1A)

		Before constraints		After constraints	
		ASR against DNN1	**ASR after transferability**	**ASR against DNN1**	**ASR after transferability**
Untargeted attack	DeepFool	99.99%	99.80%	0.0044	0.004
	C&W	99.30%	51.80%	42.2%	36.6%
Targeted attack	DeepFool	99.78%	63.30%	0.001	0.001
	C&W	99.54%	41.90%	51%	36.91%

* **ASR after transferability**: ASR when transfer AEs from DNN1 to DNN2

TABLE X
TRANSFERABILITY OF WB AES AMONG MODELS WITH DIFFERENT STRUCTURES (SCENARIO 2A)

		Before constraints		After constraints	
		ASR against DNN1	**ASR after transferability**	**ASR against DNN1**	**ASR after transferability**
Untargeted attack	DeepFool	99.8%	1%	0.005	0.001
	C&W	99.30%	12.7%	42.2%	11.6%
Targeted attack	DeepFool	99.98%	0.001	0.001	0.001
	C&W	99.6%	4%	50.9%	3%

* **ASR after transferability**: ASR when transfering AEs from DNN1 to DBN

TABLE XI
TRANSFERABILITY OF BB AES AMONG MODELS WITH SIMILAR STRUCTURES (SCENARIO 1B)

		Before constraints		After constraints	
		ASR against DT	**ASR after transferability**	**ASR against DT**	**ASR after transferability**
Untargeted attack	ZOO	19.7%	0.0174	19.5%	0.018
Targeted attack	ZOO	10.4%	0.008	10.3%	0.008

* **ASR after transferability**: ASR when transfer AEs from DT to GB

TABLE XII
TRANSFERABILITY OF BB AES AMONG MODELS WITH DIFFERENT STRUCTURES (SCENARIO 2B)

		Before constraints		After constraints	
		ASR against DT	**ASR after transferability**	**ASR against DT**	**ASR after transferability**
Untargeted attack	ZOO	19.5%	0.004	19.7%	0.003
Targeted attack	ZOO	10.4%	0.003	10.3%	0.003

* **ASR after transferability**: ASR when transfer AEs from DT to DNN1

explanation is that ZOO *AEs* rely on a local search strategy to find perturbations that lead to misclassifications when applied to ML models, which make such perturbations sensitive to a specific model structure and decision boundaries.

3) Adversarial Training: Adversarial training is one of the defense mechanisms used to strengthen ML/DL models against adversarial attacks [17]. It involves generating then using *AEs* into the model's training process. By exposing the model to *AEs* during the training process, its performance in detecting similar *AEs* is supposed to improve. In traditional ML approaches, the model's loss function measures the deviation between model's predictions and the ground-truth labels. However, when employing adversarial training, an additional loss function is used to evaluate the deviation between the model's predictions on *AEs* and the targeted class determined by the attacker. As a result, the core of adversarial training lies in modifying the model's loss function, which becomes a weighted combination of the regular loss function for clean examples and a loss function specifically designed for *AEs*. By incorporating the adversarial loss function into the training process, the model learns to defend against potential adversarial attacks by explicitly considering their impact during optimization.

We conducted a series of experiments to assess the effectiveness of adversarial training on the *ASR* for both constrained and unconstrained *AEs*. We investigate the effects of varying the percentage of *AEs* with respect to clean examples during the model training phase and through considering both targeted and untargeted attacks modes. Furthermore, we evaluated the robustness of adversarial training under two distinct cases: same attack signature (i.e., SS) and different attack signature(i.e., DS). In the first case, we retrained the model using the same type of *AEs* that were employed to attack it initially. In the second case, we retrained the model using *AEs* from a different type of attack that is used against that model.

We further explored the first case through two different approaches: similar model same signature (SMSS) and different model same signature (DMSS). The SMSS approach involves two models $M1$ and $M2$ having a similar structures. Specifically, we attacked $M1$ and used the *AEs* generated from $M1$ to retrain $M2$. We then tested $M2$ on the *AEs* generated from $M1$ to evaluate the robustness of $M2$ after adversarial training. In this approach, $M1$ and $M2$ have similar structures in terms of the number and structure of layers and the activation function used (e.g., DNN1 and DNN2).

The DMSS approach involves retraining and testing using *AEs* that have SS. We used two models $M1$ and $M3$ with different structure or different categories (e.g., DNN1 and GB). Figure 5 describes the DMSS approach where we trained two different models $M1$ and $M3$ on clean examples, then

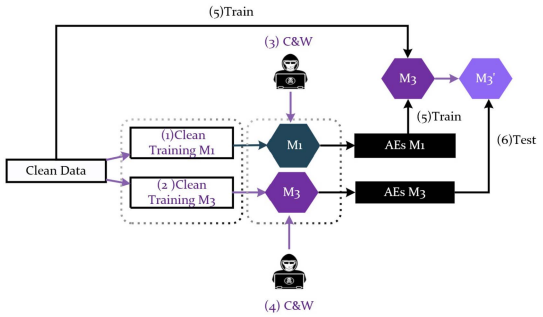


Fig. 5. Description of adversarial training process.

