Evaluation and Detection of Adversarial Attacks in ML-based NIDS

Huda Ali Alatwi

Supervisor: Dr. Charles Morisset

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

Secure and Resilient Systems Group September 12, 2024

I dedicate this thesis to my dear son Faris...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Huda Ali Alatwi
September 12, 2024

Acknowledgements

I thank Allah for guiding me and providing the strength to successfully complete my PhD research. I would like to express my deepest gratitude to my supervisor, Dr. Charles Morriset, for his invaluable guidance, unwavering support, and insightful feedback throughout the course of my research. I extend my deepest appreciation to my family and friends for their steadfast encouragement and support. To my dear son Faris, your laughter and joy have been my greatest inspiration, brightening every step of this journey. I also sincerely thank my home country Saudi Arabia and Tabuk University for providing me with a full scholarship to pursue my doctoral studies.

Abstract

A Network Intrusion Detection System (NIDS) monitors network traffic to detect unauthorized access and potential security breaches. A Machine Learning (ML)-based NIDS is a security mechanism that uses ML algorithms to automatically detect and identify suspicious activities or potential threats in a network by analyzing traffic patterns, distinguishing between normal and malicious behaviors, and alerting or blocking unauthorized access. Despite high accuracy, ML-based NIDS are vulnerable to adversarial attacks, where attackers modify malicious traffic to evade detection and transfer these tactics across various systems. To the best of our knowledge, several crucial research gaps persist in this area that have not yet been addressed. First, there are no systematic threat models for identifying and analyzing potential threats and vulnerabilities in ML-based NIDS. This lack of structured threat modeling hinders the development of comprehensive defense strategies and leave these systems vulnerable to adversarial attacks that exploit unknown weaknesses in the ML algorithms or system architecture. The current literature employs generic adversarial attacks mainly designed for image recognition domain to assess the resilience of ML-based, but no research has verified the realism and compliance of these attacks with network domain constraints. Investigating whether these attacks produce valid network is crucial to determine their real-world threat level and the suitability of ML-based NIDS for deployment. Another gap in the literature is the lack of comprehensive evaluations that include a wide range of models, attack types, and defense strategies using contemporary network traffic data. This gap makes it difficult to verify the generalizability and applicability of the findings for real-world. The absence of standardized metrics further hampers the ability to evaluate and compare the resilience of ML-based NIDS to adversarial attacks. Finally, there is no a lightweight solution that effectively detects and classifies adversarial traffic with scoring high accuracy on both clean and perturbed data with proven efficiency over recent dataset and across various attack types and defenses.

These gaps hinder the robustness of ML-based NIDS against adversarial attacks. Therefore, this Ph.D. thesis aims to address these vulnerabilities to enhance the ML-based NIDS resilience.

The overall contributions include; 1) A threat modeling for ML-based NIDS using STRIDE and Attack Tree methodologies; 2) An investigation of the realism and performance of generic adversarial attacks against DL-based NIDS; 3) A comprehensive evaluation for adversarial attacks' performance consistency, models' resilience, and defenses' effectiveness; 4) Adversarial-Resilient NIDS, a framework for detecting and classifying adversarial attacks against ML-based NIDS.

Table of Contents

List of Figures x

List of Tables xi

List of Abbreviations xii

1 Introduction 1

1.1 Research Problem . . . 3
1.2 Research Aim . . . 4
1.3 Research Questions . . . 4
1.4 Contributions . . . 8
1.5 Thesis Structure . . . 8
1.6 Publications . . . 9

2 Background 11

2.1 Summary . . . 11
2.2 Network Intrusion Detection Fundamentals . . . 11
2.3 Machine Learning Fundamentals . . . 12
2.3.1 Machine Learning Approaches . . . 12
2.3.2 Machine Learning Tasks . . . 12
2.3.3 Machine Learning Depth . . . 13
2.4 Machine Learning Pipeline . . . 13
2.4.1 Problem Definition . . . 14
2.4.2 Data Collection . . . 14
2.4.3 Data Preprocessing . . . 14
2.4.4 Model Selection . . . 15
2.4.5 Model Training . . . 15
2.4.6 Model Evaluation . . . 15
2.4.7 Model Deployment . . . 16
2.5 Deep Learning Fundamentals . . . 16
2.6 Adversarial Machine Learning Fundamentals . . . 18
2.6.1 Adversarial Threat Model . . . 19
2.6.2 Adversarial Attacks . . . 22
2.7 Conclusion . . . 22

3 Threat Modeling for ML-based NIDS **24**
3.1 Summary . . . 24
3.2 Introduction . . . 24
3.3 Related Work . . . 25
3.4 Background . . . 26
3.4.1 Threat Modeling . . . 26
3.4.2 STRIDE Model . . . 26
3.4.3 Attack Tree Model . . . 27
3.5 Threat Modeling for ML-Based NIDS . . . 27
3.5.1 ML-based NIDS Components . . . 28
3.5.2 Adversary Model . . . 29
3.5.3 Attack Surface . . . 29
3.5.4 Data Flow Diagram . . . 30
3.5.5 Threat Assessment . . . 30
3.6 Attack Tree Model For ML-Based NIDS . . . 31
3.7 STRIDE Model For ML-Based NIDS . . . 35
3.8 Discussion . . . 37
3.9 Conclusion . . . 38

4 Adversarial Machine Learning in NIDS Domain: A Systematic Review **39**
4.1 Summary . . . 39
4.2 Introduction . . . 39
4.3 Related Work . . . 40
4.4 Methodology . . . 41
4.4.1 Research Questions . . . 41
4.4.2 Search Strategy . . . 42
4.4.3 Inclusion and Exclusion Criteria . . . 42
4.4.4 Data Extraction . . . 42
4.5 Generating Adversarial Attacks for ML-based NIDS . . . 43
4.5.1 Reinforcement Learning Attacks . . . 43
4.5.2 Generative Adversarial Networks Attacks . . . 43
4.5.3 Surrogate Model Attacks . . . 44
4.5.4 Genetic Algorithms Attacks . . . 45
4.5.5 Constrained Generic Attacks . . . 45
4.5.6 Certain Features Manipulation Attacks . . . 46
4.5.7 Other Approaches . . . 46
4.6 Evaluating ML-based NIDS Resilience to Adversarial Attacks . . . 49
4.7 Defending ML-based NIDS Against Adversarial Attacks . . . 52
4.7.1 Adversarial Training . . . 53
4.7.2 Ensemble Learning . . . 54
4.7.3 Feature Reduction . . . 54

4.7.4 Hybrid Approaches . . . 55
4.7.5 Other Approaches . . . 55
4.8 Discussion . . . 59
4.8.1 Findings Analysis . . . 59
4.8.2 Research Questions . . . 63
4.9 Conclusion . . . 66

5 Realism vs. Performance for Adversarial Examples Against DL-based NIDS 67
5.1 Summary . . . 67
5.2 Introduction . . . 67
5.3 Network Traffic Constraints . . . 68
5.4 Literature Drawbacks . . . 69
5.5 Experimental Setup . . . 70
5.5.1 Datasets . . . 70
5.5.2 Dataset Preprocessing . . . 71
5.5.3 Adversarial Attacks & Target Model Implementation . . . 72
5.5.4 Evaluation Metrics . . . 72
5.6 Experimental Results & Analysis . . . 73
5.6.1 Attacks Performance . . . 74
5.6.2 Attacks Unrealism . . . 76
5.7 Discussion . . . 77
5.7.1 Attacks Unrealism . . . 78
5.7.2 Attacks Infeasibility . . . 79
5.8 Conclusion . . . 79

6 Resilience Evaluation and Detection of Adversarial Attacks in ML-based NIDS 81
6.1 Summary . . . 81
6.2 Introduction . . . 81
6.3 Literature Drawbacks . . . 83
6.4 Resilience Index . . . 84
6.5 Adversarial-Resilient Network Intrusion Detection System (AR-NIDS) . . . 86
6.6 Experimental Setup . . . 87
6.6.1 Adversary Model . . . 87
6.6.2 Dataset . . . 87
6.6.3 Dataset Preprocessing . . . 88
6.6.4 Adversarial Attacks & Models Implementation . . . 88
6.6.5 Evaluation Metrics . . . 90
6.7 Resilience Evaluation of ML-based NIDS to Adversarial Attacks . . . 91
6.7.1 Models Performance Over Clean Test Data . . . 91
6.7.2 Models Performance: Clean vs. Adversarial . . . 92

6.7.3 Models Adversarial Resilience . . . 93
6.7.4 Adversarial Attacks Performance . . . 93
6.8 Evaluation of Adversarial-Resilient NIDS Framework . . . 95
6.8.1 Baseline Ensemble (BE) . . . 95
6.8.2 Adversarially Trained Models (ATM) . . . 96
6.8.3 Adversarially Trained Ensemble (ATE) . . . 97
6.8.4 Performance of Adversarial Attacks Classifier . . . 97
6.9 Impact of Defense Strategies on Adversarial Attacks Efficacy . . . 98
6.10 Discussion . . . 99
6.11 Conclusion . . . 100

7 Conclusion **101**
7.1 Summary . . . 101
7.2 Discussion . . . 101
7.2.1 Threat Modeling Analysis for ML-Based NIDS: Uncovering Hidden Risks . . . 101
7.2.2 Adversarial Evasion Attacks on ML-based NIDS: Reviewing Current Knowledge 102
7.2.3 Evaluating the Realism of Adversarial Attacks: Bridging Theory and Reality . 102
7.2.4 Enhancing ML-Based NIDS Resilience: Shielding the Shield . . . 103
7.3 Future Work . . . 104
7.4 Challenges and Research Opportunities . . . 104
7.5 Conclusion . . . 105

Appendix A Experiments Reproducibility **106**
A.1 Introduction . . . 106
A.2 Computer Configuration . . . 106
A.3 Packages . . . 106
A.4 Experiment 1: Realism vs. Performance for AEs Against DL-based NIDS . . . 107
A.4.1 Datasets . . . 107
A.4.2 Model Parameters . . . 107
A.4.3 Code Repository . . . 107
A.5 Experiment 2: Evaluating and Detecting AEs in ML-based NIDS . . . 108
A.5.1 Datasets . . . 108
A.5.2 Model Parameters . . . 108
A.5.3 Code Repository . . . 108

Appendix B Datasets Description **109**
B.1 WSN-DS Dataset . . . 109
B.2 BoT-IoT Dataset . . . 109
B.3 NF-UQ-NIDS Dataset . . . 110

References **111**

List of Figures

2.1 Machine Learning Pipeline . . . 13
2.2 Neuron Structure in DNN . . . 17
2.3 DNN Structure . . . 17
2.4 Threat Model of Adversarial Attacks . . . 19

3.1 Threat Modeling Scheme . . . 28
3.2 Components of ML-based NIDS . . . 29
3.3 DFD of ML-based NIDS . . . 30
3.4 ML-based NIDS Attack Tree Threat Model . . . 35

4.1 Distribution of Studies per Category . . . 59
4.2 Distribution of Studies per Environment . . . 60
4.3 Distribution of Studies per Dataset . . . 60
4.4 Distribution of Studies per Setting . . . 60
4.5 Distribution of Studies per Defense Mechanism . . . 61
4.6 Most Utilized Adversarial Attacks . . . 62

5.1 Evasion Rate vs. Unrealism Index over WSN-DS . . . 75
5.2 Evasion Rate vs. Unrealism Index over BoT-IoT . . . 75
5.3 White-box Attack Unrealism Metrics Percentages . . . 76
5.4 Black-box Attack Unrealism Metrics Percentages . . . 76

6.1 Adversarial-Resilient NIDS . . . 87
6.2 ML-NIDS Resilience to Adversarial Attacks . . . 90
6.3 Baseline Models Resilience Index . . . 93
6.4 Attack Success Rates Against Baseline Models . . . 94
6.5 Attack Success Rates Against Defense Models . . . 95
6.6 Defense Models Resilience Index . . . 95
6.7 Comparison Baseline Models vs. Defense Models . . . 96
6.8 Impact of Defense Strategies on Adversarial Attacks Efficacy . . . 98

List of Tables

3.1 Threats Assessment for ML-based NIDS Attack Tree Model 34
3.2 Threats Assessment for ML-based NIDS Using STRIDE Model 36
3.3 Countermeasures for ML-based NIDS STRIDE Model Threats 37

4.1 Literature Classification Characteristics 42
4.2 Generating Adversarial Attacks for NIDS Studies 48
4.3 Generating Adversarial Attacks for NIDS Studies 49
4.4 Evaluating NIDS Resilience to Adversarial Attacks Studies 52
4.5 Defending NIDS Against Adversarial Attacks Studies 57
4.6 Defending NIDS Against Adversarial Attacks Studies 58
4.7 Overview of Used NIDS Datasets in the Literature 59

5.1 Comparison with Related Works 70
5.2 Used Features in the Datasets 71
5.3 Feed-Forward DNN Model Parameters 72
5.4 Attacks Assessment on WSN-DS & BoT-IoT Datasets 75
5.5 Attacks Unrealism Metrics on WSN-DS & BoT-IoT Datasets 78

6.1 Employed Attacks and Defense Comparison [$\odot$=Attacks, $\oplus$=Defense] 84
6.2 Statistics of Used NF-UQ-NIDS Dataset 88
6.3 Samples Distribution in Train and Test Data 88
6.4 Parameters of ML-based NIDS Models 89
6.5 Performance Metrics for Binary Classification 90
6.6 Baseline Models Performance over Clean vs Adversarial Data 92
6.7 Adversarial Attacks Classifier Performance 97

