Gremlins in the Network

How Adversarial AI Can Evade Network Detection

A technical paper prepared for presentation at SCTE TechExpo24

Kyle Haefner, Ph.D.
Principal Security Architect
CableLabs
K.haefner@cablelabs.com

Chad Schwenke
Security Software Engineer
CableLabs
C.Shwenke@cablelabs.com

Table of Contents

List of Figures

List of Tables

1. Introduction

Network traffic analysis plays a critical role in cybersecurity. Artificial Intelligence (AI) and Machine learning (ML) models are increasingly deployed to classify and identify malicious traffic. However, these models are susceptible to adversarial attacks where slight alterations in the data can cause the model to misclassify attacks. Adversarial AI in network traffic involves crafting malicious network traffic that appears benign to ML-based intrusion detection systems (IDS) and traffic classifiers. This paper explores how attackers can manipulate network traffic data to bypass detection and achieve their goals. We discuss techniques for generating adversarial network traffic and the challenges defenders face in mitigating these attacks.

2. Background

2.1. Overview of Adversarial AI

A large corpus of adversarial learning research has been done in the visual domain by examining how to perturb (adjust pixels in) images in a way that the human would still see the intended image, but a machine classifier would produce a predicted label that was very different from the expected label. Authors Goodfellow et al. wrote one of the foundational works on adversarial learning. They showed that small perturbations in the pixel data could create images that would look indistinguishable from the original image to a human, but the machine learning classifier would incorrectly classify the image. The authors also developed a simple and fast method for creating adversarial examples called the Fast Gradient Sign Method (FGSM) [1].

To counter this, Madry et al. developed a standard approach for improving model robustness against adversarial attacks by considering the worst-case examples during training. The simple take away from this work is that the only reliable method of withstanding adversarial attacks is to increase the capacity of the neural network i.e. train on more examples including some adversarial examples. [2]

Humans are generally good at ignoring noise in imagery and inferring correctly what the image is supposed to be, thus adversarial AI in the visual domain inherently produces results that are easy for humans to see even though the algorithm is drastically misled. Images have millions of features, the number of pixels times the color possibilities for each pixel. This means there is a lot of possible entropy, i.e. degrees of freedom in images to hide perturbations. For example, even simple images are highly dimensional in nature. Figure 1 shows the number 5 (left image of Figure 1) as part of the MNIST handwritten numbers. Each image in the MNSIT dataset is comprised of 28x28 pixels for a total of 784 pixels. Each pixel can have a 0-255 greyscale value (shown in middle image of Figure 1). This gives 784*255=199,920 degrees of freedom to manipulate the image in an adversarial manner. After we run the Fast Gradient Sign Method (FGSM) and generate a new image (shown in the right of Figure 1) that to a human still resembles a number 5 but the classifier now misclassifies as the number 0.

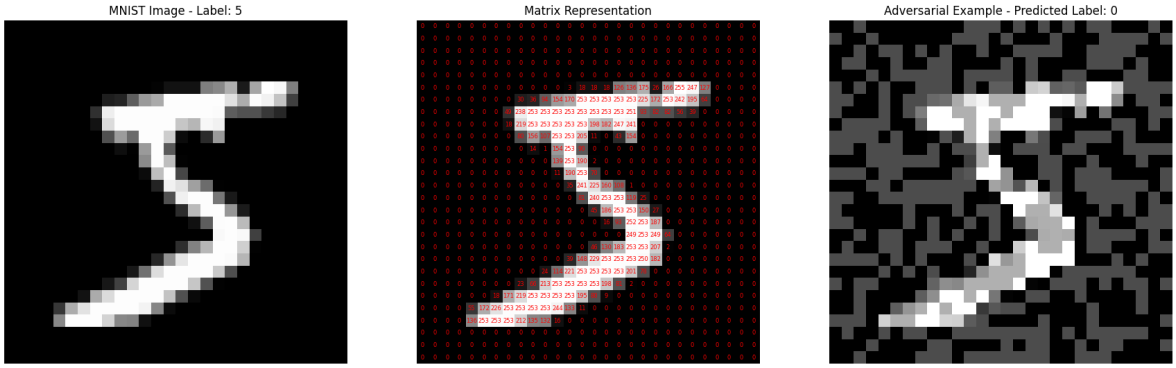

Figure 1: Number 5 from MNIST dataset, Matrix representation showing greyscale values, adversarial image misclassified as a 0.