TABLE XIII
ADVERSARIAL TRAINING SCENARIOS

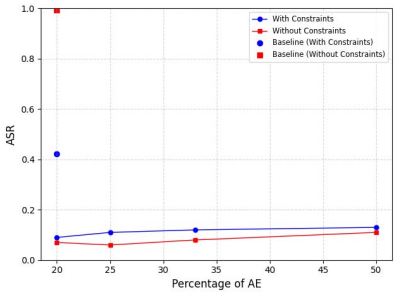
Scenario	AEs for training	Model	AEs for testing
SMSS	C&W	DNN1, DNN2	C&W
SMSS	ZOO	DT, GB	ZOO
SMDS	DeepFool	DNN1, DNN2	C&W
DMSS	C&W	DNN1, DNN3	C&W
DMDS	C&W	DNN1, GB	ZOO
DMDS	DeepFool	DNN1, DNN3	C&W

we generated C&W *AEs* by attacking $M1$ and $M3$. We then retrained $M3$ using the clean data and the *AEs* generated on $M1$. This results in a trained Model $M3'$ which was then tested on *AEs* generated from $M3$.

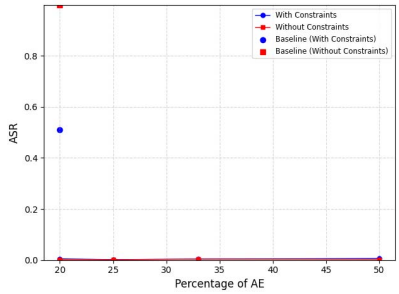
We examined the second case using two distinct approaches: similar model different signature (SMDS) and different model different signature (DMDS). Table XIII shows the characteristics of each approach, specifically the *AEs* used for training, the models employed, and the *AEs* used for testing model's robustness. In this series of experiments, we created a DNN3 DL model with a higher degree of intricacy, consisting of six layers, each with a different number of neurons. Each of the first five layers includes 256 neurons, and the output layer includes 2 neurons. We used the *ReLU* activation function for all layers except for layer 4, where the *Tanh* activation function is employed. The *Adamax* optimizer is employed for training the DNN3 model.

The results of the experiments on adversarial training while varying the percentage of *AEs* used for training are shown in Figure 6. The following scenarios outline the specific details of each experimental case.

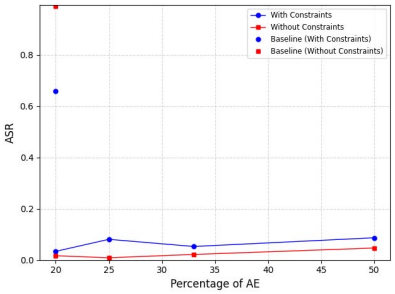
1) *Same Signature:*
   a) SMSS-WB attacks: we initially targeted the DNN2 model using C&W attack. Subsequently, we retrained the DNN1 model using C&W *AEs* generated through the attack on the DNN2 model. Figure 6(a) and Figure 6(b) show the results for this experiment. Before applying adversarial training, the *ASR* on the DNN1 model using constrained and unconstrained untargeted *AEs* was 42.20% and 99.30%, respectively. However, after applying adversarial training, the *ASR* significantly dropped to 9% and 7%. We noticed that the *ASR* was comparable when varying the percentages of *AEs*. We observed that both constrained and unconstrained *AEs* improved the model's robustness. Furthermore, we observed that retraining the model with just 20% of C&W *AEs* was sufficient to enhance its robustness. A similar behavior was observed under the targeted attack mode, but the targeted attack led to a more robust model as it yielded lower *ASR* (Figure 6(b)).
   b) DMSS-WB attacks: we targeted the DNN1 model using C&W attack. Subsequently, we retrained the DNN3 model using C&W *AEs* generated through attacking the DNN1 model. Figure 6(c) and Figure 6(d) show the results for this experiment. Before applying adversarial training, the *ASR* of the DNN3 model using the constrained and unconstrained untargeted attacks was 66.6% and 99.6% respectively. After applying adversarial training, the *ASR* significantly dropped to 0.034% and 0.017% (Figure 6(c)). Our observations indicated that both unconstrained and constrained adversarial training under the untargeted mode improved the model's resilience to the tested *AEs*. Under the untargeted mode, both the constrained and unconstrained *AEs* exhibited a similar behavior in terms of robustness. The targeted mode yielded relatively similar results. The DMSS adversarial training approach improved the model's resilience to the C&W attack using both constrained and non-constrained targeted attacks (Figure 6(d)).
   c) SMSS-BB attacks: In this experiment, we generated ZOO *AEs* based on a DT model, then we used the resulting *AEs* to train and test a GB model. Figures 6(e) 6(f) show the results for this experiment. Prior to the adversarial training, the *ASR* on the GB model for constrained and unconstrained untargeted attacks was 35.4% and 37%, respectively. However, after applying adversarial training, the *ASR* significantly dropped to 0.072 and 0.048 using only 20% of *AEs* (Figure 6(e)). Both constrained and unconstrained *AEs* improved the model's robustness in untargeted attacks mode. The values of *ASR* were comparable when varying the percentages of *AEs*.
   We observed slight variations in the *ASR* when using different percentages of *AEs*. Using 33% of *AEs* with constraints demonstrated the highest effectiveness in reducing the *ASR* under the targeted mode. In contrast, employing 20% of the unconstrained *AEs* exhibited the most significant reduction in *ASR* as depicted in Figure 6(f). After applying adversarial training, we noticed a significant decrease in the *ASR*. For the constrained *AEs*, the *ASR* dropped from 19.7% to 0.005. Similarly, for the unconstrained *AEs*, the *ASR* decreased from 21.5% to 0.002.
2) *Different Signature:*
   a) SMDS-WB attacks: In this scenario, we utilized C&W adversarial attacks to target the DNN1 model and performed adversarial training using those *AEs*. Subsequently, we employed the DeepFool *AEs* generated by attacking a DNN2 model