A.1 Feed-Forward DNN Model Parameters 107
A.2 Parameters of ML-based NIDS Models 108

B.1 Description of WSN-DS Dataset 109
B.2 Description of BoT-IoT Dataset 109
B.3 Description of NF-UQ-NIDS-v2 Dataset 110

List of Abbreviations

A2PM	Adaptive Perturbation Method
AE	Adversarial Example
AIDAE	Anti-Intrusion Detection AutoEncoder
AIDTF	Adversarial Intrusion Detection Training Framework
AR-NIDS	Adversarial-Resilient Network Intrusion Detection System
ART	Adversarial Robustness Toolbox
ASR	Attack Success Rate
AT	Adversarial Training
ATE	Adversarially Trained Ensemble
ATM	Adversarially Trained Models
Adaboost	Adaptive Boosting
BA	Balanced Accuracy
BB	Black Box
BFAM	Bruteforce Blackbox Method
BIM	Basic Iterative Method
BiGAN	Bidirectional Generative Adversarial Network
CIFGSM	Constraint Iteration Fast Gradient Sign Method
CNN	Convolutional Neural Network
CPGD	Constrained Projected Gradient Descent
CSV	Comma Separated Values
CW	Carlini and Wagner
Catboost	Categorical Boosting
DAE	Denoising Autoencoder
DAGMM	Deep Autoencoding Gaussian Mixture Model
DBN-LSTM	Deep Belief Network Long Short Term Memory
DDoS	Distributed Denial of Service
DFD	Data Flow Diagram
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
DoS	Denial of Service
EL	Ensemble Learning
EN	Elastic Net
ER	Evasion Rate
ET	Extra Trees
FCDNN	Fully Connected Deep Neural Network
FENCE	Feasible Evasion Attacks on Neural Networks in Constrained Environments
FFDNN	FeedForward Deep Neural Network
FGDM	Feature Grouping and Multimodel Fusion Detector
FGSM	Fast Gradient Sign Method
GA	Genetic Algorithms
GAN	Generative Adversarial Network
GAN-AT	Generative Adversarial Network Adversarial Training
GB	Gradient Boosting
GBDT	Gradient Boosting Decision Tree
GenAAL	Generative Adversarial Active Learning
HSJ	HopSkipJump
IDS	Intrusion Detection System
IF	Isolation Forest

IFGSM	Iterative Fast Gradient Sign Method
IoT	Internet of Things
JSMA	Jacobian Saliency Map
KNN	K-Nearest Neighbors
LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LDA	Linear Discriminant Analysis
LGBM	Light Gradient Boosting Machine
LR	Logistic Regression
LSTM	Long Short Term Memory
MANDA	MANifold and Decision boundary-based
MFA	Multi-Factor Authentication
MIFGSM	Momentum Iterative Fast Gradient Sign Method
ML	Machine Learning
MLP	Multi-Layer Perceptron
MoEvA	Multi-Objective Evolutionary Adversarial Attack
NB	Naive Bayes
NIDS	Network Intrusion Detection System
NIDSFM	Network Intrusion Detection System Flow Merge
PCA	Principal Component Analysis
PF	Perturbed Features
PGD	Projected Gradient Descent
PSO	Particle Swarm Optimization
QDA	Quadratic Discriminant Analysis
QoS	Quality of Service
RBM	Restricted Boltzmann Machine
REDNN	Robust, Effective, and Resource Efficient Fully Connected Neural Network
RF	Random Forest
RFE	Recursive Feature Elimination
RI	Resilience Index
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RONI	Reject on Negative Impact
RePO	Reconstruction from Partial Observation
SAE	Stacked Autoencoder
SE	Stacking Ensemble
SGD	Stochastic Gradient Descent
SIGSM	Selective and Iterative Gradient Sign Method
SNN	Self-Normalizing Neural Network
SPSA	Simultaneous Perturbation Stochastic Approximation
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
SVM	Support Vector Machine
SeqGAN	Sequence Generative Adversarial Network
TANTRA	Timing-Based Adversarial Network Traffic Reshaping Attack
UASG	Universal Adversarial Sample Generator
UI	Unrealism Index
VAE	Void Adversarial Example
WB	White Box
WGAN	Wasserstein Generative Adversarial Network
XAI	Explainable Artificial Intelligence
XGB	eXtreme Gradient Boosting
ZOO	Zeroth Order Optimization

Chapter 1. Introduction

As the first line of defense, the network intrusion detection system (NIDS) carries the responsibility of protecting and securing the network against the attacks generated by malicious traffic. A NIDS is a network security detective control that monitors network traffic in order to recognize any malicious and anomalous activities that can be part of an attack [29]. It secures the network resources against cyberattacks, destruction, and unauthorized access or modification; hence it ensures resources' availability, confidentiality, and integrity [29]. It can be broadly classified into signature-based and anomaly-based systems [100]. Signature-based NIDS use predefined patterns to detect known threats, excelling at identifying these but failing with new threats. Anomaly-based NIDS detect deviations from established normal behavior, which helps in identifying novel attacks but can lead to more false positives. In essence, signature-based NIDS are precise for known threats, while anomaly-based NIDS offer wider coverage for unknown threats [100].

Machine learning techniques are integrated into NIDS due to their ability to adapt to evolving cyber threats, handle large volumes of data, and provide high accuracy rates. By automatically extracting relevant features and continuously improving detection through re-training, machine learning enhances NIDS effectiveness against diverse attacks [100]. Despite these benefits, machine learning-based security solutions face serious issues, notably adversarial attacks [70]. These attacks involve carefully crafted inputs that cause models to make incorrect classifications which raises concerns about the reliability of ML-based NIDS [151].

These attacks exploit vulnerabilities in the underlying machine learning algorithms of the NIDS. Poisoning attacks inject altered data into the training set to manipulate ML-based NIDS behavior. This compromises the model's integrity, hinders its ability to accurately distinguish between normal and malicious traffic, and leads to degraded performance. [34, 135]. Oracle attacks leverage the model's responses to crafted queries to extract sensitive information, compromising the confidentiality and integrity of the NIDS and facilitating the development of advanced evasion techniques [34, 135]. Evasion attacks manipulate the malicious traffic to evade detection by exploiting vulnerabilities in the model's decision boundary [34, 135]. These attacks can bypass security measures and allow undetected malicious activities to harm the network.

Failing to recognize and address the vulnerabilities that can be exploited by these attacks renders ML-based NIDS highly susceptible to them. This susceptibility can lead to potential breaches and compromised network security which undermines the core function of these systems. Without identifying and addressing these vulnerabilities, ML-based NIDS systems cannot anticipate and counter the adversarial attacks which mislead the detection system, degrade its

performance, extract confidential information, or trigger hidden malicious activities [135]. This allows malicious actors to bypass defenses, infiltrate networks, and potentially cause significant security breaches.

Systematic threat modeling emerges as a vital solution to this problem. It offers a structured approach to identify, quantify, and prioritize security risks [145]. By systematically modeling threats, we can pinpoint the vulnerabilities inherent in ML algorithms and the others posed by inadequate security measures, allowing for proactive threat mitigation before harm occurs. Moreover, threat modeling enhances the security and efficacy of ML-based NIDS by effectively quantifying and prioritizing threats [145]. This process guides the development of targeted countermeasures, addresses immediate vulnerabilities, and supports the continuous improvement of detection algorithms and system architecture. Consequently, NIDS can effectively detect malicious activities and maintain resilience against evolving threats.

Many studies in the literature have demonstrated that generic adversarial attacks can effectively trick DL-based NIDS and evade detection [42, 45, 52, 69, 72, 75, 89, 92, 117, 123, 125, 169, 170, 175, 149]. However, the degree to which these manipulations accurately represent real network behavior was not explored. These approaches were designed for unconstrained domains such as image recognition, where features can be independently altered without restrictions [14]. In contrast, NIDS traffic data must follow domain-specific constraints, including interdependent and fixed or limited feature values [14]. The absence of a metric for quantitatively measuring the realism of outputs from generic adversarial attacks poses a significant challenge in the objective evaluation and comparison of how closely they resemble actual network traffic. Without such a metric, it is difficult to assess the true efficacy and threat level of adversarial examples in real-world scenarios. This gap hinders the ability to determine the practical impact of these attacks on ML-based NIDS and their viability for real-world deployment.

Developing this metric would bridge this critical gap, enabling the evaluation of whether adversarial traffic adheres to the constraints and patterns of legitimate network traffic. This would enhance our understanding of the real threat posed by adversarial attacks and inform the applicability of ML-based NIDS for real-world deployments.

In the context of ML-based NIDS, the concept of resilience has garnered increasing attention in the literature [166, 140]. Resilience refers to a model's ability to maintain its effectiveness in the face of adversarial attacks or other perturbations, a characteristic that is essential for ensuring robust and reliable performance over time. Several studies have highlighted the importance of resilience, noting that traditional metrics like accuracy or precision may not fully capture a model's ability to resist or recover from adversarial manipulation [166, 140]. Given the dynamic and hostile environments in which NIDS operate, ensuring resilience is crucial. NIDS must reliably detect and respond to various attacks without frequent retraining or manual intervention. This highlights the need for a standardized metric that not only evaluates traditional performance but also measures a model's consistency and reliability under adversarial conditions, providing a comprehensive assessment of its resilience.

Significant research has been conducted on evaluating and defending against adversarial attacks in ML-based NIDS [133, 46, 132, 92, 45, 52, 125, 140, 1, 178]. Despite these advancements, much of the literature focuses on isolated aspects of the problem, often evaluating specific attacks or defenses in a limited scope. A comprehensive assessment that account for the consistent performance of various attacks, models resilience, and the effectiveness of defenses are still lacking. This issue is exacerbated by the reliance on outdated datasets, which diminishes the relevance of findings to current real-world network traffic [134]. Furthermore, the absence of a standardized metric to measure resilience hampers the objective assessment of ML-based NIDS resilience to adversarial attacks. Therefore, a comprehensive evaluation encompassing a wide range of attacks, models, and defenses, using recent datasets and standardized metrics, is crucial. It ensures real-world applicability, identifies strengths and weaknesses, establishes benchmarks, and provides reliable and credible findings.

Researches on enhancing the resilience of ML-NIDS against adversarial attacks have explored various strategies, including adversarial training [132, 92, 52, 125, 140, 1, 178], ensemble learning [133, 46], and the development of specialized adversarial detectors [45]. These efforts have demonstrated some improvements in the ability of ML-based NIDS to withstand adversarial perturbations. However, the necessity for ongoing research to enhance these solutions becomes apparent when considering the trade-offs associated with current methods. Increased computational overhead can limit the scalability of NIDS, while reduced accuracy on clean data can compromise the overall effectiveness of the system in identifying legitimate threats. Higher complexity and latency can also impede the real-time detection capabilities essential for mitigating cyber threats promptly. Moreover, the limited generalization capabilities of many existing approaches mean that they may not perform well against a wide range of adversarial attacks, reducing their practical applicability in dynamic and evolving network environments.

Therefore, it is essential to develop a solution that can effectively detect and classify adversarial traffic, while maintaining resilience to perturbed data and ensuring high performance on clean data. Additionally, the solution must demonstrate generalized efficiency through comprehensive evaluations across a variety of attack types, comparison to the sate-of-art defense mechanisms, and over recent datasets. This rigorous approach will validate its robustness and practical applicability.

1.1. Research Problem

Based on the reasoning on the previous section, we aim to address the following research gaps:

- Absence of threat models for ML-based NIDS, which systematically identify and analyze potential threats and vulnerabilities, puts network infrastructures at risk if these threats are not identified and proactively mitigated.

- Lack of investigation into the alignment of outputs from generic adversarial attacks with actual network traffic constraints. Assessing whether these attacks comply with network

traffic constraints is crucial to determine if they pose real threats and potentially render the ML-based NIDS unsuitable for real-world deployment.

- Insufficient comprehensive evaluations that encompass various models, defense approaches, and different types of attacks while utilizing recent and representative real-world network traffic data. Additionally, the absence of a standardized metric for consistent assessments and comparisons of ML-based NIDS resilience further complicates this issue. Without such rigorous evaluations and standardized metrics, the generalizability and applicability of the findings to real-world scenarios are difficult to validate.

- Need for a solution that can detect and classify adversarial traffic, remain resilient against attacks and maintain high performance on clean data, proving its robustness and practical applicability through comprehensive evaluations against various attacks and state-of-the-art defenses.

1.2. Research Aim

This thesis aims to enhance the resilience of ML-based NIDS against adversarial attacks in order to improve their reliability and security for real-world deployments, while ensuring they remain highly effective and efficient.

1.3. Research Questions

This thesis examines the following research questions:

RQ1. What are the potential threats and vulnerabilities associated with employing machine learning approaches in NIDS, and what effective countermeasures can be devised to enhance their security and trustworthiness? (Chapter 3)

We identified and categorized 22 distinct threats specifically associated with machine learning models in NIDS. These threats are thoroughly documented in Section 3.6. Utilizing attack tree threat modeling approach, we systematically organized these threats based on their occurrence, whether before or after model deployment, as depicted in Figure 3.4. Additionally, we conducted a detailed risk assessment, summarized in Table 3.1, to evaluate the potential impact and likelihood of each identified threat. Our risks assessment provides a clear picture of the ML-based NIDS's vulnerabilities and helps in prioritizing threats that require immediate attention.