2.2. Machine Learning for Network Traffic Analysis

AI and machine learning (ML) techniques in networking technologies have been around for many years and can be found in AI-Based detection in IDS (Intrusion Detection Systems) and IPS (Intrusion Prevention Systems). Sowmya et al. review 72 research papers that use AI-based mechanisms to detect attacks [3]. The authors show that AI-based IDSs can be broken into two broad categories, machine-learning-based, and deep-learning based. Each category uses two main methods of learning from the data: supervised and unsupervised learning. Supervised learning methods use a large corpus of data that has labels informing the machine learning model how to classify each input and predict a label for the output. As the machine learns on the data, it uses the label to classify it. When the model sees new data, it can predict the label using what it learned in the training dataset. Common supervised learning methods include regression techniques like linear regression and logistic regression, decision trees, support vector machines, and neural networks.

Supervised learning faces significant limitations especially as it applies to network traffic. It requires labeled training data, which can be expensive and time-consuming to acquire or produce. Additionally, it's vulnerable to class imbalance issues, where one class may significantly outnumber other data. Class imbalance can lead to biased predictions favoring the dominant class. This is especially relevant in network traffic scenarios where normal traffic often vastly outweighs attack instances. Lastly, supervised learning models struggle with "unknown" classes, as they are designed to always predict one of the learned labels, even when no suitable label exists. This can result in misclassifications when encountering novel or unfamiliar data patterns.

Unsupervised learning does not require data that has labels, instead the algorithms learn patterns and structures from the data itself without knowledge of what the output should be. Unsupervised methods include statistical analysis, clustering techniques and anomaly detection. In this paper we examine both supervised and unsupervised learning, we develop adversarial attacks on both, show how they can affect detection, and we provide recommendations for preventing adversarial attacks on network traffic.

2.3. Adversarial AI in Network Traffic

Generating adversarial attacks on network traffic, particularly network flows, differs from those in computer vision. Computer vision adversarial attacks function by perturbing pixels of the image by changing the color values. Pixels in an image have many degrees of freedom that can be perturbed to

create a new adversarial image with the main constraints being the specification of the image format. Network traffic is temporally ordered and highly variable but must conform to prescribed protocol definitions to be valid. This inherent variability, combined with the need to preserve the structure and semantics of network traffic (e.g., preserving packet order, maintaining connection state), introduces a unique set of challenges for generating effective adversarial attacks that can evade detection by network security tools.

Authors Zolbayar et. al focus on generating practical adversarial network traffic flows to evaluate and potentially bypass machine learning-based Network Intrusion Detection Systems (NIDS). The authors developed an attack algorithm called NIDSGAN, based on Generative Adversarial Networks (GANs), demonstrating its ability to successfully evade various NIDS models with high success rates in different threat models—99% in whitebox, 85% in blackbox, and 70% in restricted-blackbox settings. The study highlights the vulnerabilities of these systems to adversarial examples and suggests that current defenses are insufficient, stressing the need for more robust strategies to protect against such attacks. [4]

3. Methodology

In this work we ran several ML algorithms both supervised and unsupervised over a large NetFlow dataset. These algorithms gave us a baseline of accuracy dependent on the algorithm. We then took examples of attacks in the dataset and used several techniques to perturb them with the goal to make the algorithm recognize the attacks as normal (thus evading the algorithm's classification of them as attack). We employed several techniques to assure that these attacks fit within the constraints of NetFlow protocol. We then re-ran these attacks back through the classifier to determine their efficacy in evading the trained classifier.