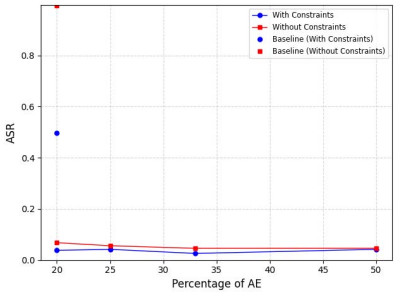
(a) SMSS-WB: adversarial training on C&W and testing on C&W-untargeted



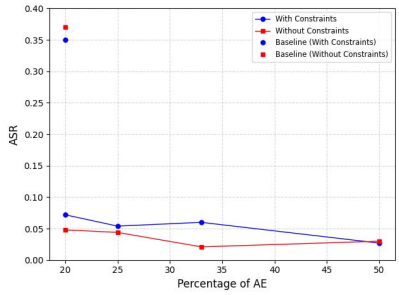
(b) SMSS-WB: adversarial training on C&W and testing on C&W-targeted



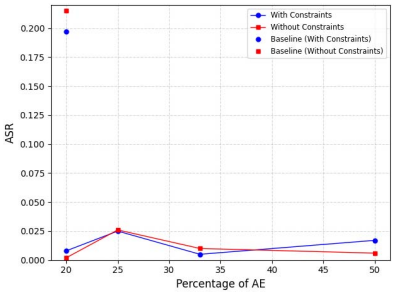
(c) DMSS-WB: adversarial training on C&W and testing on C&W-untargeted



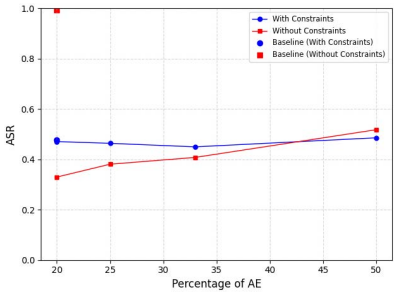
(d) DMSS-WB: adversarial training on C&W and testing on C&W-targeted



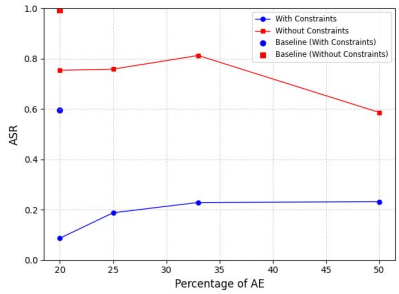
(e) SMSS-BB: adversarial training on ZOO and testing on ZOO-untargeted



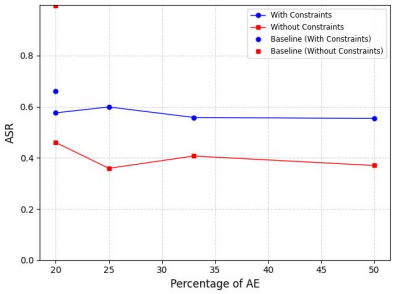
(f) SMSS-BB: adversarial training on ZOO and testing on ZOO-targeted



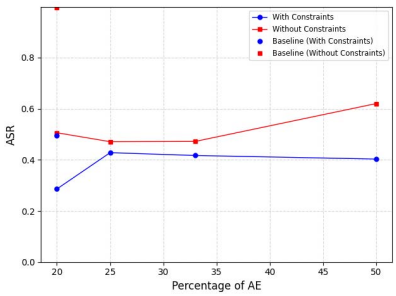
(g) SMDS-WB: adversarial training on DeepFool and testing on C&W-untargeted



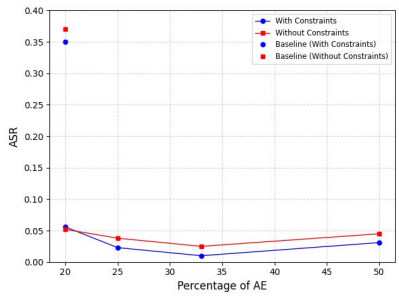
(h) SMDS-WB: adversarial training on DeepFool and testing on C&W-targeted



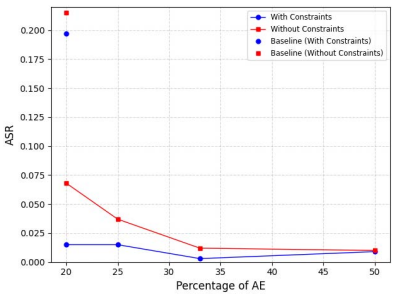
(i) DMDS-WB: adversarial training on DeepFool and testing on C&W-untargeted



(j) DMDS-WB: adversarial training on DeepFool and testing on C&W-targeted



(k) DMDS-BB: adversarial training on C&W and testing on ZOO-untargeted



(l) DMDS-BB: adversarial training on C&W and testing on ZOO-targeted

Fig. 6. Results of adversarial training on constrained/unconstrained AEs for both targeted/untargeted attacks.