Moreover, we advised a series of robust countermeasures, detailed in Table 3.1, encompassing both technical and procedural strategies. Implementing these countermeasures can significantly enhance the security and trustworthiness of ML-based NIDS and ensure their resilience against the identified threats.

To provide holistic threat modeling, we also applied the STRIDE method to systems' data flow to spot other threats that imposed by inadequate security measures, detailed in Table 3.2. In total, 46 threats were identified using the Attack Tree and STRIDE approaches, necessitating

mitigation for deploying a resilient ML-based NIDS. Our comprehensive threat modeling and risk assessment serve as a critical foundation for enhancing the resilience and reliability of these systems. Despite the dynamic nature of adversarial attacks and the possibility of new threats emerging, the structured models we presented ensures that ML-based NIDS can be effectively adapted and fortified against a wide range of vulnerabilities.

RQ2. What adversarial evasion attacks are employed for ML-based NIDS, what mitigation strategies exist, and what are the limitations and research improvement considerations? (Chapter 4)

We conducted a systematic review of evasion adversarial attacks against ML-based NIDS and identified various techniques used to generate tailored adversarial attacks. These include reinforcement learning, GANs, surrogate models, genetic algorithms, constrained generic attacks, and specific feature manipulations, as comprehensively documented in Section 4.5. We also identified the employed mitigation strategies to counter these attacks in Section 4.7, which include adversarial training, ensemble learning, feature reduction, and hybrids of these approaches.

However, we identified several limitations, as presented in Section 4.8.2. The proposed adversarial attack generation methods often assume detailed adversary knowledge of the target NIDS, which may not be realistic. Techniques like RL-based frameworks, GANs, and genetic algorithms can be computationally expensive and time-consuming, limiting their real-time applicability. GANs and surrogate models require large, high-quality datasets, and their generated adversarial examples may lack transferability across different models. Additionally, some techniques are restricted to specific attacks or require labeled data, which is not always practical. The employed defense mechanisms have drawbacks: adversarial training leaves models vulnerable to new attacks, feature reduction can reduce discriminative power, and ensemble learning has high computational overhead.

In Section 4.8.2, we addressed the limitations of current generation methods by suggesting the design of attacks in a black-box setup, considering real-world factors like network dynamics and traffic volumes, and ensuring resource efficiency for real-time deployment. For defenses, we recommended strategies that generalize well across various attack types, efficiently use resources, and maintain high detection accuracy without significant computational overhead or latency. Additionally, comprehensive evaluation using standardized datasets and metrics is crucial for ensuring their effectiveness and reliability in real-world scenarios.

Our findings highlight the need for ongoing research to refine these techniques, ensuring they are both practical and effective in dynamic, real-world settings. Strengthening the resilience of NIDS against adversarial attacks requires a balanced approach that considers both the robustness of detection mechanisms and the operational feasibility of the proposed solutions.

RQ3. To what extent do the outputs from generic adversarial attacks align with the characteristics of real network traffic, and how does evaluating the realism and

feasibility of these attacks contribute to assessing the suitability of ML-based NIDS for real-world deployments? (Chapter 5)

We found that the outputs from generic adversarial attacks often do not align well with the characteristics of real network traffic due to several unrealistic modifications, as detailed in Sections 5.6.2 and 5.7. Specifically, many white-box attacks, such as BIM, PGD, and CW, manipulate an extensive number of features, often exceeding 85%. This level of perturbation is unrealistic in practical scenarios as adversaries typically cannot access or control such a vast number of features in real network environments. Furthermore, these attacks introduce out-of-range values, which are values outside the expected range for certain features, making the traffic easily detectable by traditional systems and thus less effective in evading detection.

Additionally, many adversarial attacks alter binary features by introducing non-binary values, which are nonsensical in a real-world context where features should only be 0 or 1. For example, features that denote a true or false condition in the network traffic are assigned decimal or negative values, which break the semantic integrity of the traffic. Similarly, categorical features are often perturbed to trigger multiple categories simultaneously, which is unrealistic because a categorical feature should only belong to one category at a time.

These unrealistic modifications underscore the infeasibility of such adversarial attacks in real-world scenarios, as they disrupt the natural structure and behavior of network traffic. This finding implies that many current adversarial attack methods may not provide a realistic assessment of the robustness of ML-based NIDS when deployed in real-world environments. We need to utilize more sophisticated attack techniques that maintain the realism of network traffic to better evaluate and improve the resilience of ML-based NIDS.

RQ4. What metric can be used to effectively evaluate and rank the resilience of ML-based NIDS for deployment? (Chapter 6)

We introduced the Resilience Index (RI) as a comprehensive metric for evaluating and ranking the resilience of ML-based NIDS models against adversarial attacks, as described in Section 6.7.3. RI combines balanced accuracy (BA_{adv}), macro F1 score ($F1_{adv}$), and Attack Success Rate (ASR). BA_{adv} and $F1_{adv}$ provide insight into detection accuracy, while ASR measures vulnerability to evasion. This unified metric balances detection accuracy with adversarial resilience, enabling effective selection of robust NIDS models for deployment.

However, the Resilience Index is not without limitations. Its effectiveness depends on the quality and diversity of the adversarial examples used for evaluation. If the adversarial examples are not representative of real-world scenarios, the RI might overestimate a model's resilience. Additionally, while RI combines multiple metrics, it may still not capture all dimensions of resilience, such as computational efficiency and real-time performance.

Our findings suggest that the Resilience Index can significantly improve the evaluation process for ML-based NIDS, facilitating the comparison and selection of robust models. However, to ensure its reliability and practical applicability, the Resilience Index must be validated using diverse and realistic adversarial examples.

RQ5. Which attacks maintain consistent performance across different model architectures, and how are they impacted by various defense strategies? (Chapter 6) Among the attacks we analyzed in Section 6.7.4, the PGD, FGSM, and DeepFool attacks demonstrate consistent performance across different model architectures. These attacks exhibit high median success rates, ranging between 0.88 and 0.82, indicating their reliability and effectiveness in overcoming different ML architectures and exploiting common model weaknesses.

The attacks are impacted differently by various defense strategies, with distinct patterns observed in their effectiveness, as detailed in Section 6.9. Adversarial training significantly reduces the success rates of Group 1 attacks (PGD, FGSM, DeepFool, HSJ, and CW2), dropping success rates to as low as 0.05 for PGD and 0.03 for FGSM, indicating its effectiveness in mitigating these attacks. However, ensemble learning proves counterproductive for this group, increasing success rates to 0.98 for PGD and 0.95 for FGSM, suggesting it may introduce exploitable vulnerabilities. Conversely, for Group 2 attacks (ZOO, JSMA, and CW∞), ensemble learning effectively reduces success rates, such as lowering JSMA to 0.13 and CW∞ to 0.09, highlighting its robustness. Adversarial training, on the other hand, increases susceptibility for these attacks, with success rates rising to 0.44 for ZOO and 0.39 for CW∞. This analysis underscores the importance of tailoring defense strategies to specific attacks to enhance overall model resilience.

RQ6. How can adversarial attacks be effectively detected and classified while maintaining high performance on both clean and perturbed data and minimizing overhead, and can integrating adversarial training with ensembling techniques achieve this? (Chapter 6)

We integrated adversarial training with ensembling to increase overall model resilience, as detailed in Section 6.8.3 and 6.5. Adversarial training enhances a model's ability to detect adversarial examples by exposing it to perturbed data during training, thereby improving its robustness. Ensembling, on the other hand, diversifies model perspectives by combining multiple models, which boosts resilience against attacks through varied data interpretations. By leveraging the strengths of both approaches, we achieved better generalization across clean and adversarial examples, providing a multi-layered defense against adversarial attacks.

We trained a range of models using adversarial training and selected the best-performing one. To enhance it further, we built a homogeneous ensemble of this top model. We selected homogeneous ensembling to reduce overhead, as it involves using multiple instances of the same model type. This approach simplifies optimization and parallelization, streamlines implementation, and ensures efficient resource allocation. Additionally, it offers more predictable and consistent training and inference times. In contrast, heterogeneous ensembling introduces higher complexity and resource demands. We employed two techniques for building this ensemble: bagging and boosting. Bagging involves training multiple models on different subsets of the data and aggregating their predictions, enhancing stability and accuracy. Boosting involves

iteratively training the models and improving performance on misclassified instances, resulting in a robust final model.

1.4. Contributions

This thesis has made significant contributions to adversarial machine learning in network intrusion detection, including:

1. We conducted threat modeling for ML-based NIDS using STRIDE and Attack Tree techniques that comprehensively identified and assessed potential threats and vulnerabilities. The proposed threat models enable designing effective countermeasures to prevent potential attacks and deploy resilient ML-based NIDS. **(In answer to RQ1)**

2. We surveyed the body of knowledge and discussed strengths and limitations. We conducted a meta-data analysis to uncover research trends and patterns and derived key findings. We suggested enhancements and identified directions for impactful future research. **(In answer to RQ2)**

3. We validated the compliance of generic adversarial attacks with network domain constraints and assessed their feasibility for real-world scenarios. We implemented and compared these attacks on a DL-based NIDS using multiple datasets, examined the generated perturbations and established a notion of "attack unrealism." This assessment reveals their performance and realism characteristics. **(In answer to RQ3)**

4. We provided a comprehensive evaluation of various adversarial attacks, detection models, and defense methods. We introduced the Resilience Index a standardized measure to compare the ML-based NIDS's resilience for informed deployment decisions, and analyzed attacks performance's consistency across different defense mechanisms for devising tailored countermeasures for each attacks type. **(In answer to RQ4 and RQ5)**

5. We conducted a comprehensive evaluation of ensemble learning and adversarial training and compared their strengths and weaknesses. We developed the Adversarial-Resilient NIDS, a framework featuring a lightweight adversarially trained ensemble that excels at detecting adversarial attacks while maintaining high accuracy on clean data. **(In answer to RQ6)**

1.5. Thesis Structure

The rest of this thesis is structured as follows:

- **Chapter 2. Background:** This chapter presents the fundamental concepts that form the basis of this research. It provides background information related to network intrusion detection, machine learning, and adversarial machine learning.

- **Chapter 3. Threat Modeling for Machine Learning-Based Network Intrusion Detection Systems:** This chapter proposes threat models for ML-based NIDS using

Attack Tree and STRIDE approaches. Attack Tree modeling identifies threats exploiting ML algorithm vulnerabilities, while STRIDE uncovers additional technical threats in the system's data flow. The analysis identified 46 potential threats, providing insights into various attack vectors and aiding in the development of hardening measures to prevent such attacks. **(Contribution 1)**

- **Chapter 4. Adversarial Machine Learning in Network Intrusion Detection Domain: A Systematic Review:** This chapter surveys current literature, categorizing research into three areas: creating tailored adversarial attacks for ML-based NIDS, assessing system resilience, and developing defense mechanisms. It discusses the literature's strengths, limitations, and improvement opportunities. It also addresses current shortcomings, and offers suggestions for future research. **(Contribution 2)**

- **Chapter 5. Realism versus Performance for Adversarial Examples Against DL-based NIDS:** This chapter investigates the vulnerability of DL-based NIDS to adversarial examples by analyzing the effectiveness and realism of various generic attacks across two different datasets. It provides examination of the perturbations generated by these attacks and establishes "attack unrealism" using a set of characteristics whose presence invalidates the realism of the perturbed traffic data. It offers a constrasting analysis of the attacks' performance and realism, and provides a discussion on the practicality and feasibility of these attacks in real-world scenarios. **(Contribution 3)**

- **Chapter 6. Resilience Evaluation and Detection of Adversarial Attacks in ML-based NIDS:** This chapter presents a comprehensive evaluation of various adversarial attacks, models, and defense methods. It introduces the Resilience Index to evaluate the resilience of ML-based NIDS and analyzes attack performance consistency across different defenses and detection models. It compares ensemble learning and adversarial training as defense mechanisms. Finally, it introduces the Adversarial-Resilient NIDS, a multi-layered framework that effectively detects and classifies adversarial attacks, excelling in detection accuracy and adversarial resilience using a lightweight adversarially trained ensemble. **(Contribution 4 & 5)**

- **Chapter 7. Conclusion:** This chapter provide a discussion for the thesis, outlines our future work, and highlights challenges and further research opportunities.

1.6. Publications

Throughout the thesis, certain chapters 3,4, and 5 have been formed from my publications, which represent my original work and were conducted under the editorial and supervisory guidance of my supervisor, Charles Morisset. The following is a list of these publications:

- Alatwi, Huda Ali, and Amjad Aldweesh. "Adversarial black-box attacks against network intrusion detection systems: A survey." 2021 IEEE World AI IoT Congress (AIIoT). IEEE, (2021). [9] **(Chapter 4)**.

- Alatwi, Huda Ali, and Charles Morisset. "Threat Modeling for Machine Learning-Based Network Intrusion Detection Systems." 2022 IEEE International Conference on Big Data (Big Data). IEEE, (2022) [11]. **(Chapter 3)**
- Alatwi, Huda Ali, and Charles Morisset. "Realism versus Performance for Adversarial Examples Against DL-based NIDS." In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, (2023) [12]. **(Chapter 5)**

Chapter 2. Background

2.1. Summary

In this chapter, we establish the foundation for comprehending network intrusion detection, machine learning, deep learning, and adversarial machine learning. We start by distinguishing between host-based and network-based intrusion detection systems and presenting detection methodologies. Then, we explore the fundamentals of machine learning and the differentiation between shallow and deep learning. Lastly, we present the concepts of adversarial machine learning, followed by a detailed examination of various adversarial attack strategies and the associated threat model.