3.1. Netflow Dataset

For this research we used a dataset from NF-UQ-NIDS-v2 Network Intrusion Detection Dataset [5]. This dataset is a merge of several network based datasets NF-UNSW-NB15-v2, NF-ToN-IoT-v2, NF-BoT-IoT-v2, NF-CSE-CIC-IDS2018-v2. The dataset contains a total of 75 million examples, out of which 25 million (33.1%) are benign/normal flows and 50 million (66.88%) are anomaly/attacks. The attacks are further broken into 20 different attack types show in Table 1: Attack Types. There are 46 columns of which three are labels identifying if the example is an anomaly/attack, the type of attack, and the source dataset. Figure 2: Feature Importance shows each feature and ranks them by the importance to decision function splits in the random forest algorithm.

Table 1: Attack Types

Attack	Number of Examples
DDoS	21748351
DoS	17875585
scanning	3781419
Reconnaissance	2633778
xss	2455020
password	1153323
injection	684897

Bot	143097
Brute Force	123982
Infiltration	116361

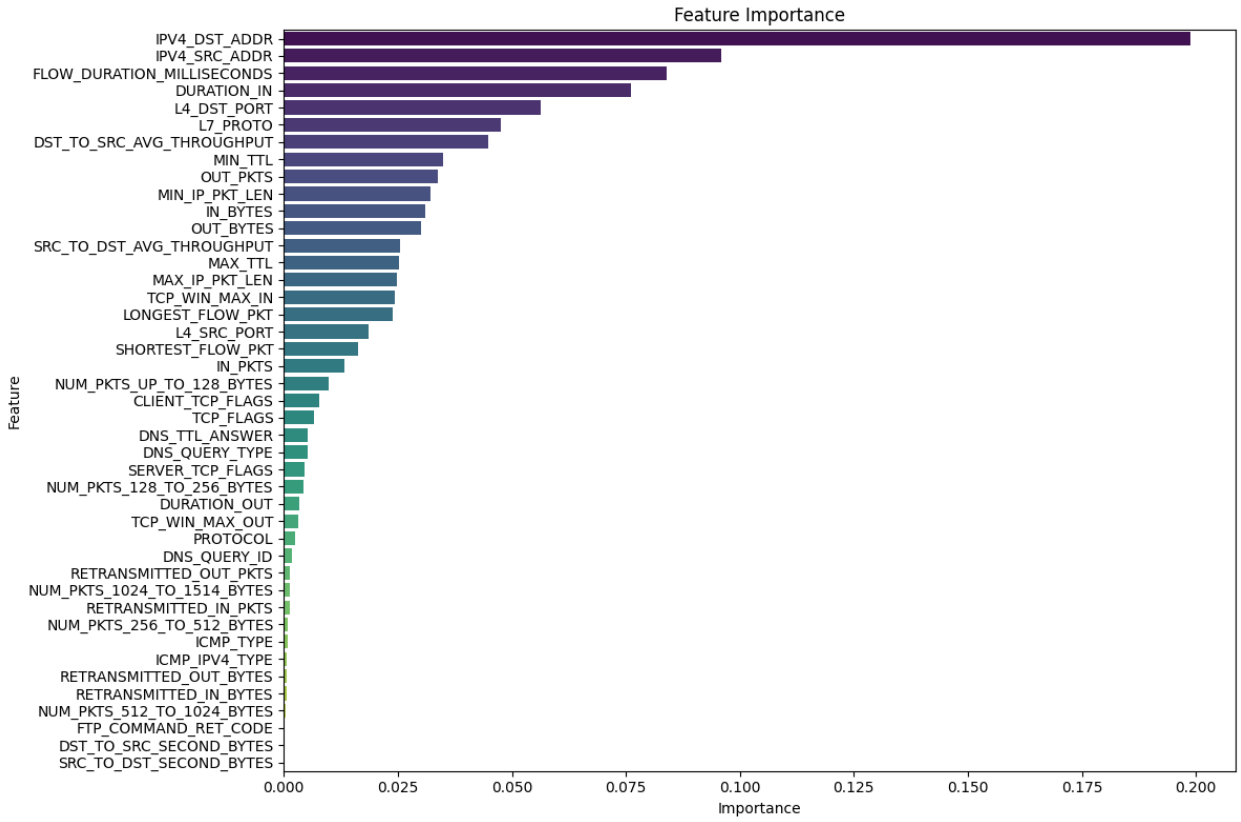

Figure 2: Feature importance using Gini impurity to evaluate quality of decision splits in random forest ensemble.