to assess the robustness of the DNN1 model. Figures 6(g) and 6(h) show the results for this experiment. Prior to applying adversarial training, the *ASR* of the DNN1 model on untargeted attacks was 47.7% for the constrained *AEs* and 99.3% for the unconstrained *AEs*. The constrained untargeted *AEs* do not add much robustness to the DNN1 model, since the *ASR* dropped from 47.7% to 45% using 33% of *AEs* for model retraining. However, the unconstrained *AEs* significantly decreased the *ASR* from 99.3% to 33% (Figure 6(g)). The results were significantly different under the targeted *AEs*. As shown in Figure 6(h)) constrained *AEs* increased the robustness of the model.

b) DMDS-WB attacks: In this experiment, we initiated C&W adversarial attack against a DNN3 model. Subsequently, we retrained the

DNN3 model using DeepFool *AEs* generated by attacking a DNN1 model. We then tested the DNN3 against C&W *AEs*. The results of this experiment are illustrated in Figure 6(i) and 6(j). Overall, both the constrained and unconstrained *AEs* improved model robustness. For the untargeted attack mode, we noticed that the unconstrained *AEs* add more robustness to the DNN3 model compared to the constrained Examples. The constrained *AEs* performed better under targeted attacks (Figure 6(j)). For both untargeted and targeted modes, varying the percentages of *AEs* resulted in an analogous *ASR*. In general, 20% and 25% yielded the best results in most of the experiments.

c) DMDS-BB attacks: In this experiment, we generated ZOO *AEs* against a GB model. Subsequently, we retrained the GB model using C&W *AEs* generated by attacking a DNN1 model. The GB model was then tested against ZOO *AEs*. The results of this scenario are illustrated in Figures 6(k) and 6(l). The *ASR* dropped significantly to 0.056 and 0.052 for constrained and unconstrained untargeted attacks. This finding indicates that the adversarial training process using 20% percentage of *AEs* effectively enhanced the model's robustness against untargeted attacks (Figure 6(k)). Likewise, in the scenario of targeted attacks, both constrained and unconstrained *AEs* improved model robustness using adversarial training defense (Figure 6(l)).

4) Time Analysis of AEs Generation: We conducted another experiment to measure the time taken to generate AEs for each model-attack combination (Figure 7). The bars represent the time it takes to generate AEs for each type of attack with constraints(C) and without Constraints (NoC). For each attack, we changed the optimizer or the type of the ML model used to measure results consistency across different scenarios. We observed no significant differences between the time to generate AEs for the DeepFool attack. The creation of AEs using DeepFool, which is a WB attack, is relatively fast, which can be attributed to the attack's inherent efficiency in generating AEs. In addition, DeepFool is a directional attack that focuses on finding the smallest possible perturbation that can fool the target model making it an efficient attack compared to others. C&W takes longer to generate AEs, specifically when applying constraints. This is attributed to the approach that C&W attack uses to generate AEs. It is based on the constrained optimization problem of minimizing the *L2* norm of the perturbation while ensuring that the model is fooled. The ZOO attack also takes longer time than DeepFool to generate AEs. BB attacks such as ZOO are generally more time-consuming than white-box attacks. This is because they lack direct access to model gradients and must estimate them. To determine if the differences in attack times with and without constraints are statistically significant, we applied a paired sample

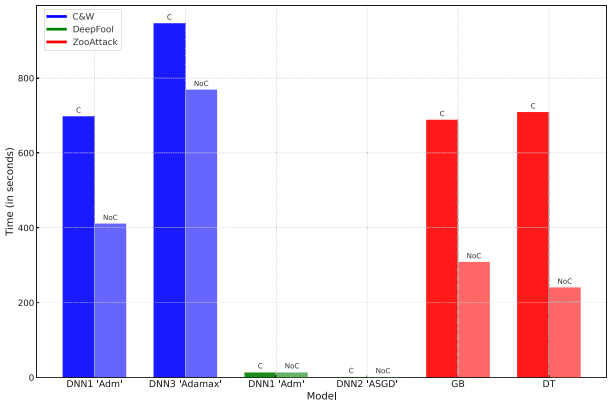


Fig. 7. AEs generation: time analysis.

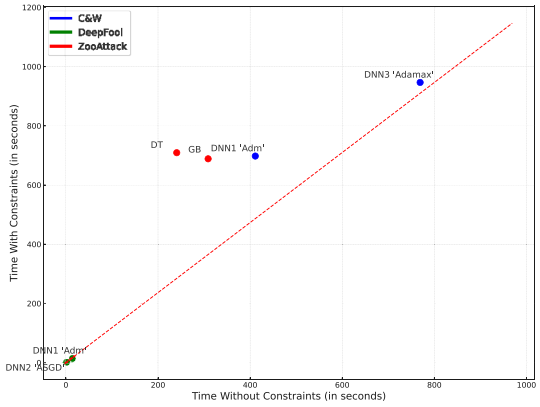


Fig. 8. Scatter plot analysis of AEs generation time: model comparison and attack type Visualization."

t-test. The null hypothesis we tested is that there is no significant difference between the two cases with and without constraints. With t-value $= 2.7468$, p-value $= 0.0405$, and significance level $\alpha = 0.05$, the p-value is less than α. That is, there is a statistically significant difference between the times taken to generate AEs with constraints and without constraints.

To determine the type of the relationship between the times with constraints and without constraints, we first plot the execution times on the scatter plot (figure 8) to assess the relationship. We then calculated the Pearson correlation coefficient, which may indicate the strength of the linear relationship between the two sets of values. We then fitted two types of models: linear and exponential to the set of values and determined which one has the best fit based on the values of the coefficient of determination R^2.