2.2. Network Intrusion Detection Fundamentals

Intrusion detection is the process of dynamically monitoring and inspecting events over a network or system for suspicious behaviors that pose possible threats [29, 13]. An **intrusion detection system** is a hardware device or software application that performs this task, and it can be primarily host-based or network-based [29]. The **host-based IDS** monitors and logs malicious behaviors on a single host, such as unauthorized access attempts or file integrity violations. In contrast, the **network-based IDS** monitors network traffic and inspects different network layers for suspicious activities, such as a flooding flow indicating a DoS attack [29]. The NIDS utilizes three major methodologies to detect malicious activities: signature-based, anomaly-based, and stateful protocol analysis [29]. A **signature-based IDS** (i.e., misuse detection) detects intrusions by matching network traffic to predefined patterns of known attacks with very low false alarms [13]. The significant disadvantage of this approach is that it fails to detect zero-day attacks. An **anomaly-based IDS**, also known as behavior-based IDS, leverages statistical and machine learning techniques to identify anomalies and novel attacks by comparing network traffic against a baseline of normal behavior [13]. This approach aims to detect deviations that may indicate potential threats. Zero-day attacks, which exploit previously unknown software vulnerabilities before a patch can be released, pose a significant challenge to network security because they leave systems vulnerable without an immediate defense [29]. While anomaly-based IDS can potentially identify these attacks, its effectiveness is often hindered by a lack of accuracy and a high rate of false positives [29]. A **stateful protocol analysis IDS** (i.e., deep packet inspection) inspects the packets' content and tracks the state of application protocols used over the network. It compares observed traffic to the normal activity of a protocol to discover deviations [29].

2.3. Machine Learning Fundamentals

Machine learning empowers computers to accomplish tasks without being explicitly programmed by learning from a provided dataset to solve the problem at hand. Machine learning encompasses various algorithms that can be categorized according to their approach, application, and depth [88].

2.3.1. Machine Learning Approaches

- **Supervised learning,** is a type of machine learning where the algorithm is trained on a dataset with labeled examples. Each input data point in this dataset is paired with a corresponding output, which could be a discrete label (in the case of classification) or a continuous value (in the case of regression). The algorithm's objective is to learn the relationship between the inputs and their associated outputs so that it can accurately predict the output for new, unseen data [88, 100, 91].
- **Unsupervised learning,** is a type of machine learning where the algorithm is trained on data that has no labeled outputs. Unlike supervised learning, where the goal is to predict a known label, unsupervised learning seeks to find hidden patterns or intrinsic structures in the input data. The algorithm works independently to group, cluster, or reduce the dimensions of the data, helping to uncover the underlying relationships between data points [88, 100, 91].
- **Semi-Supervised learning,** is a machine learning approach that lies between supervised and unsupervised learning. It involves training a model using a combination of a small amount of labeled data and a large amount of unlabeled data. The idea is to leverage the small labeled dataset to guide the learning process, while the large unlabeled dataset helps improve the model's accuracy and generalization [88, 100].
- **Reinforcement learning,** is a machine learning approach where an agent learns to make decisions by interacting with its environment [91]. Unlike traditional methods that require labeled data, RL involves the agent exploring the environment, taking actions, and receiving feedback in the form of rewards or penalties. The agent's objective is to develop a strategy, or policy, that maximizes the total accumulated rewards over time. By continuously refining its actions based on the outcomes, the agent progressively improves its decision-making, ultimately aiming to achieve the best possible results in a given task.

2.3.2. Machine Learning Tasks

Based on the approach and nature of available data, machine learning algorithms are used to build models that can achieve one of these common tasks [47]:

- **Classification** learns a predictive function from a pre-labeled dataset that can assign a class label to an unseen data instance. It approximates a mapping function (f) from labeled input instances (X) to discrete output classes (y).

- **Clustering** discovers hidden patterns in the feature space of unlabeled inputs and groups these points based on similarities or distance measures into unknown classes (i.e., clusters).
- **Dimensionality Reduction** transforms a dataset from a high-dimensional space into a compact low-dimensional space while maintaining the significant characteristics of the original dataset as much as possible.
- **Regression** learns a predictive function by examining the relationship between a set of predictor (independent) variables and response (dependent) variables, estimating a continuous-value outcome for a given observation accordingly.
- **Association Rule Learning** uses metrics of interestingness to identify strong rules that relate the antecedent (X) to the consequence (Y).

2.3.3. Machine Learning Depth

- **Shallow Learning** encompasses conventional machine learning techniques that do not utilize multiple layers or hidden connections [88].
- **Deep Learning** uses multiple layers of nodes in artificial neural networks to extract high-level representations of features from raw inputs [88].

2.4. Machine Learning Pipeline

The ML process involves a series of interconnected steps designed to build, evaluate, and deploy models that can learn from data and make predictions. Figure 2.1 outlines the key stages in the ML pipeline, from data collection and data processing to model selection, training, evaluation, and deployment [50].

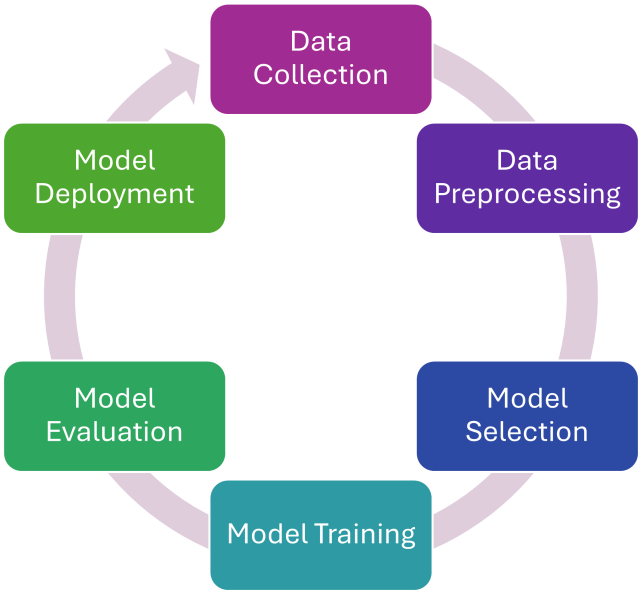

Figure 2.1 Machine Learning Pipeline

2.4.1. Problem Definition

The first step in developing a ML-based NIDS is to clearly define the problem by understanding the network environment and identifying the types of data and potential threats [50]. Objectives must be specified, such as focusing on detecting known attacks, discovering novel threats, or both. It's also important to determine the appropriate machine learning approach—supervised, unsupervised, or semi-supervised—based on the availability of labeled data [50]. Additionally, understanding requirements like acceptable false positive rates, real-time detection, and scalability is crucial to ensuring the NIDS design is effective.

2.4.2. Data Collection

After defining the problem, the next crucial step is data collection, which directly impacts the effectiveness of a ML-based NIDS. Data can be sourced from network traffic captures (e.g., Wireshark, Tcpdump), firewall and router logs, IDS alerts, and synthetic datasets [105]. It should reflect both normal network behavior and a variety of malicious activities.

For supervised learning, the data needs to be labeled as benign or malicious. This labeling process is labor-intensive and requires expertise. If labeled data is not available, unsupervised learning techniques may be employed, though they face challenges in interpreting anomalous behavior. The collected data must be representative of the real-world network environment, including various times and conditions [105]. Data preprocessing tasks include packet decoding, feature extraction (e.g., IP addresses, port numbers), and aggregation of network flow [105]. Regular updates are necessary to address evolving threats and maintain the NIDS's effectiveness.

2.4.3. Data Preprocessing

Raw data is often noisy, incomplete, and inconsistent, making data preprocessing a crucial step in the machine learning pipeline. Data preprocessing involves several tasks, including [50, 55, 7, 105]:

- **Data Cleaning:** This step involves handling missing values, removing duplicates, and correcting errors. Techniques such as imputation, where missing values are filled in based on statistical methods, or outlier detection, where anomalous data points are identified and handled, are commonly used [50, 55, 7, 105].

- **Data Transformation:** Transforming data into a suitable format or structure is essential for model training. This may include normalization or standardization of numerical features, encoding categorical variables, and feature extraction to reduce dimensionality [50, 55, 7].

- **Data Splitting:** To evaluate the performance of a machine learning model, the data is typically split into training, validation, and test sets. The training set is used to fit the model, the validation set is used to tune hyperparameters and prevent overfitting, and the test set is used to assess the model's generalization ability on unseen data [50, 55, 7, 105].

2.4.4. Model Selection

Model selection in ML-based NIDS involves choosing algorithms suited to detecting network anomalies and attacks. Key models include [105]:

- **Linear Models:** Simple and interpretable, such as logistic regression, useful for detecting specific patterns but may lack depth for complex attack scenarios.
- **Decision Trees and Ensemble Methods:** Decision trees offer clarity in rules, while ensemble methods like Random Forests and Gradient Boosting improve detection accuracy and handle varied attack patterns.
- **Support Vector Machines (SVM):** Effective in high-dimensional feature spaces, SVMs can distinguish between normal and malicious network activities with high precision.
- **Neural Networks:** Deep learning models, including Convolutional Neural Networks (CNNs) for feature extraction and Recurrent Neural Networks (RNNs) for sequential data, are adept at identifying sophisticated and evolving attack patterns.
- **Clustering Algorithms:** Techniques like k-means and DBSCAN are used for anomaly detection by grouping similar network behaviors and highlighting deviations indicative of potential attacks.

2.4.5. Model Training

Once a model is selected, the training process begins. Model training involves feeding the training data into the algorithm and adjusting the model's parameters to minimize the error between its predictions and the actual outcomes. This is typically done by optimizing a loss function, such as cross-entropy [50, 55, 7].

- **Optimization Algorithms:** Techniques like Gradient Descent, including its variants (SGD, Adam), are used to minimize the error between predicted and actual network behaviors by adjusting model parameters.
- **Hyperparameter Tuning:** Key settings, such as learning rate and number of layers, are optimized using methods like grid search or Bayesian optimization to enhance model performance.
- **Regularization:** To avoid overfitting, methods such as L1/L2 regularization, dropout, and early stopping are applied to ensure the model generalizes well to new, unseen network data.

2.4.6. Model Evaluation

After training, the model must be evaluated to ensure it effectively detects network anomalies and attacks. This is achieved by using validation and test datasets to assess how well the model generalizes to new and unseen network traffic. Key aspects of model evaluation include [50, 55, 7]:

- **Performance Metrics:** metrics such as accuracy, precision, recall, and ROC-AUC are used to assess how well the model detects network anomalies and attacks.
- **Cross-Validation:** Cross-validation is a technique where the training data is split into multiple folds, and the model is trained and evaluated multiple times, each time on a different fold. This provides a more robust estimate of the model's performance and helps in detecting overfitting.

2.4.7. Model Deployment

After training, the model must be evaluated to ensure it effectively detects network anomalies and attacks. This is achieved by using validation and test datasets to assess how well the model generalizes to new and unseen network traffic. Key aspects of model evaluation include [50, 55, 7]:

- **Scalability:** The model must handle high volumes of network traffic and provide real-time threat detection with minimal latency.
- **Monitoring:** Continuous monitoring is essential to detect any performance degradation due to data drift, where the network behavior changes over time, affecting the model's accuracy.
- **Model Retraining:** Regular retraining with new network data is crucial to ensure the model remains effective against evolving threats. Automated retraining pipelines are often implemented to streamline this process in production environments.

2.5. Deep Learning Fundamentals

Deep learning is a subfield of machine learning based on artificial neural networks. **Neural Networks** are biologically inspired by the structure of the human brain and imitate its behavior to solve complex data-driven problems. A neural network consists of layers of perceptrons, also known as neurons, which are the core processing units of the network. A perceptron is a fundamental unit in a neural network that takes input values, applies a set of weights to them, and passes the weighted sum through an activation function to produce an output. This process enables the network to learn and make decisions based on the input data [13, 33, 58].

A neural network mainly consists of an input layer that acquires the inputs, an output layer that predicts the final result, and one or multiple hidden layers (i.e., deep neural network) in between that perform complex computations [13, 33, 58]. The neurons in one layer are connected to neurons in the next layer through links. These links are initially assigned random numerical values termed **weights** [58]. Each neuron multiplies its inputs by their corresponding weights and adds a **bias**, which is a constant value specific to that neuron. The result, known as the **weighted sum**, is then passed through a **non-linear activation function**, which determines whether the neuron should be activated or not. This activation function serves as a threshold, allowing the neuron to produce an output only if certain criteria are met.

When a neuron is activated, its output is transmitted to neurons in the subsequent layer; this process of passing data through the network is called **forward propagation**. In the output layer, the neuron with the highest activation value determines the final output, which is expressed as a probability for the prediction [13, 33, 58].

The network is trained using the **back-propagation** algorithm, which adjusts the weights to minimize prediction error, or the difference between the actual output and the predicted output [13, 33, 58]. During back-propagation, the error is propagated backward through the network. This involves calculating the gradient of the error with respect to each weight and bias using the chain rule of calculus [13, 33, 58]. The algorithm updates the weights by moving them in the direction that reduces the error, effectively improving the network's accuracy. Weights that contribute to correct predictions are increased, while weights leading to incorrect predictions are decreased [13, 33, 58]. The network undergoes iterative cycles of forward propagation and back-propagation, refining its weights with multiple inputs until it achieves a high accuracy in predictions. Figure 2.2 shows an illustration of a neuron in a neural network[1].