3.2. AI-based Network Attack Detection

We ran several ML classifiers largely from the Scikit-Learn library over the dataset. These can be largely broken into supervised techniques where the labels of the dataset were used and unsupervised techniques where the structure of the data is learned by the algorithm.

3.2.1. Data Preprocessing

Data preprocessing is a crucial step in preparing data for machine learning models, ensuring the quality and consistency of the data to achieve optimal performance in subsequent analyses. We employed several preprocessing techniques to handle IP addresses, cope with missing and infinite values, and normalize the dataset. To create a balanced dataset, we pulled equal examples from both benign and attack classes. We converted IP addresses to their integer representations to facilitate numerical analysis. The data is also clipped to remain within the representable range of float64 datatypes.

For anomaly detection, the dataset is partitioned into anomalous, benign data sets, with labels and non-informative columns removed. Each data subset undergoes a final standardization through scaling to adjust the data to have consistent ranges and scales, which is critical for algorithms that are sensitive to the magnitude of feature values [6]. The two scalars we utilized were minmax scaling and standard scaling. In minmax scaling, every feature column is independently scaled such that all values fall between the range of 0 and 1. For standard scaling, each feature column is independently scaled such that the mean is 0 and the standard deviation is 1. Since the standard scaling allows values that are beyond the range of 0 to 1, more floating-point precision can be stored for each value. We found that with the minmax scalar, some samples could not be returned to their exact original form after scaling due to rounding. Hence, we utilize the standard scalar in most experiments for better precision and use the minmax scalar for experiments that do not support holding features outside the range of 0 to 1.

3.2.2. Supervised Detection Techniques

Supervised learning technique use the labels in the dataset for both learning and evaluation. In this work we used the binary label of attack (0,1) where the example is labeled benign if zero, otherwise one if an attack. The two supervised techniques we examined were Random Forest and Multi-Layer Perceptron.

3.2.2.1. Random Forest

Random forest is an ensemble learning technique used for classification by constructing multiple decision trees during training and outputting the mode of the classes (for classification) of the individual trees. Random forest typically has high predictive accuracy while controlling for overfitting by averaging the outcomes of the different random trees ensuring that the trees are diverse, robust, and less sensitive to noise in the data compared to a single decision tree [7].

3.2.2.2. Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) [8] is a type of feedforward artificial neural network consisting of multiple layers of artificial neurons (called perceptrons) with each layer fully connected to the next one. These networks typically include an input layer which is fed examples from the data, multiple hidden layers that learn complex patterns, and an output layer that generates the final prediction or classification. Each neuron, in the network generates a weighted sum of its inputs, by processing this sum through an activation function which passes the result to the next layer. To learn and reduce error, MLPs use a technique called back propagation which works by computing the gradient of the loss function with respect to each weight, starting from the output layer and propagating backwards through the hidden layers to the input layer. This gradient is then used to adjust the weights in the direction that reduces the error (gradient decent), in this way the network learns to improve its performance over time.

3.2.3. Unsupervised Detection Techniques

These techniques do not use the labels of the data and instead learn underlying characteristics from the data itself to distinguish between normal and abnormal. The two techniques e used in this paper were isolation forest and one class support vector machine. We trained the algorithms on the normal data and tested it on a random sampling of normal and abnormal examples using the attack label as a ground truth in determining performance metrics.