The scatter plot shows the relationship between the times without constraints and with constraints for the attack scenarios shown in figure 7. Each point represents a model-attack combination. The red dashed line represents the diagonal where both times would be equal. Most points lie above the diagonal, which indicates that for most model-attack combinations, the time with constraints is greater than without constraints. The points do not cluster around a straight line, which shows that

the relationship is not perfectly linear. The spread in the points indicates a variability in how constraints affect different model-attack combinations.

We found that Pearson correlation coefficient $= 0.8914$, which indicates a strong positive linear relationship between the times with constraints and without constraints. We then fitted both linear and exponential models by comparing their R^2 values. For the linear model, the $R^2 = 0.7947$ which suggests that approximately 79.47% of the variance in the time with constraints is explained by the time without constraints using a linear relationship. This supports our earlier observation from the Pearson correlation coefficient, which indicated a linear relationship. The R^2 value for the exponential model, which $= -\infty$ indicates that the exponential model is not suitable for this difference. Although constraints lead to a longer time for generating AEs, this increase in time does not follow an exponential trend across various model-attack combinations.

5) Comparison With Existing Approaches: We conducted a comparative analysis between applying our constraints to adversarial attacks and the approach proposed by Sheatsley et al. in [8]. The authors proposed an algorithm to incorporate network constraints into JSMA WB adversarial attack. Their objective was to explore the effects of enforcing a set of constraints they developed on both model accuracy and transferability. To conduct this comparison, we employed JSMA attack and enforced our constraints during the process of generating *AEs*. To maintain consistency in the experimental setup, we adopted the same evaluation metrics outlined by Sheatsley et al. Specifically, we measured *ASR* and transferability on the same dataset (UNSW-NB15). They investigated the transferability of the constrained *AEs* based on two approaches, intra-transferability and inter-transferability. The intra-transferability involves measuring the transferability of *AEs* within the same model. This was accomplished by partitioning the dataset into five distinct subsets labeled as *A*, *B*, *C*, *D*, and *E*. The evasiveness of the crafted *AEs* using dataset *A* was then tested on datasets *B*, *C*, *D*, and *E*. To ensure comparable transferability results, we divided the UNSW-NB15 dataset into five separate partitions using a stratified shuffle-split technique [8]. This technique ensures that each split preserves the class proportions of the original dataset, facilitating better learning and model generalization. The inter-transferability examined the transferability of *AEs* across models that use different learning mechanisms. Specifically, the authors considered four ML models namely DT, Logistic Regression (LR), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).

We adopted a similar model framework as in [8], and we incorporated analogous architectures and hyperparameters in our comparison. We used a DBN model with a learning rate equal to 0.01 and using 10 epochs. Table XIV illustrates the results of our comparison. The second column from left represents the source models used for generating *AEs*, while the top row represents the target models for testing *AEs*. The values represent both inter and intra-transferability. The diagonal entries denote the *ASR* when the source and target models are the same.

The comparison shown in table XIV indicates that our constraints when employed to JSMA yield a lower *ASR* compared to the constraints proposed by Sheatsley et al. This signifies that our constraints introduced higher robustness to the model by constraining the perturbation magnitude available to the attacker. Subsequently, the transferability of *AEs* was lower using our approach. This was not the case for Inter-transferability values, which show that our constraints lead to more transferable *AEs* when applied to certain models such as LR and SVM. This observation aligns with the results obtained by Sheatsley et al. who discovered that inter-technique transferability rates were largely model-dependent.

We further assessed our constraints when generating adversarial sketches using the perturbation histogram approach proposed in [8]. Sketches are universal perturbations that adhere to network domain constraints. Similar to the authors in [8], we generated those sketches through: 1) creating AEs using the JSMA attack, 2) constructing a perturbation histogram to identify the most influential set of features having the highest potential to fool the targeted model, 3) crafting adversarial sketches by selecting the top n features from the perturbation histogram, and 4) employing these sketches to attack the model and report both *ASR* and transferability. In our experiment, we adopted $n = 9$ for the UNSW-NB15 dataset to ensure a consistent comparison. The results presented in Table XIV demonstrated that when our constraints are applied to create adversarial sketches, they lead to a relatively lower average (ASR) compared to the constraints proposed by the authors in [8]. This indicates that our constraints enhanced the model's robustness even when attackers focus their effort to perturb the most influential features. While our approach results in lower intra-transferability of adversarial sketches, this does not hold for inter-transferability values of models such as LR and SVM, which shows that inter-transferability of adversarial sketches relies on the model used.

6) Summery of Findings: The summary of our experiments can be condensed into three investigated areas as follows:

- *Attack effectiveness using constrained/unconstrained AEs:* During the assessment of the attack effectiveness of both constrained and unconstrained *AEs*, we made several observations. First, we found that the existence of constraints did not significantly impact the *ASR* for the ZOO BB attack. Both constrained and unconstrained *AEs* displayed comparable performance in successfully fooling the model. However, when it came to the DeepFool WB attack, the *ASR* was significantly affected by applying network domain constraints.