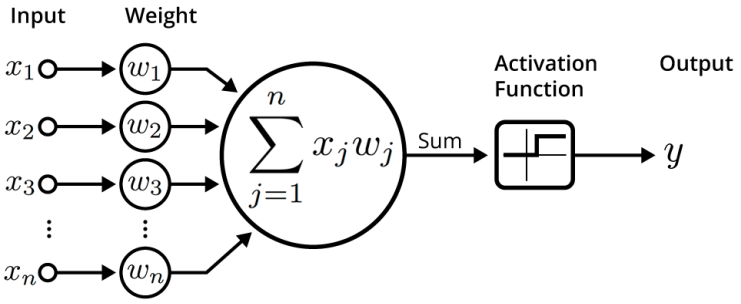

Figure 2.2 Neuron Structure in DNN

Deep neural networks (DNN) are advanced enough to perform feature engineering using raw inputs independently [13, 58]. The use of multiple hidden layers enables the extraction of features and the discovery of latent structures in unprocessed and unlabeled inputs. Figure 2.3 graphically illustrates the architecture of a typical deep neural network composed of multiple layers (at least two hidden layers) of neurons. Each layer learns to transform its multidimensional inputs into a more compact and abstract representation. Therefore, the network can be viewed as a highly complex non-linear mapping function that converts high-dimensional inputs into a lower-dimensional space [13, 58].

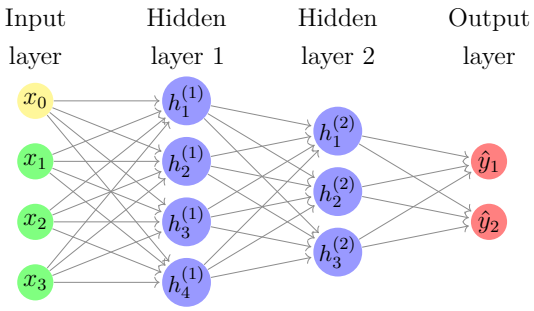

Figure 2.3 DNN Structure

[1]Source: becominghuman.ai

2.6. Adversarial Machine Learning Fundamentals

Szegedy et al.[151] revealed for the first time that small but intentionally crafted perturbations (i.e., adversarial examples) added to the training inputs could lead neural network models to misclassify, such as wrongly classifying a "panda" as a "gibbon". These examples were found to be transferable between models; examples designed to fool a particular a deep learning model can also be effective in misleading different models. This property is called transferability. Such examples that can transfer between different models are known as cross-modal examples. Another form of transferability is cross-dataset, where adversarial examples generated over a particular dataset can be used to attack another model trained on a different dataset[71]. Furthermore, adversarial examples generated to deceive a specific neural network can fool the same network even when it is trained on different datasets. In practice, adversaries utilize these transfer-based attacks by using outputs from the targeted model to train a substitute model and then crafting adversarial examples for the surrogate model that can transfer to the original model in a black-box setting [118].

Machine learning algorithms depend on data for building models and performing classification tasks. These learned models can make incorrect decisions with high confidence due to carefully crafted and manipulated perturbations added to legitimate inputs, known as **adversarial examples**[33]. In other words, an adversarial example is a data instance with tiny intentional feature perturbations that deceive models and cause them to misclassify. Adversarial examples undermine machine learning algorithms' fundamental assumption that training and testing datasets have the same distribution (data stationarity)[33]. The models assume that the distribution of test data follows the training data, which is not always the case in the real world. Adversaries exploit this flawed assumption and use combinatorial optimization, local search, and convex programming to discover adversarial examples that compromise model security [158].

Most approaches for generating adversarial examples add a calculated perturbation (γ) to a legitimate example (x) to generate a new version (x') while minimizing the distance between the legitimate example (x) and the adversarial example (x'), and shifting the prediction to the intended adversarial result. Adversarial attacks involve feeding these adversarial examples (i.e., perturbed inputs) to the model either during the training phase (poisoning attack) or inference phase (evasion attack) to force it into making wrong decisions. From the attacker's perspective, these examples are crafted to bypass detection by the models, degrade their performance, and probe them to gain information about model parameters and training datasets. From the defender's perspective, they are used to conduct vulnerability assessments and debugging for the models, evaluate their resilience to noise, and increase their generalization abilities. Adversarial machine learning is an emerging research field focused on making the models robust against adversarial attacks by assessing vulnerabilities and designing appropriate defensive mechanisms [70].

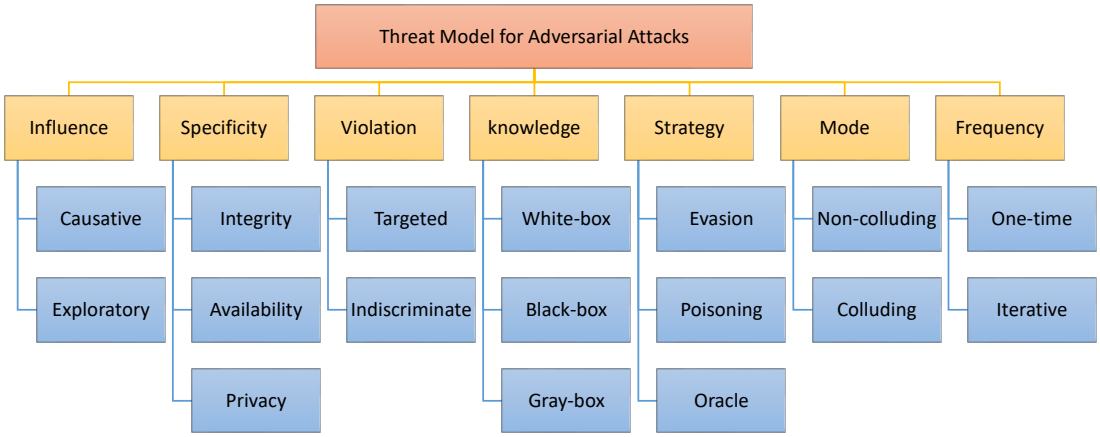

Figure 2.4 Threat Model of Adversarial Attacks

2.6.1. Adversarial Threat Model

The adversarial threat model provides an abstraction to profile the adversary based on aspects such as its capabilities, goals, knowledge, and strategies. A widely established taxonomy of adversarial attacks is proposed by Huang et al. [70]. It classifies attacks based on three main characteristics as follows:

- **Influence** refers to the adversary's capabilities over the target model. Accordingly, these attacks can be causative or exploratory. **Causative attacks** occur during training and aim to affect the learning process of the model, also known as poisoning attacks. **Exploratory attacks** occur after the model's deployment and may include probing the model to discover information about its internals or its training dataset, such as in model stealing attacks.

- **Security violation** refers to the element of the CIA triad (confidentiality, integrity, availability) that the adversary compromises. It can be classified into integrity attacks, availability attacks, and privacy attacks. **Privacy attacks** result in the disclosure of information that violates the model's secrecy or the privacy of its users. **Integrity attacks** aim to manipulate a model so that it misclassifies certain data instances. These attacks typically cause true positives—correctly identified anomalies—to be classified as false negatives, incorrectly labeling them as normal. The goal is to undermine the model's accuracy by introducing adversarial inputs that evade detection. Such attacks are particularly dangerous in critical systems like network intrusion detection, where undetected threats can have severe consequences. **Availability attacks** seek to render a model unusable by overwhelming it with inputs that cause widespread errors, including both false positives and false negatives. Rather than targeting specific instances, these attacks degrade the model's overall performance, making it unreliable. This is often done by flooding the system with noise, leading to a breakdown in the model's functionality.

  The key difference between these attacks is their focus: integrity attacks selectively alter specific outcomes to reduce accuracy, while availability attacks disrupt the entire system, leading to broader operational failure.

- **Specificity** refers to the span of the adversary's inputs that are planned to be misclassified. It can be classified into targeted attacks or indiscriminate attacks. **Targeted attacks** aim to force the model to misclassify a particular data instance or a small subset of data instances. **Indiscriminate attacks** aim to force the model to misclassify a general class of data instances, such as classifying all true positive data points as false negatives. This attack degrades the model's reliability by maximizing prediction errors in general.

This taxonomy has been expanded to comprehensively cover definitive assumptions, which include the following [30]:

- **Adversary knowledge** refers to the extent of accessible knowledge about the model and its structural properties to the attacker. It can be categorized into white-box attacks, black-box attacks, and gray-box attacks. In **white-box attacks**, the adversary has complete knowledge about the model and its parameters (e.g., features, training dataset, algorithm, hyperparameters, and in the case of a neural network, the model's architecture, weights, activation function, and the number of layers). Such knowledge is exactly what the creator of the model has, which in the majority of real-world scenarios is not feasible for the adversary to have. In **black-box attacks**, the adversary has no knowledge about the model or its parameters and only knows the model's returned output (i.e., labels or confidence scores). This setting is commonly assumed for attacking online machine learning services. Most adversarial attacks are white-box; however, due to the transferability of adversarial examples (AEs), they can be used for black-box attacks as well. In **gray-box attacks**, the adversary has constrained knowledge or limited access (e.g., access to the training dataset, access to the predicted classes, access to the predicted probabilities, or a definite number of queries to the model).

- **Strategy** refers to the adversary's phases of actions for launching the attack, which can be one of the following: evasion attacks, poisoning attacks, or oracle attacks. **Evasion attacks**, also known as exploratory attacks, involve manipulating inputs to deceive a previously trained model. In this attack, no influence over the training data is assumed. For instance, in a NIDS case, the attack payload is encoded to evade detection, compromising the targeted system. Additionally, the adversary may aim to provoke concept drift in the system, causing continuous retraining and consequently degrading its performance. Evasion attacks typically use an optimization approach to find a small perturbation (σ) that maximizes the loss function. Such an increase in the loss function can be significant enough to cause the model to misclassify. **Poisoning attacks**, also known as causative attacks, involve corrupting the training data or model structure during training to compromise the learning process, violate the model's integrity, and degrade its performance in the deployment phase. The adversary trains an anomaly-based NIDS with a labeled attack dataset as ground truth, causing the system to fail to detect cyber attacks. In an **oracle attack**, the adversary creates a substitute model that retains most of the original model's functionality and designs attacks against it, which are then launched against the original targeted model. Although this type of attack enables attacking the targeted system in

a black-box setting, it suffers from transfer loss as not all adversarial examples transfer effectively from one model to another. Therefore, a very large number of training instances are needed to train the substitute model to enhance the attack's efficacy.

Other researchers have extended the threat model further to cover the following:

- **Attack Mode**: It is commonly assumed in most related literature that the adversary works alone to achieve the attacks, known as **non-colluding attacks**. Another possible assumption is that multiple adversaries can cooperate to boost attack efficiency and hide their traces, known as **colluding attacks**[126].

- **Attack Frequency** refers to how often an adversarial example is optimized. Attacks are categorized into one-time or iterative attacks based on the frequency of optimizing or updating the generated adversarial examples. **One-time attacks** optimize the generated adversarial examples once without any iteration. On the other hand, **iterative attacks** update adversarial examples multiple times. For a real-time attack, adversaries should choose a one-time attack instead of an iterative attack. Iterative updating results in better optimized adversarial examples that overcome one-shot attacks; however, this approach is computationally expensive and requires more interactions with the targeted model (i.e., more queries)[176].

Adversarial examples are intended to be as close as possible to the original examples. Therefore, introducing tiny perturbations is the fundamental premise for generating them. There are three aspects to analyze the introduced perturbation: perturbation limitation, perturbation scope, and perturbation measurement [176].

- **Perturbation limitation**
  - **Optimized perturbation**, define perturbation as the target of the optimization problem and aim for minimizing the introduced perturbation as much as possible.
  - **Constraint perturbation**, define perturbation as the constraint of the optimization problem and demand it to be small enough.

- **Perturbation scope**
  - **Individual attacks**, craft distinct perturbations for each clean instance of the inputs.
  - **Universal attacks**, craft a general perturbation for the entire dataset, and this general perturbation is applicable to any clean input. Most of the existing adversarial techniques create the perturbations individually. Universal perturbations facilitate deploying adversarial examples in the real world. In such a case, the adversary does not need to amend the perturbation when the input changes.

- **Perturbation measurement**
  - l_p measures introduced perturbations by p-norm distance [176]. There are three l_p distance metrics in the literature l_0, l_2, and l_∞. l_0 counts the number of features to be perturbed. l_2 measures the Euclidean distance between the original and adversarial example. l_∞ denotes maximum magnitude of perturbation added to each feature.

– Psychometric perceptual adversarial similarity score (PASS) is a metric for quantifying adversarial images.

2.6.2. Adversarial Attacks

Adversarial examples are used to attack machine learning models at training time (poisoning attacks) or at inference time (evasion attacks). To recall, white-box attacks can be launched when the adversary has access to model gradients, such as model weights. In contrast, in black-box attacks, the adversary knows almost nothing about the targeted model and depends on the returned outputs to tweak the perturbations. White-box attacks are more powerful than black-box ones as they succeed in crafting adversarial examples that target the victim model with minimal effort and time and do not rely on the attacks' transferability. The robustness of adversarial example generation techniques depends on their ability to produce adversarial examples that are as close as possible to the original examples.

Jacobian-based Saliency Map Attack (JSMA) identifies the most influential features of input data by analyzing the Jacobian matrix, which measures how changes in each feature affect the model's output. It then manipulates these critical features to craft adversarial examples that mislead the model while making minimal alterations to the input [119]. The Basic Iterative Method (BIM) applies iterative small perturbations to the input data, optimizing them over multiple steps to generate adversarial examples [84]. Carlini-Wagner (CW) formulates the generation of adversarial examples as an optimization problem, aiming to find the minimum perturbation that induces misclassification while considering a specific distance metric [31].