3.2.3.1. Isolation Forest

Isolation Forest [9] is an unsupervised machine learning algorithm for anomaly detection. Isolation Forest leverages the fact that anomalies are easier to isolate than normal points. By using random partitions to

create trees, it builds a model that encodes the "difficulty" of isolating a point. Anomalies, being rare and different, result in shorter paths in the trees, thereby allowing the model to flag them effectively.

3.2.3.2. One Class Support Vector Machine

Another anomaly detection classifier used in this research was the single or one class support vector machine (OCSVM) [10]. It works by training on data from a single class to capture the core properties and structure of that class. The algorithm attempts to construct a hyperplane in a high-dimensional space that maximizes the separation between the origin and the data points. During prediction, this hyperplane is used to determine whether new data points belong to the same class or are anomalies. Points lying outside the hyperplane's boundary are flagged as anomalies. OCSVM is particularly useful in scenarios where it is challenging to obtain comprehensive data on what constitutes an anomaly, and it is well-suited for applications involving high-dimensional data and complex distributions.

3.3. Techniques for Generating Adversarial Network Traffic

We explored several different methods for generating adversarial examples. These include adversarial synthetic annealing, adversarial genetic algorithm, zeroth-order optimization attack (ZOO), Carlini & Wagner method, and projected gradient decent (PGD). The techniques can be split into two main categories: techniques against non-gradient-based classifiers and against gradient-based ones. Not all attacks are relevant to all algorithms. For example, Carlini & Wagner and PGD require that the classifier utilizes gradients. Furthermore, the library we utilized for the ZOO attack did not contain support for the IF and OCSVM classifiers. Table 1 shows which attacks are relevant to which algorithms.

Table 1: Attack and Algorithm Matrix

	Annealing	Genetic	Zoo	Carlini	PGD
MLP	✓	✓	✓	✓	✓
OCSVM	✓	✓	X	X	X
RF	✓	✓	✓	X	X
IF	✓	✓	X	X	X

3.3.1. Non-gradient Techniques

Many of the anomaly detection algorithms do not use the gradient for the loss function, which is a step that is often exploited in adversarial attacks. To generate adversarial examples against anomaly detection classifiers we employed two different search methods, synthetic annealing and genetic algorithms.

3.3.1.1. Adversarial Synthetic Annealing

Synthetic annealing is a probabilistic optimization technique inspired by the physical process of annealing in metallurgy, where a material is heated and then slowly cooled to remove defects and optimize its structure. In the context of machine learning, this metaphorical heating and cooling process helps in finding a global minimum or an optimal solution in a complex search space. The algorithm begins with an initial solution and iteratively explores neighboring solutions by accepting or rejecting them based on a probability that depends on a "temperature" parameter. Initially, the temperature is set high, allowing the

algorithm to accept worse solutions with higher probability to escape local minima. As the temperature gradually decreases, the algorithm becomes more selective, favoring improvements and eventually converging to an optimal or near-optimal solution.

When applied to generate adversarial examples against anomaly detection algorithms, synthetic annealing exploits the search mechanism to find input perturbations that are subtle yet sufficient to mislead the detection model. The process starts with an input sample labeled as attack and iteratively introduces small, controlled random modifications. During each iteration, the modified input is evaluated based on its ability to evade detection based on the inverse of the decision function of the classifier while attempting to retain key characteristics of the original NetFlow traffic such as valid IP addresses, port numbers etc. The annealing process ensures that the solution does not get trapped in obvious, easily detectable modifications by occasionally accepting suboptimal changes. Over time, the method converges to adversarial examples that appear normal but trigger incorrect classifications from the anomaly detection algorithm.

3.3.1.2. Adversarial Genetic Algorithm

Genetic algorithms (GAs) [11] are a class of optimization algorithms inspired by the principles of natural selection and genetics. They work by evolving a population of candidate solutions over a series of generations to solve complex problems. The process begins with an initial population randomly selected from an existing example. These candidates undergo genetic operations such as selection, crossover, and mutation. Selection chooses the fittest individuals based on a fitness function that evaluates how well they score against the inverse of the decision function of the classifier. Crossover combines pairs of individuals (parents) to create new offspring by randomly selecting points from each parent and concatenating the two parents together, simulating reproduction. Mutation applies random alterations to an individual's genes to introduce variability. Over successive generations, the population gradually evolves toward optimal solutions through these mechanisms of natural selection and genetic variation.