TABLE XIV
COMPARISON WITH OTHER APPROACHES

Approach	Attack	Model	M_A	M_B	M_C	M_D	M_E	LR	SVM	DT	KNN
Our Approach	JSMA	M_A	60%	15%	79%	79%	72%	100%	97%	9%	4%
		M_B	50%	56%	32%	52%	34%	100%	93%	1%	6%
		M_C	60%	14%	98%	86%	87%	99%	96%	2%	4%
		M_D	49%	30%	61%	97%	55%	99%	97%	3%	5%
		M_E	60%	38%	98%	91%	95%	99%	96	8%	8%
	Histogram	M_A	100%	98%	98%	17%	100%	16%	16%	70%	94%
		M_B	22%	32%	100%	100%	16%	16%	16%	70%	88%
		M_C	100%	100%	100%	100%	100%	15%	15%	69%	83%
		M_D	16%	21%	16%	100%	100%	16%	16%	65%	78%
		M_E	100%	17%	16%	100%	100%	16%	16	67%	81%
The approach in [8]	JSMA	M_A	100%	97%	92%	96%	96%	72%	81%	29%	53%
		M_B	50%	100%	72%	94%	71%	62%	62%	12%	26%
		M_C	73%	81%	100%	93%	87%	71%	76%	19%	49%
		M_D	69%	64%	55%	100%	59%	53%	48%	8%	25%
		M_E	66%	80%	90%	96%	100%	66%	69%	15%	38%
	Histogram	M_A	88%	99%	95%	96%	96%	99%	96%	29%	37%
		M_B	99%	100%	99%	100%	99%	99%	100%	20%	62%
		M_C	93%	97%	100%	77%	95%	73%	100%	27%	41%
		M_D	80%	99%	100%	100%	99%	74%	99%	13%	30%
		M_E	98%	100%	100%	94%	92%	85%	100	28%	39%

The constrained DeepFool *AEs* demonstrated a significant decrease in their ability to deceive the intrusion detection models compared to their unconstrained ones. Constraints applied to C&W WB attack showed a moderate impact on the *ASR* compared to that DeepFool attack. The observation that constraints have larger effect on the *ASR* for WB attacks compared to BB attacks suggests that the knowledge and accessibility of the model's architecture and parameters play a role in the effectiveness of constraints. This shows that the applied constraints appear to reduce the attack surface for WB attacks. We also found that the validity of the resulting *AEs* does not depend on the used threat model. Before applying any constraints, the percentage of the valid *AEs* was higher for the ZOO BB attacks compared to both WB attacks.

- *Transferability of the constrained/unconstrained AEs:* We investigated the transferability of the constrained and unconstrained *AEs* under two scenarios, similar model structures and different model structures. When considering similar model structures, we observed that the unconstrained DeepFool *AEs* yielded good transferability, while the constrained *AEs* exhibited a low transferability. In contrast, both the constrained and unconstrained C&W *AEs* demonstrated a reasonable transferability between similar models. For the ZOO *AEs*, both constrained and unconstrained *AEs* exhibited a low transferability. The low transferability observed in ZOO attack can be attributed to its local search process to generate *AE*. When generating adversarial perturbations, ZOO relies on specific features that are distinctive to the original model(the one on which the attack was trained), which may not align with the vulnerabilities of the targeted model.
  The transferability analysis of different model structures revealed that the *AEs* generated by the three attack methods exhibited limited ability to fool models with different architectures. The differences in model boundaries between the source and target architectures lead to significant impacts on transferability. Regardless of the threat model used, the transferability of *AEs* between models with different architectures did not appear to be significantly influenced by the application of constraints.
- *Adversarial training using the constrained/unconstrained AEs:* We assessed the robustness of different models against constrained and unconstrained *AEs* through adversarial training. When adversarial training is applied using *AEs* with different signatures to retrain the model, we found that the drop rate in the *ASR* was lower compared to the same signature retraining. Overall, we found that similarities and differences between the signatures of *AEs* used for retraining and testing play a reasonable role in the effectiveness of adversarial training against WB attacks. The adversarial training was less effective to mitigate *AEs* when they are 1) unknown to the model, 2) constrained, or 3) when the attack is untargeted.

Table XV presents a summary of the attacks we experimented with, their effectiveness, transferability within similar model structures, transferability to different model structures, and their utility in adversarial training. To represent the impact levels on each of the investigated areas, we used the abbreviations "H" for High, "M" for Medium, and "L" for Low. The following observations can be drawn from this table

a) When no constraints are applied, WB adversarial attacks demonstrate a comparable high level of transferability. However, this pattern changes when constraints are imposed. Based on our experiments, we assert that transferability of WB *AEs* under constraints depends on the characteristics of the attack used.

TABLE XV
SUMMARY OF RESULTS

Attacks	WB/BB	Attack Mode	ASR		Transferability (similar models)		Transferability (different models)		Effectiveness of adver training (same signature)		Effectiveness of adver training (different signature)	
			C	NC	C	NC	C	NC	C	NC	C	NC
DeepFool	WB	Targeted	L	H	L	H	L	L	L	H	L	H
		Untargeted	L	H	L	H	L	L	L	H	L	H
C&W	WB	Targeted	H	H	H	H	L	L	H	H	H	L
		Untargeted	H	H	H	H	L	L	H	H	L	H
ZOO	BB	Targeted	L	L	L	L	L	L	H	H	H	H
		Untargeted	M	M	L	L	L	L	H	H	H	H

* **C**: with constraints, **NC**: no constraints

b) Both WB and BB *AEs* do not transfer well between models with different architectures. Applying constraints do not seem to have significant impact on transferability between models with different architectures.
c) Effectiveness of using *AEs* generated by each attack as a defense mechanism was effective in most of the experiments. The results show a promising direction of using existing signatures of *AEs* to mitigate novel variations of adversarial attacks or zero-day adversarial attacks against NIDS.