DeepFool calculates the minimum perturbation required to shift the model's decision boundary for a given input, crafting adversarial examples with minimal changes that are highly likely to be misclassified by the target model [104]. Fast Gradient Sign Method (FGSM) is a one-step attack that perturbs input data in the direction of the gradient of the loss function to create adversarial examples [59]. Projected Gradient Descent (PGD) combines the principles of BIM and FGSM by iteratively applying small perturbations and projecting the perturbed input back onto a valid data space [90].

Zeroth Order Optimization (ZOO) leverages only model query access to iteratively generate adversarial examples, effectively bypassing model defenses and posing a significant threat in scenarios where model architecture and parameters are unknown [39]. Hopskipjump (HSJ) generates adversarial examples by combining gradient-based optimization with a greedy search, exploiting model vulnerabilities with limited queries and making it a potent threat against deep learning models [36].

2.7. Conclusion

In this chapter, we laid the groundwork for understanding network intrusion detection, machine learning, deep learning, and adversarial machine learning. We distinguished between host-based and network-based intrusion detection systems, investigating their detection methodologies.

Additionally, we delved into various machine learning approaches, tasks, and the differentiation between shallow and deep learning. We introduced the concept of adversarial machine learning and provided a detailed exploration of various adversarial attack strategies and their associated threat models.

In understanding the foundational concepts of ML-based NIDS, we set the stage for identifying the inherent vulnerabilities and potential threats these systems face in real-world applications. This leads us to the critical task of threat modeling, where we systematically analyze and address these security challenges to enhance the robustness and reliability of ML-based NIDS, which will be discussed in detail in the following chapter.

Chapter 3. Threat Modeling for ML-based NIDS

3.1. Summary

To ensure NIDSs play their vital roles in securing the network against cyber attacks, it is necessary to identify how they can be attacked by adopting a viewpoint similar to the adversary to identify vulnerabilities and defenses hiatus. Accordingly, effective countermeasures can be designed to thwart any potential attacks. NIDS, employing machine learning approaches, found to be susceptible to adversarial attacks where subtle perturbations are inserted into inputs during inference to evade classifier detection or during training to degrade performance. Yet, modeling adversarial attacks and the associated threats of employing the machine learning approaches for NIDSs was not addressed. One of the growing challenges is to avoid ML-based systems' diversity and ensure their security and trust. In this chapter, we conduct threat modeling for ML-based NIDS using STRIDE and Attack Tree approaches to identify the potential threats on different levels. We model the threats that can be potentially realized by exploiting vulnerabilities in machine learning algorithms through a simplified structural attack tree. To provide holistic threat modeling, we apply the STRIDE method to systems' data flow to uncover further technical threats. Our models revealed a noticing of 46 possible threats to consider. These presented models can help to understand the different ways that a ML-based NIDS can be attacked; hence, hardening measures can be applied to prevent these potential attacks from achieving their goals.

3.2. Introduction

Despite ML-based NIDS's ability to detect known and unknown malicious traffic with high accuracy, adversaries continually evolve their techniques to evade detection, with adversarial attacks emerging as a prominent example. [9, 10]. These attacks manipulate input data to evade detection at inference time, or inject malicious data during training to compromise the integrity of the model [9, 10]. Additionally, they can be utilized to probe the detection model with crafted queries to reverse-engineer it, potentially revealing confidential information. [159, 71].

The attack surfaces of ML-based systems are widening, and their security and trust are significant concerns. Yet, modeling the security threats against these systems, particularly the ML-based NIDS, was not considered. To address this gap, we propose two threat models for ML-based NIDS that identify the potential threats that can be realized due to loopholes in

the machine learning algorithms or the lack of security controls. Therefore, the main research question this chapter attempts to answer:

What are the potential threats and vulnerabilities associated with employing machine learning approaches in NIDS, and what effective countermeasures can be devised to enhance their security and trustworthiness?

Threat modeling is a well-accepted paradigm for developing a secure system that involves identifying, enumerating, and prioritizing the potential security threats [145]. It intends to equip the system defenders with an analysis of what security measures need to be applied to mitigate the identified threats. There are different threat modeling approaches; however, we adopt the Attack Tree and STRIDE models based on our careful assessment. Attack trees provide a systematic representation of systems security based on varying attacks [108]. We utilize the attack tree modeling to enumerate all the possible threats associated with vulnerabilities and loopholes within machine learning approaches that an adversary can exploit to attack an ML-based NIDS. To spot the other technical threats, we apply the STRIDE strategy at the components level including the data flow among them. STRIDE is an acronym for six groups of threats Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [98]. Our proposed models identified 46 noticing threats against ML-based NIDS. We believe these threat models will assist ML-based NIDS designers in assessing their approaches and in designing hardening measures.

Contributions: The contributions of this chapter are as follows:

- **C1:** To identify potential threats and attack vectors that could compromise the ML-based NIDS security.
- **C2:** To assess the vulnerabilities within the ML-based NIDS that could be exploited by the identified threats.
- **C3:** To Assess threats impact and likelihood to determine their associated risk levels.
- **C4:** To suggest countermeasures to mitigate the identified risks.

Organization: The rest of the chapter is organized as follows: Sec 3.3 presents some related work. Sec 3.4, provides a brief background on threat modeling approaches. Sec 3.5, explains our threat modeling methodology. The proposed threat models for ML-based NIDS are given in Sec 3.6 and Sec 3.7. Sec 3.8 provides a discussion. Sec 3.9 presents the conclusion.

3.3. Related Work

Research on ML-based NIDS security encompasses three key areas: developing adversarial attacks, evaluating model resilience against evasion techniques, and improving model resilience. The first area focuses on proposing evasion attacks tailored for NIDS, employing techniques like Reinforcement Learning [23], Generative Adversarial Networks [183, 14], and Genetic Algorithms [14]. The second area assesses the resilience of various NIDS models against generic evasion adversarial approaches [77, 43, 124, 132, 92, 45, 52]. The third area aims to bolster ML-

based NIDS resilience against evasion attacks through methods such as adversarial training [132, 92, 45, 52], feature removal [140, 53], and ensemble learning [140, 53]. While significant research attention has been directed towards addressing adversarial attacks, particularly evasion attacks, it's noteworthy that the majority of studies overlook other potential threats confronting ML-based NIDSs realized by outsider or insider threat agents.

Threat modeling is a proactive engineering process that systematically analyzes a system to uncover security flaws. Without it, identifying which parts of the system need strengthening becomes challenging. This chapter employs attack tree and STRIDE threat modeling processes to analyze potential threats against ML-based NIDSs. Additionally, we conduct risk assessment and propose security countermeasures for identified threats. This chapter represents the first attempt to propose threat models for NIDSs utilizing machine learning approaches for malicious traffic detection.

3.4. Background

In this section, we provide brief background knowledge on the fundamentals of threat modeling and the threat modeling approaches utilized in this chapter: the STRIDE model and the Attack Tree model.

3.4.1. Threat Modeling

Threat modeling is a systematic process to identify the potential threats and vulnerabilities of the system from the adversary's point of view and what possible countermeasures can be applied to mitigate these threats [145]. This process is composed of five steps that involve identifying the following elements [107]: **1) Assets,** are items of value and criticality for delivering some operations and attracting the adversary to compromise, such as hardware, software, services, and data. **2) Attack surface,** are the different points of the system that vulnerable to unauthorized entry by some adversary. **3) Adversary model,** are the characteristics that define the adversary, such as who he is and what are his motives and capabilities. **4) Vulnerabilities and Threats,** vulnerabilities refer to the weaknesses within the asset that the adversary can leverage for security compromise. Threats are events in which the adversary exploits the asset's vulnerabilities to mount an attack. **5) Mitigation measures,** are the security solution that can be applied to prevent, detect, or reduce the impact of threats.

3.4.2. STRIDE Model

STRIDE is a threat modeling approach covering six security threat categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege [98]. Each element of the STRIDE approach corresponds to a violation of some desirable security property: authenticity, integrity, non-reputability, confidentiality, availability, and authorization. **Spoofing,** is providing false data that fools the target system into gaining authorized access. **Tampering,** is applying unauthorized updates or modifications to the system, system's

components, or data. **Repudiation,** is denying the responsibility of performing some actions intentionally. **Information Disclosure,** is revealing confidential data to unauthorized parties. **Denial of Service,** is disrupting the system services availability, efficiency, and performance. **Elevation of Privilege,** is an unauthorized increase of privileged access beyond what is initially given.

3.4.3. Attack Tree Model

An attack tree is a conceptual and hierarchical diagram demonstrating how an asset can be compromised [108]. It provides a schematic presentation of the system' security as a function of any viable attacks. The model is tree-structured where the root node is the target and connected with multiple leaf nodes representing the different paths to accomplish the attack goal [108]. The tree shows how low-level malicious activities cooperate to accomplish the adversary's goal. The adversary may reap benefits, and the target endures some impacts at any level of the tree; however, the impacts get more harmful at the higher levels. A successful attack traverses the tree from accomplishing the actions at the leaf nodes to overcoming the target at the root. The fundamental building blocks of the attack tree are as follows [108]:

- **Root node,** is the top node and represents the target of the attack.
- **Leaf nodes,** are the lowest nodes in the tree and denote the actions the adversary can take to reach the attack goal.
- **Intermediate nodes,** are the nodes between the root node and the leaf nodes and represent the adversary's sub-goals and intermediate states towards the ultimate goal. These nodes are either AND or OR nodes, often denoted by the Boolean Algebra shapes. The activities represented by nodes immediately beneath an AND node must all be performed to attain the goal represented by the above AND node. On the other hand, if any of the nodes beneath an OR node are achieved, then the OR node state is also attained.

Certain combinations of leaf nodes' actions will meet the tree's AND/OR nodes' logic and lead to one or more paths toward the ultimate goal. These combinations identify all the possible actions the adversary may take, known as attack scenarios. Different attack scenarios demand different resources and depending on the adversary's capabilities some attacks scenarios more preferable than others. Analyzing which scenario suits the best a given adversary is known as capabilities analysis.

3.5. Threat Modeling for ML-Based NIDS

This chapter aims to perform threat modeling for ML-based NIDS by following the consolidated scheme depicted in Figure 3.1. We identify the components of a ML-based NIDS that reveal functionalities, interactions, and relationships between them. Next, we derive the attack surface based on the access points of each component. Furthermore, we detail the interactions among

the system components using a data flow diagram to derive a complete threat model that covers all possible threats. Afterward, we use two threat identifying approaches to model the potential threats against a ML-based NIDS. We employ the Attack Trees to model threats associated with machine learning algorithms' vulnerabilities. Additionally, we apply the STRIDE method to the data flow diagram to spot further technical threats. As a result, we provide an assessment of the identified threats. Finally, we present possible defensive mechanisms that can aid in mitigating the identified threats.

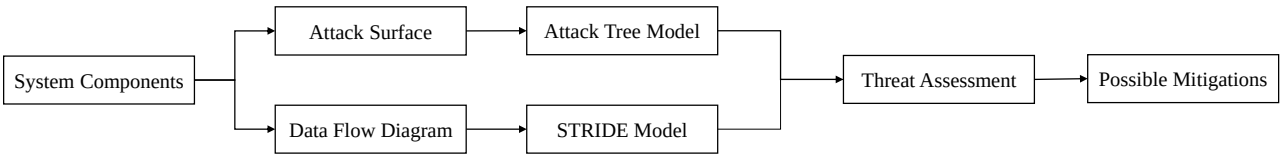

Figure 3.1 Threat Modeling Scheme

3.5.1. ML-based NIDS Components

Figure 3.2 depicts the major components of a typical ML-based NIDS: a data source, a pre-processing module, a detection module, and security responses [105]. **Data source,** is the fundamental component of any NIDS to build up the detection module. Network traffic data needs to be collected first for constructing offline data set that compromises a large diversity of normal and abnormal traffic to train the model. After training the model, real-time traffic is monitored for anomaly detection. A packet sniffer module is responsible for continuously capturing the network traffic and can be implemented by either passive or in-line deployment approaches. Network flows are aggregated at checkpoints based on one feature at a given time, such as protocols or source/destination IP addresses[105]. **Pre-processing module,** transforms the raw traffic data into an understandable format that can be fed as input into the machine learning model. It comprises four activities: features generation, features selection, features conversion, and features normalization. **Feature generation,** constructs feature vectors from the captured packets using tools such as Netflow, Netmate, Tcptrace, Argus, and BRO-IDS. **Feature reduction,** eliminates duplicated, irrelevant, unimportant, and noisy features. It involves two substeps: features selection and features extraction. Features selection identifies a subset of the original features that contribute most to the class labels. Features extraction converts feature vectors from high to low dimensional space using dimensionality reduction approaches. **Feature conversion,** maps the categorical features to numeric vectors using a unified format. **Feature normalization,** applies a scaling function to convert the numeric values to a common scale without amending the differences between them to prevent some features from dominating the others. Common approaches for scaling data are Z-score normalization and Min-Max scaling. **Detection module,** uses the constructed model to analyze the network flow for malicious behavior and makes a classification decision. **Response module,** uses the classification result from the detection module to invoke the appropriate actions for the given flow according to a set of predefined policies.