To generate adversarial examples against anomaly detection algorithms using genetic algorithms, our approach works by evolving attack input samples to mislead the model into predicting normal samples. Starting with a population of attack input samples, the GA iteratively applies selection, crossover, and mutation to create modified features. The fitness function in this context is the inverse of the decision function of the trained classifier. The genetic fitness function may penalize changes that make the input easily detectable or stray too far from valid data distributions. Crossover allows combining traits of different successful adversarial examples, while mutation introduces novel alterations. Over time, the GA identifies and refines perturbations that effectively make the model predict that they are normal.

3.3.1.3. Zeroth-Order Optimatization Attack

The Zeroth-order Optimization attack is a black-box method designed to create adversarial examples particularly against deep neural networks, without needing access to the model's internal parameters or architecture [6]. By iteratively querying the model and observing its outputs, the attack approximates the gradients of the loss function with respect to the inputs, typically using numerical methods such as finite differences. These estimated gradients guide the perturbation of inputs to maximize misclassification while keeping changes imperceptible to humans. The goal is to craft adversarial examples efficiently, minimizing queries to avoid detection, and leveraging the transferability property where adversarial inputs effective on one model may also compromise other, unknown models. This highlights the need for robust defenses in machine learning systems to ensure security against such sophisticated attacks.

3.3.2. Gradient-Based Adversarial Techniques

To generate adversarial examples against supervised algorithms the method varies based on the algorithm. For example, projected gradient decent (PGD), and Carlini/Wagner are only useful against algorithms that employ gradient decent in their loss function. In this work MLP is the only algorithm that uses a gradient based loss function.

3.3.2.1. Carlini and Wagner (C&W)

The Carlini & Wagner (C&W) attack [12] is a type of adversarial attack designed to generate adversarial examples that deceive machine learning models, specific to neural networks. C&W initially was used to generate adversarial examples against visual domain models. The C&W attack is particularly notable for its ability to generate adversarial examples that are very subtle in the visual domain, meaning they are often indistinguishable from legitimate examples to the human eye but can still cause a machine learning model to make incorrect predictions.

3.3.2.2. Projected Gradient Decent (PGD)

The Projected Gradient Descent (PGD) adversarial attack [2] is a technique for crafting adversarial examples by refining initial inputs with small, random perturbations and iteratively adjusts these inputs to maximize the model's loss. Each iteration involves computing the gradient of the loss relative to the input, updating the input in the gradient's direction, and projecting the perturbed input back to ensure the perturbation remains within a pre-defined constraint set. This projection maintains the perturbation's magnitude, ensuring it does not excessively alter the original input.

4. Results

This section reviews the results from running the machine learning algorithms trained on the NF-UQ-NIDS-v2 dataset. We call the results from the unperturbed dataset ‘baseline’ results as they establish a baseline on how the algorithm performs without adversarial examples. Additionally, we examine the effectiveness of the adversarial examples in perturbing attack examples, so they are recognized as normal. Finally, we re-run the baseline algorithms with the perturbed examples to evaluate how well overall the attack is functioning.

4.1. Baseline Detection Results

Baseline results were tabulated by taking a random sampling of 100k each of benign and attack samples training on 80% and testing on 20% . We did this a total of ten times and took the average of accuracy, precision, recall and F1-score as can be seen in Table 1. As the table shows the supervised algorithms (MLP, RF) show markedly higher performance with scores above 98% across the metrics, while the unsupervised algorithms (OSVM,IF) scored in the 70 percent in accuracy and upper 80 percent in overall F1-Score. This is consistent with supervised algorithms being more accurate due to the feedback error loop inherent within these methods.