VI. CONCLUSION

This paper investigated the impact of the constrained WB/BB adversarial examples against DL/ML based NIDS. We created a set of network domain constraints and applied them to generate WB and BB-based AEs. The examination of the AEs encompassed three key perspectives: 1) evaluating the success rate of the attack, 2) assessing the transferability of the AEs to similar/different models, and 3) analyzing their effectiveness in defending against adversarial attacks through adversarial training. We found that WB attacks exhibited a high ASR without any constraints. Adding constraints limited the success rate of those attacks. However, the impact of these constraints can vary depending on the specific characteristics of the attack. We found that constraints have less impact on transferability of AEs to models with different learning approaches compared to the original or source models that generate those AEs. Both WB and BB AEs do not transfer well between models with different structures. The results indicated a promising potential for utilizing existing signatures of AEs to mitigate novel variations or zero-day adversarial attacks against NIDSs.

This study is limited to experiments on three types of attacks. We plan to extend it to examine other existing attacks. We also plan to extend this work to consider AEs that are generated from a raw network traffic such as pcap files (i.e., problem space attacks). Finally, we plan to extend our approach to detect zero-day AEs against NIDSs.

REFERENCES

[1] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6882–6897, Aug. 2020.
[2] J. Lansky et al., "Deep learning-based intrusion detection systems: A systematic review," *IEEE Access*, vol. 9, pp. 101574–101599, 2021.
[3] 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.
[4] S. Gurung, M. K. Ghose, and A. Subedi, "Deep learning approach on network intrusion detection system using NSL-KDD dataset," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 3, pp. 8–14, 2019.
[5] O. Ibitoye, 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.
[6] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[7] H. Xu et al., "Adversarial attacks and defenses in images, graphs and text: A review," *Int. J. Autom. Comput.*, vol. 17, pp. 151–178, Mar. 2020.
[8] R. Sheatsley, N. Papernot, M. J. Weisman, G. Verma, and P. McDaniel, "Adversarial examples for network intrusion detection systems," *J. Comput. Secur.*, vol. 30, no. 5, pp. 727–752, 2022.
[9] S. Guo, J. Zhao, X. Li, J. Duan, D. Mu, and X. Jing, "A black-box attack method against machine-learning-based anomaly network flow detection models," *Secur. Commun. Netw.*, vol. 2021, pp. 1–13, Apr. 2021.
[10] N. Alhussien and A. Aleroud, "A novel poisoning attack on few-shot based network intrusion detection," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, 2023, pp. 1–5.
[11] N. Alhussien, A. Aleroud, R. Rahaeimehr, and A. Schwarzmann, "Triggerability of backdoor attacks in multi-source transfer learning-based intrusion detection," in *Proc. IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol. (BDCAT)*, 2022, pp. 40–47.
[12] 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.
[13] D. Shu, N. O. Leslie, C. A. Kamhoua, and C. S. Tucker, "Generative adversarial attacks against intrusion detection systems using active learning," in *Proc. ACM Workshop Wireless Security Mach. Learn.*, 2020, pp. 1–6.
[14] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu, "Adversarial attacks against network intrusion detection in IoT systems," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10327–10335, Jul. 2021.
[15] E. Alhajjar, P. Maxwell, and N. Bastian, "Adversarial machine learning in network intrusion detection systems," *Expert Syst. Appl.*, vol. 186, Dec. 2021, Art. no. 115782.
[16] A. Chernikova and A. Oprea, "Fence: Feasible evasion attacks on neural networks in constrained environments," *ACM Trans. Privacy Secur.*, vol. 25, no. 4, pp. 1–34, 2022.
[17] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," 2021, *arXiv:2102.01356*.
[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[19] Y. Dong et al., "Boosting adversarial attacks with momentum," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9185–9193.
[20] R. Wiyatno and A. Xu, "Maximal Jacobian-based saliency map attack," 2018, *arXiv:1808.07945*.
[21] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "Ead: Elastic-net attacks to deep neural networks via adversarial examples," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–19.
[22] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Artificial Intelligence Safety and Security*. Boca Raton, FL, USA: Chapman and Hall/CRC, 2018, pp. 99–112.
[23] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Berlin, Germany: Springer, 2003.