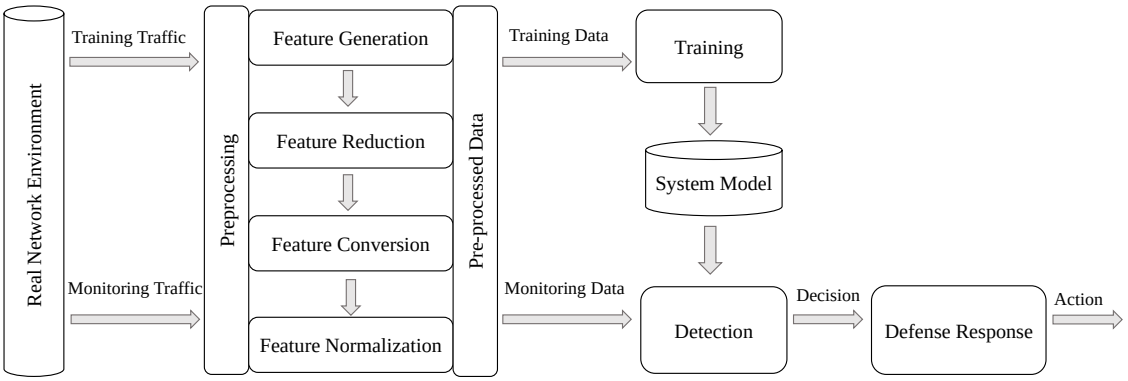

Figure 3.2 Components of ML-based NIDS

3.5.2. Adversary Model

Our threat model considers two profiles of adversaries: insider and outsider. The insider adversary is within the perimeter of the network and can access the ML-based NIDS directly, and the harm he causes can result from intentional or accidental activities. The outsider adversary does not have direct access to the target. However, he is able to initiate the attack remotely or through a compromised node in the network that has direct access to a component within the system. Regarding the adversary's goals, we assume the adversary seeks to launch targeted or reliability attacks. In targeted attacks, the adversary aims to enforce the system to induce a definite prediction for a given input. On the other hand, in reliability attacks, the adversary aims to maximize prediction errors overall without necessarily inducing a certain prediction. Furthermore, we define the capabilities of the adversary as causative or exploratory. In Causative attacks, the adversary is able to affect either the learning process or inference of the detection system. In exploratory attacks, the adversary is able to probe the system to extract information that can be exploited for different purposes.

3.5.3. Attack Surface

We define the attack surface of a ML-based NIDS as all the access points are susceptible to being exploited by both insider and outsider adversaries to launch their attacks. This can be described with respect to the pipeline of the system architecture in Figure 3.2. The system development begins with constructing a detection model based on the pre-collected offline training dataset. After system deployment, the network traffic is collected via sensors and processed before being fed to the detection module, which makes prediction decisions. This output is then communicated to the defense response module to be acted upon. In such a system, the adversary may seek to 1) tamper with collection or pre-processing of input data, 2) subvert the detection model, 3) exploit the model outputs, or 4) interfere with them. Each of these attack surface points is a relevant attack vector.

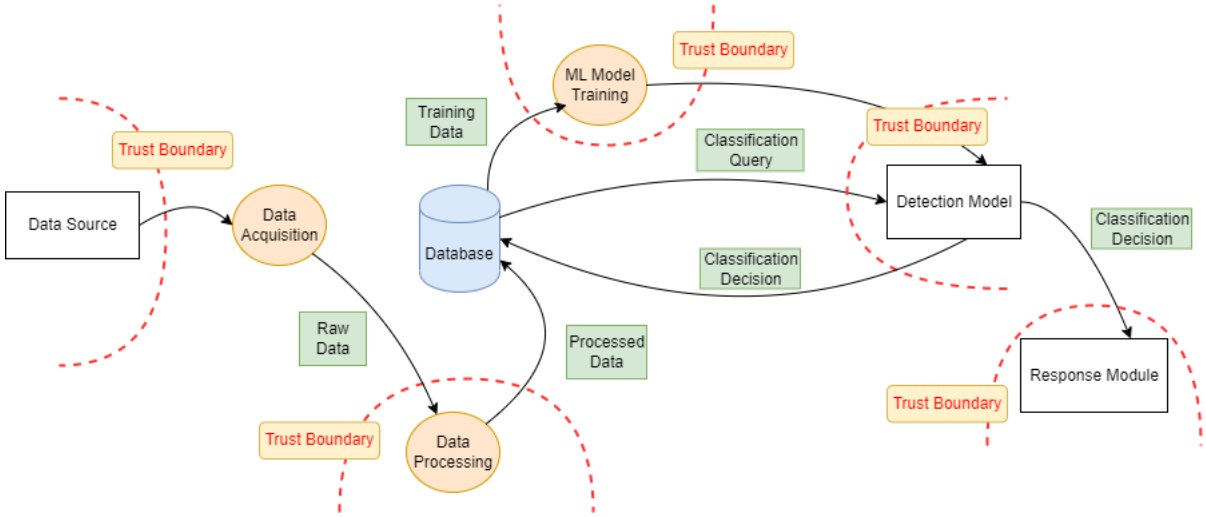

Figure 3.3 DFD of ML-based NIDS

3.5.4. Data Flow Diagram

Figure 3.3 represents the DFD of a ML-based NIDS that shows. The DFD consists of five elements: an external entity, data store, data flow, trust boundary, and process. An external entity, depicted as a rectangle, sends/receives data or requests/responses services with the other system components. Data flow, represented as an arrow, illustrates the directional movement of data such as network traffic, input/output data, functional calls, and remote procedure calls between data stores, external entities, and processes. A process, represented as a sphere, handles an incoming data flow and produces an output for a system service or component. The data store, represented as, handles a location to retain the data for later access. A trust boundary, represented in dashed lines, illustrates changes in trust levels or privileges. Here, we consider the machine learning components as processes as they take input data and perform some operations to produce output. We use this DFD to identify potential threats using the STRIDE model.

3.5.5. Threat Assessment

In this section, we conduct a qualitative assessment of potential threats to a ML-based NIDS. The assessment is structured around three critical dimensions: **Vulnerability**, **Potential Impact**, and **Conditions** [16]. Each threat is evaluated based on these factors to provide a comprehensive understanding of the risks involved.

- **Vulnerability:** This refers to the specific weaknesses in the system that could be exploited by an adversary. We identify and categorize vulnerabilities within the system, focusing on areas such as authentication, access controls, and system configuration.

- **Potential Impact:** The potential impact assesses the consequences of a successful exploitation of a vulnerability. This includes the severity of damage to the system's integrity, availability, confidentiality, or any other critical aspect that the attack may affect. The impact is generally classified as low, medium, or high, depending on the extent of the damage.

- **Conditions:** This dimension considers the specific conditions or prerequisites necessary for a threat to be realized. This includes factors such as the presence of certain system configurations, the accessibility of certain network resources, or the need for insider knowledge. Understanding the conditions helps in estimating the likelihood of the threat materializing.

These factors are used to assess the threats identified using the Attack Tree and STRIDE models in the following sections 3.6 and 3.7.

3.6. Attack Tree Model For ML-Based NIDS

In this section, we present our attack tree model for the ML-based NIDS. The attack tree was constructed based on the attack surface identified in Section 3.5.3. Starting from this generic attack tree, ML-based NIDS threats can be investigated, and more complex attack trees can be structured for each of them. To analyze the machine learning detection model, we define the adversary's main goal as to compromise the system in the root node and then his sub-goals which are the different ways of achieving the main goal in the intermediate nodes. Finally, we specify the adversary's actions to realize each sub-goal in the leaf nodes.

In the visual attack tree for ML-based NIDS depicted in Figure 3.4, the structure begins with a root node that represents the overarching attack goal, which is compromising the functionality of the NIDS. This root node is then subdivided into primary branches, each categorizing different attack methods based on their timing relative to the deployment of the model. The branches are divided into two main categories: attacks that occur before model deployment and those that occur after. Attacks occurring before model deployment target the model during its development or training phases, while attacks occurring after model deployment target the model once it is operational. Each of these main categories is further broken down into sub-nodes, which detail specific attack types and techniques. For example, under the "Poisoning" branch, there are sub-nodes for "Data Injection," "Label Manipulation," and "Input Manipulation," each representing different ways an adversary might corrupt the training data. Similarly, the "Evasion" branch is split into nodes such as "White-box Evasion", "Black-box Evasion," and "Gray-box Evasion", with further sub-nodes describing various strategies within those categories. This hierarchical structure allows for a clear and organized representation of how different attack methods relate to one another and contribute to the overall goal of compromising the NIDS. Each node in the tree captures the nature of the attack and its potential impact, providing a comprehensive view of possible threats and vulnerabilities within the system.

Our proposed attack tree model is shown in Figure 3.4, and classified 22 attacks generally based on their times of occurrence before or after the model deployment. The risk assessment of the identified threats is provided in Table 3.1. We provide a detailed description of the identified attacks below.

1.1 Poisoning: adversary corrupts the training data or the logic of the model. The adversary may aim to force the model to induce wrong predictions on specific inputs or reduce

its overall performance and the quality of its decisions making it unreliable or unusable after the deployment.

- **1.1.1 Data Injection:** adversary inserts new malicious inputs to alter the data distribution.
- **1.1.2 Manipulation:** adversary alters the existing training data.
  - **1.1.2.1 Label Manipulation:** adversary manipulates the class labels of the original training dataset.
  - **1.1.2.2 Input Manipulation:** adversary manipulates the features values of the original training dataset.
- **1.1.3 Logic corruption:** adversary subverts the way the model algorithm learns making it unable to learn correctly.

1.2 Backdoor: adversary causes targeted misclassifications or accuracy degradation of the model over specific inputs that meet some hidden property.

- **1.2.1 Trigger Backdoor:** adversary embeds hidden behaviors into the model, which only activate and induce misclassification on inputs containing certain triggers while keeping the model function as intended in normal conditions. The adversary may implant the backdoor by manipulating the model parameters directly or injecting poisoned inputs into the training dataset.
- **1.2.2 Triggerless Backdoor:** adversary exploits vulnerabilities in the working mechanism of the model. For instance, dropout layers in neural networks prevent the model from over-fitting; however, the model performs perfectly on the training data but badly on real-world data.

2.1 Evasion: adversary manipulates traffic to force the detection model into an incorrect decision and bypass detection.

- **2.1.1 White-box Evasion:** adversary crafts traffic to evade detection based on complete knowledge about the model (i.e ., parameters, hyperparameters, training data features, architecture, model internals like weights or coefficient values).
- **2.1.2 Black-box Evasion:** adversary has no knowledge about the model nor training data. However, he collects input-output pairs or accesses training data distribution to exploit the detection model [33]. Using the collected input-output pairs, he trains a surrogate model. Then, he uses white-box approaches to craft adversarial inputs over the surrogate model, which then they can transfer over the target detection system [33].
  - **2.1.2.1 Strict Evasion:** adversary collects the input-output pairs from the detection model to train a substitute model.
  - **2.1.2.2 Adaptive Evasion:** adversary accesses the detection model as an oracle, so he can craft adaptive inputs and observes the oracle's outputs.
  - **2.1.2.3 Non-adaptive Evasion:** adversary can access the distribution of training data, and then he trains a local model over data examples from the same distribution.

- **2.1.3 Gray-box Evasion:** adversary has partial knowledge of the model, such as the model parameters, architecture, or training data.

2.2 Model Inversion: adversary attempts to reconstruct the dataset was used in training the targeted model. The adversary maliciously quires the model repeatedly for maximum confidence results to reveal secret data inputs based on the returned outputs. Countermeasures for ML-based NIDS Threats Identified Using STRIDE Model

2.3 Model Extraction: adversary sends malicious queries to the detection model to expose the algorithm's internal details (e.g., parameters, weights, etc.). The adversary infers this information based on the model output class probabilities and predictions with respect to chosen inputs. The adversary may use this information to train a substitute model in order to steal the functionality of the targeted system.

2.4 Inference: adversary probes the detection model with different inputs to reveal secret information about the model by weighting the outputs.

- **2.4.1 Attribute Inference:** adversary infers missing attributes by exploiting public information received from the system. The adversary has partial knowledge of a particular record in the training data and uses the model and the record's incomplete information to infer the missing information.

- **2.4.2 Membership Inference:** adversary infers whether a specific data sample was a part of the model training data or not based on the returned output. The model outputs stronger confidence scores when fed with its training data, as opposed to new and unseen data. The adversary aims to rebuild the records used for model training.

2.5 Model Reprogramming: adversary uses crafted queries to repurpose the detection system and makes it execute unexpected tasks that was not programmed for.

2.6 Trojan: adversary controls the detection system's behavior after malicious modification or distribution of the model that works as expected in normal conditions.

2.7 Model Manipulation: adversary provides falsified data as feedback to the system steering the model improvements in the wrong direction. machine learning models keep learning and improving by taking constant feedback from the environment, such as in reinforcement-based NIDS systems.

2.8 Model Replacement: adversary manipulates the pre-trained model or replaces the genuine model with a malicious one.

2.9 Model Supply Chain: adversary compromises the model after being deployed through vulnerabilities in the system dependencies provided by 3rd parties.

2.10 Model Dependencies: adversary exploits traditional vulnerabilities such as buffer overflow in the model software dependencies to control the system.