Table 2: Baseline scores of algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
Multi-Layer Perceptron (MLP)	0.982662	0.9838	0.9815	0.9827
One-Class Support Vector Machine (OSVM)	0.7845	0.8094	0.9559	0.8766
Random Forest (RF)	0.99656	0.9969	0.9962	0.9966
Isolation Forest (IF)	0.7943	0.8737	0.868	0.8707

4.2. Adversarial Attacks

We took each attack technique and generated up to 1,000 examples that could successfully evade the algorithm. For each technique we evaluated how efficient the generation of successful perturbed examples was in terms of how many of the perturbed examples were misclassified as normal versus how many continued to be classified as attack/abnormal by the baseline algorithm. Additionally, we evaluated both when the adversarial algorithm could manipulate all features, and when we held the source and destination IP address static, and the adversarial algorithm could only manipulate the remaining features.

Table 3 shows the average of 5 experiments against each type of classifier on how successful the attack was with a value of 1.0 representing a 100% effective attack and all perturbed samples were detected as normal.

Table 3: Attack Success All Features Perturbed

	ZOO	Carlini-Wagner	PGD	Annealing	Genetic
MLP	1.0	0.9994	0.9160	1.000	1.0
Random Forest	0.8462	NA	NA	1.0	1.0
OSVM	NA	NA	NA	0.1984	1.0
Isolation Forest	NA	NA	NA	0.8692	0.4

Table 4 show the average of 5 experiments against each type of classifier when both IP source and IP destination features were set or held static and how successful the attack was again with 1.0 representing an attack where all perturbed examples were recognized as normal by the respective algorithm.

Table 4: Attack Success IP Source, IP Destination Held

	ZOO	Carlini-Wagner	PGD	Annealing	Genetic
MLP	1.0	0.9996	1.0	1.0	1.0
Random Forest	0.9442	NA	NA	1.0	1.0
OSVM	NA	NA	NA	0.2420	1.0
Isolation Forest	NA	NA	NA	0.8626	0.4950

Over all the attacks were successful most of the time in perturbing an attack to make it appear normal to the algorithms. The two notable exceptions to this were the annealing attack against OSVM, and the genetic attack against isolation forest. We also note that for the annealing attack the success rate went up by over 4% when the IP addresses were statically held. This was a surprising result given fewer degrees of perturbation we would expect the success rate to fall. We believe that the strong importance of the IP source and IP destinations address to the classifiers may more heavily weight the results toward normal depending on the addresses chosen. We ran both non-routable private addresses and routable addresses held statically through the annealing attack with a similar result.

4.3. Examples of Output

Table 5 shows an example of the effect the annealing attack has on an anomalous NetFlow record. The original sample corresponds to an XSS attack.

Table 5: Original versus adversarial example for Annealing with MLP

Type	Sample
Original XSS Sample (classified as anomaly)	192.168.1.35,55008,176.28.50.165,80,6,7,2245,7,590,5,27,27,27,0,0,0,64,64,1500, 52,52,1500,2245,590,0,0,0,0,17960000,4720000,9,0,2,0,1,29200,5792,0,0,0,0,0,0
Adversarial Sample (classified as benign)	186.82.43.208,58763,175.185.166.122,1922,8,1,7774,1,18776,28,53,40,21,631890, 110,0,78,54,1663,28,61,1614,3315773,1276177,1981,1,2354,1,14747210,4389822, 196,3,8,0,39,32508,1899,0,6,6756,3,10216,1

Table 6 shows an example of the effect the annealing attack has on an anomalous NetFlow record with the source and destination IP addresses held static. The original sample corresponds to a DoS attack. This result demonstrates that annealing is possible even when holding various features static.