[24] C. Szegedy et al., "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[26] M. Clerc, *Particle Swarm Optimization*. Hoboken, NJ, USA: Wiley, 2010, pp. 1–17.
[27] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, pp. 65–85, Jun. 1994.
[28] K. Kotecha et al., "Enhanced network intrusion detection system," *Sensors*, vol. 21, no. 23, p. 7835, 2021.
[29] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," 2018, *arXiv:1801.02610*.
[30] H. A. Alatwi and A. Aldweesh, "Adversarial black-box attacks against network intrusion detection systems: A survey," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, 2021, pp. 0034–0040.
[31] S. Alemany and N. Pissinou, "The dilemma between data transformations and adversarial robustness for time series application systems," in *Proc. SafeAI@ AAAI*, 2022, pp. 1–8.
[32] C. Zhang, P. Benz, A. Karjauv, J. W. Cho, K. Zhang, and I. S. Kweon, "Investigating top-k white-box and transferable black-box attack," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15085–15094.
[33] M. A. Ayub, W. A. Johnson, D. A. Talbert, and A. Siraj, "Model evasion attack on intrusion detection systems using adversarial machine learning," in *Proc. Annu. Conf. Inf. Sci. Syst. (CISS)*, 2020, pp. 1–6.
[34] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016, *arXiv:1611.02770*.
[35] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2137–2146.
[36] D. Lee, S. Moon, J. Lee, and H. O. Song, "Query-efficient and scalable black-box adversarial attacks on discrete sequential data via Bayesian optimization," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 12478–12497.
[37] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.
[38] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 1277–1294.
[39] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa, "GADoT: Gan-based adversarial training for robust DDoS attack detection," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2021, pp. 119–127.
[40] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," in *Proc. Adv. Knowl. Discov. Data Min.*, 2022, pp. 79–91.
[41] A. Granados, M. S. Miah, A. Ortiz, and C. Kiekintveld, "A realistic approach for network traffic obfuscation using adversarial machine learning," in *Proc. Int. Conf. Decis. Game Theory Secur.*, 2020, pp. 45–57.
[42] 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.*, 2019, pp. 14–21.
[43] H. Jmila and M. I. Khedher, "Adversarial machine learning for network intrusion detection: A comparative study," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109073.
[44] C. Zhang, X. Costa-Perez, 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.
[45] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 1332–1349.
[46] K. Sauka, G.-Y. Shin, D.-W. Kim, and M.-M. Han, "Adversarial robust and explainable network intrusion detection systems based on deep learning," *Appl. Sci.*, vol. 12, no. 13, p. 6451, 2022.
[47] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.
[48] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 39–57.
[49] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, 2018, pp. 559–564.
[50] N. Wang, Y. Chen, Y. Xiao, Y. Hu, W. Lou, and Y. T. Hou, "MANDA: On adversarial example detection for network intrusion detection system," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1139–1153, Mar./Apr. 2023.
[51] J. Wang, J. Pan, I. AlQerm, and Y. Liu, "Def-IDS: An ensemble defense mechanism against adversarial attacks for deep learning-based network intrusion detection," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2021, pp. 1–9.
[52] J. Clements, Y. Yang, A. A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, 2021, pp. 01–08.
[53] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," 2018, *arXiv:1807.04457*.
[54] A. Aleroud and G. Karabatis, "Contextual information fusion for intrusion detection: A survey and taxonomy," *Knowl. Inf. Syst.*, vol. 52, pp. 563–619, Feb. 2017.
[55] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using Trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019.
[56] S. Garcia, A. Parmisano, and M. J. Erquiaga, 2020, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," IoT-23 Dataset, Zenodo. [Online]. Available: http://doi.org/10.5281/zenodo.4743746
[57] C. Seger, *An Investigation of Categorical Variable Encoding Techniques in Machine Learning: Binary Versus One-Hot and Feature Hashing*, KTH Royal Inst. Technol., Stockholm, Sweden, 2018.
[58] T. A. Alhaj, M. M. Siraj, A. Zainal, H. T. Elshoush, and F. Elhaj, "Feature selection using information gain for improved structural-based alert correlation," *PloS ONE*, vol. 11, no. 11, 2016, Art. no. e0166017.
[59] O. Shamir, "Without-replacement sampling for stochastic gradient methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–36.

Nour Alhussien is currently pursuing the Ph.D. degree with the School of Computer and Cyber Sciences, Augusta University. Her research focuses on intrusion detection and adversarial machine learning attacks. With expertise on deep learning and data analysis, she has publications in conferences like NOMS and IEEE/ACM Conference on Big Data Applications.

Ahmed Aleroud is an Associate Professor with the School of Computer and Cyber Sciences. His research has been published in journals, such as IEEE Transactions, *ACM Digital Threats Research and Practice*, *Computers and Security*, *Information Security and Applications*, *Information System Frontiers*, *Knowledge*, and *Information Systems*. His work has been supported by NSF, the Office of Naval Research, European IP Networks, and the Department of Energy. His research work focuses on machine learning computational approaches and social approaches to combat security and privacy attacks, including social engineering attacks, computer network intrusions, disinformation, social media propaganda, state-linked social media attacks, online extremism, and re-identification of private information on individuals or devices. He is also working on adversarial attacks on security systems.

Abdullah Melhem received the master's degree in data science from the Jordan University of Science and Technology. He is a Graduate Research Assistant with School of Computer and Cyber Sciences, Augusta University. He has a strong background in advanced data analytic. Previously, he worked as a Data Scientist with KADDB Design and Development Center, Jordan.

Samer Y. Khamaiseh (Member, IEEE) is an Assistant Professor of Computer Science and Software Engineering with Miami University. His research focuses on cybersecurity with the integration of artificial intelligence. Specifically, adversarial machine learning, software-defined networking, software security, machine learning security, and network security. He discovers two new attacks that defeat state-of-the-art image classification neural networks. His main research interest on the area of intrusion detection systems for SDN fabric; often applied deep learning techniques to develop smart DDoS detection and prevention systems. In addition, he is applying adversarial machine learning attacks to develop adversarial traffic generation tools that penetrate real-life IDS/IPS systems. In the area of software security, his research focuses on integrating deep learning and knowledge graph on large-scale source code systems to develop state-of-the-art security vulnerability detection tools. Another area, he is heavily focusing on is adversarial machine learning.