Table 6: Original versus adversarial example for Annealing with MLP, holding IPs

Type	Sample
Original DoS Sample (classified as anomaly)	192.168.0.1,13855,192.168.0.128,80,6,7,140,1,40,1,22,2,20,4294952,0,0,0,0,140,40,40,140,140,40,0,0,0,0,1120000,320000,1,1,0,0,0,512,0,0,0,0,0,0,0
Adversarial Sample (classified as benign)	192.168.0.1,5629,192.168.0.128,2615,4,4,11860,55,0,21,42,20,21,4294966,20,3,0,2,109,51,35,190,78670,673692,443,0,12844,3,2161324,5821631,475,2,0,2,9,4558,9653,855,8,3347,0,1755,3

5. Defense Mechanisms Against Adversarial AI

There is a broad consensus in the research community that defending against adversarial attacks remains a significant challenge. Researchers are actively exploring various defense mechanisms, but no approach has proven to be universally effective yet. Authors [1] describe using adversarial examples in training data to make the model more robust. However, adversarial robustness training has some limitations in that while it provides some protections against known attacks it can be less effective against novel or more sophisticated attack strategies that were not considered during training.

Network traffic, however, has far fewer degrees of freedom for perturbation than even a simple low-resolution image. This makes attacks more difficult to accomplish though not impossible. We recommend a defense in depth approach where machine learning classification of attack traffic is combined with classic firewall architectures.

6. Limitations and Future Work

We will conduct a more in-depth analysis of the very large dataset to look at the distributions of various features. For example, we do not have a clear picture of how many examples have public IP addresses versus private and are the most common ports, protocols etc. We started with a balanced dataset based on the labels, but there is most certainly some latent majority classes that should be uncovered.

We realize that holding just the source and destination IP address is insufficient for an effect example attack. In future work, we will analyze how well adversarial examples can be generated as we gradually increase the number of static features. Additionally, we will put in place additional checks to ensure that adversarial examples conform to IANA protocol and port combinations with specific focus on commonly used non-deprecated protocols. Finally, we will run our adversarial attacks against off-the-shelf NIDS (Network Intrusion Detection Systems) and evaluate their effectiveness.

7. Conclusion

The increasing reliance on AI and ML for network traffic analysis in cybersecurity can introduce additional risks and security concerns through adversarial attacks. This research demonstrated various machine learning classifiers, both supervised and unsupervised, which can be instrumental in advanced network attack detection. However, it also showed how adversarial learning could be leveraged to generate adversarial examples that deceive these classifiers. Adversaries could employ techniques like these to orchestrate attacks that manage to bypass detection systems, highlighting a potential significant vulnerability.

Abbreviations

AI	artificial intelligence
ML	machine learning
IF	isolation forest
RF	random forest
MLP	multi-layer perceptron
NIDS	network intrusion detection system
OSVM	one class support vector machine
PGD	projected gradient decent
ZOO	zeroth-order optimalization
XSS	cross site scripting
DoS	denial of service

Bibliography

[1] J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," Axiv, 2014.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks," 2017.

[3] T. Sowmya and M. A. E.A, "A comprehensive review of AI based intrusion detection system," Measurement: Sensors, 2023.

[4] B.-E. Zolbayar, R. Sheatsley, P. McDaniel, M. Weisman, S. Zhu, S. Zhu and S. Krishnamurthy, "enerating Practical Adversarial Network Traffic Flows Using NIDSGAN," 2022.

[5] M. Sarhan, S. Layeghy, N. Moustafa and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," in Big Data Technologies and Applications, Springer International Publishing, 2021, p. 117–135.

[6] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi and H. Cho-Jui, "Chen, Pin-Yu, et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models.," in Proceedings of the 10th ACM workshop on artificial intelligence and security, 2017.

[7] L. Breiman, " Random Forests," Machine Learning, pp. 5-32, 2001.

[8] I. Goodfellow, B. Yoshua and A. Courville, Deep Learning, MIT Press, 2016.

[9] F. Lui, K. Ting and Z.-H. Zhou, "Isolation Forest," in 2008 Eighth IEEE International Conference on Data Mining, 2008.

[10] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor and J. Patt, "Support Vector Method for Novelty Detection," Advances in neural information processing systems 12, 1999.

[11] J. Holland, "Genetic Algorithms," Scientific American, pp. 66-73, July 1992.

[12] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," 2017.