2.11 Model Fuzzing: adversary uses an automated process to send random inputs to expose vulnerabilities.

Attack	Description	Vulnerability	Potential Impact	Conditions
Data Injection	Injecting malicious data to corrupt the training process or bias model predictions.	Lack of proper access controls	Corruption of the model's integrity, leading to inaccurate or unreliable predictions.	Requires access to the training data or process. Higher risk in environments with weak data controls.
Label Manipulation	Altering labels in the training dataset to mislead the model.	Lack of proper access controls, data encryption, secure development	Decreases model accuracy by associating incorrect labels with data points.	Requires access to labeled training data. More likely in systems without strong data integrity checks.
Input Manipulation	Manipulating input data to cause the model to make incorrect predictions.	Lack of proper access controls, data encryption, secure development	Causes incorrect predictions during inference, affecting the model's reliability.	Requires control over input data. Higher risk in environments with insecure data handling.
Logic Corruption	Corrupting the logic of the model through manipulation of data or parameters.	Lack of proper access controls, secure development, penetration testing	Leads to erroneous decision-making by the model, undermining its trustworthiness.	Requires deep knowledge or access to the model's logic or training process.
Trigger Backdoor	Embedding hidden triggers that cause the model to behave maliciously when activated.	Lack of proper access controls, secure development, penetration testing	Can cause targeted failures or unauthorized actions when the trigger is activated.	Requires embedding triggers during the training phase or model development.
Triggerless Backdoor	Backdoors without specific triggers that can be activated under certain conditions.	Lack of penetration testing	Allows adversaries to control the model's behavior unpredictably.	Harder to detect, can be embedded during model development or update phases.
White-box Evasion	Crafting inputs that evade detection by fully exploiting model knowledge.	Lack of proper access controls, secure development	Allows adversarial inputs to bypass the model, leading to undetected threats.	Requires full knowledge of the model, feasible in open-source or weakly secured models.
Strict Evasion	Creating adversarial inputs under strict constraints to avoid detection.	Lack of proper query controls	Enables the evasion of detection systems, leading to security breaches.	Requires an understanding of model constraints and behavior.
Adaptive Evasion	Modifying inputs based on feedback to continuously evade detection.	Lack of proper query controls	Increases the effectiveness of evasion attacks, leading to ongoing security issues.	Requires iterative access to the model's responses, common in dynamic threat environments.
Non-adaptive Evasion	Single-shot evasion without feedback, typically against less adaptive models.	Lack of proper access controls, secure development	Enables initial bypassing of the model but may be less effective long-term.	Requires knowledge of model behavior, more effective against static defenses.
Gray-box Evasion	Crafting inputs with partial knowledge of the model to evade detection.	Lack of proper access controls, secure development	Allows bypassing of security measures while exploiting limited model knowledge.	Feasible when some but not all model details are known to the adversary.
Inversion	Reconstructing sensitive training data by querying the model.	Lack of proper query controls	Potential exposure of confidential information used in training.	Requires repeated access to the model's outputs, more likely in publicly accessible models.
Extraction	Inferring the model's parameters or structure through repeated queries.	Lack of proper query controls	Unauthorized replication or understanding of the model, compromising intellectual property.	Feasible with extensive querying, more likely in poorly secured query interfaces.
Attribute Inference	Predicting missing attributes from the input data based on model outputs.	Lack of proper query controls	Exposure of sensitive or private information through model outputs.	More likely when the model outputs detailed information, like confidence scores.
Membership Inference	Determining if a particular data point was part of the model's training set.	Lack of proper query controls	Breach of data privacy and potential exposure of sensitive information.	Higher risk in models that output detailed or confidence-based responses.
Reprogramming	Reusing the model for unintended tasks, causing unpredictable behavior.	Lack of proper query controls	Leads to misuse of the model for unintended or malicious purposes.	Requires knowledge of input-output mappings, feasible in complex models.
Trojan	Embedding hidden, malicious functionalities in the model that can be activated later.	Lack of proper access controls, secure updating, system monitoring	Causes unauthorized actions when triggered, leading to severe security breaches.	Can be introduced during model development or updates. Hard to detect before activation.
Manipulation	Steering the model's learning process by providing misleading feedback.	Lack of proper query controls, system monitoring	Leads to degraded performance or incorrect learning outcomes, compromising the model's effectiveness.	Feasible in systems relying on feedback loops or continuous learning.
Replacement	Substituting the original model with a malicious one.	Lack of proper access controls, system monitoring	Complete control over the system by the adversary, leading to severe security issues.	Requires high-level access to the model deployment environment.
Supply Chain	Compromising the model or its components through third-party dependencies.	Lack of secure development, penetration testing, patching	Introduces vulnerabilities that can be exploited, potentially affecting the entire system.	More likely in environments with complex dependencies and third-party integrations.
Model Dependencies	Exploiting vulnerabilities in model dependencies or external libraries.	Lack of secure development, penetration testing, patching	Can lead to indirect compromises, affecting the model's overall security.	Feasible in systems with many external dependencies, especially those not regularly updated.
Fuzzing	Randomly testing the model with numerous inputs to find vulnerabilities.	Lack of proper query controls	May expose unexpected weaknesses, leading to future exploits.	More likely in systems with open interfaces and extensive query capabilities.

Table 3.1 Threats Assessment for ML-based NIDS Attack Tree Model

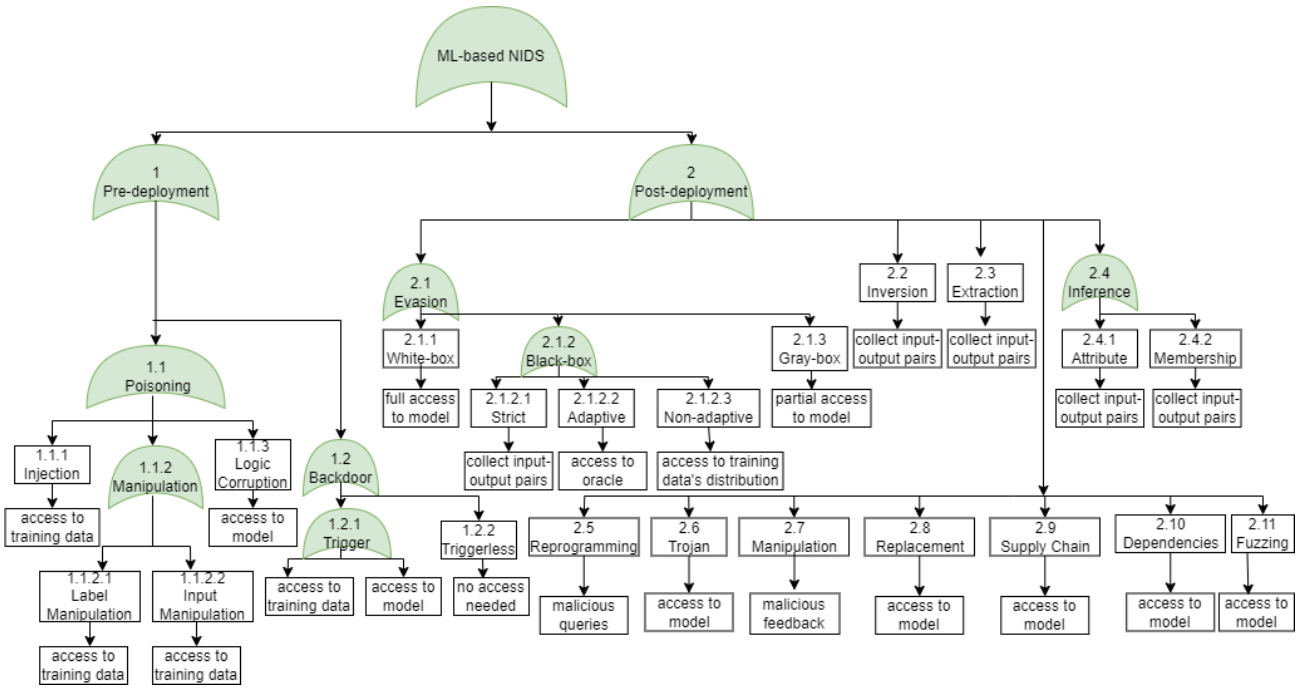

Figure 3.4 ML-based NIDS Attack Tree Threat Model

3.7. STRIDE Model For ML-Based NIDS

In this section, we use the STRIDE model to identify potential threats that target the data flows and the system services. The model was applied to the Data Flow Diagram 3.5.4 to identify security problems in data stores, data flows, and system processes. By utilizing the STRIDE approach, we found that specific threats are more likely to occur for a certain type of element in the data flow diagram. Data stores are more likely to be vulnerable to tampering, denial of service, information disclosure, and repudiation attacks. Data flow threats are likely related to tampering, information disclosure, and denial of service. The likely threats associated with an endpoint or an external entity are spoofing, repudiation, and elevation of privileges attacks. Lastly, a process is susceptible to all threats of spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges. Table 3.2 shows the identification of threats that we conducted using the STRIDE threat model. The threats are grouped according to the categories defined in the STRIDE model. Each threat is assessed in terms of its impact, severity, and probability as described in 3.5.5. The majority of potential threats are tampering (7), followed by spoofing (5), information discourse (4), and denial of service (4). The least number of threats were repudiation (2) and elevation of privileges (2). Furthermore, Table 3.3 illustrates possible mitigation solutions for each group of the identified threats.

Category	Threat	Vulnerability	Potential Impact	Conditions
Spoofing	Adversary impersonates an authorized source to request or response to system services	Lack of proper authentication	Service disruption, data manipulation, unauthorized access	Requires access to system credentials or network
	Adversary impersonates an authorized source to create, update, delete data	Lack of proper authentication	Data integrity compromise, potential data loss	Exploitation of insufficient authentication mechanisms
	Adversary impersonates an authorized source to update or delete the system configuration	Lack of proper authentication	System malfunction, increased vulnerability	Weak configuration controls or lack of logging mechanisms
	Adversary impersonates an authorized source to send or receive data	Lack of proper authentication	Data theft, loss of confidentiality	Insufficient encryption or access controls
	Adversary uses false credentials to gain admin access to the system or data	Lack of proper authentication	Total system compromise, data breach	Weak authentication, poor password management
Tampering	Adversary alters the data on the flow or at rest	Lack of proper access controls, encryption	Data corruption, loss of integrity	Lack of data integrity checks or encryption
	Adversary alters the system configuration	Lack of proper access controls, encryption	System malfunction, weakened security	Poor configuration management or lack of encryption
	Adversary alters the system detection decision outputs	Lack of proper access controls	Reduced detection accuracy, misleading system responses	Poor access control mechanisms or insufficient logging
	Adversary deletes data from the database	Lack of proper access controls	Loss of critical information, service disruption	Inadequate backup systems, weak access controls
	Adversary inserts malicious data into the database	Lack of proper access controls	Corrupted data, false system outputs	Lack of validation checks or poor access controls
	Adversary alters, creates, or deletes the system services	Lack of proper access controls	Service disruption, compromised system integrity	Poor system monitoring, insufficient logging
	Adversary alters the system/data/services access controls	Lack of proper access controls	Unauthorized access, data breach	Poor access control mechanisms or lack of encryption
Repudiation	Adversary disables logging services	Lack of proper access controls, security monitoring	Loss of audit trail, difficulty in tracing activities	Weak logging mechanisms, poor access controls
	Adversary deletes or alters the system/data/services audit trails and access logs	Lack of proper access controls, encryption, security monitoring	Loss of audit trail, difficulty in forensic analysis	Poor access controls, insufficient encryption
Information Disclosure	Adversary intercepts and relays on-flow data	Lack of proper access controls, encryption	Loss of confidentiality, data breach	Insufficient encryption, weak access controls
	Adversary accesses system/data/services meta-data	Lack of proper access controls, encryption	Exposure of sensitive information, loss of confidentiality	Poor access controls, insufficient encryption
	Adversary eavesdrops on a connection between two sources	Lack of proper access controls, encryption	Loss of confidentiality, data breach	Insufficient encryption, weak access controls
	Adversary accesses data at rest	Lack of proper access controls, encryption	Loss of confidentiality, data breach	Insufficient encryption, weak access controls
Denial of Service	Adversary floods the system's sources with massive requests	Lack of proper configuration of access control lists, load balancing	Service disruption, reduced system performance	Weak load balancing, poor access control configurations
	Adversary blocks services from accessing the system resources	Lack of proper configuration of access control lists, hardening measures	Service disruption, reduced system availability	Poor configuration of access controls, insufficient hardening
	Adversary crashes or destabilizes the system's operations by exploiting bugs	Lack of system patching	System downtime, potential system failure	Unpatched vulnerabilities, lack of system monitoring
	Adversary occupies the communication channel bandwidth	Lack of proper configuration of access control lists, congestion control	Service disruption, reduced communication efficiency	Poor access control configurations, weak congestion control
Elevation of Privilege	Adversary gains unauthorized access and escalates it to administrator access	Lack of proper access controls, security monitoring and logging	Total system compromise, data breach	Weak access controls, insufficient monitoring
	Adversary exploits flaws in the system to elevate access privileges (e.g., buffer overflow, RootKit, or backdoors)	Lack of system patching, security monitoring and logging	Total system compromise, unauthorized access	Unpatched vulnerabilities, insufficient logging and monitoring

Table 3.2 Threats Assessment for ML-based NIDS Using STRIDE Model

Threats	Possible Mitigation
Spoofing	Secure authentication and identification mechanisms, secure password requirements, multi-factor authentication (MFA), digital signatures, certificates, and strong encryption mechanisms.
Tampering	Digital signatures, cryptographic hashing, isolation and access checks, permissions, access controls, integrity checks, audit logging for accessing data, software integrity diagnosis, data integrity diagnosis, data inputs validation, outputs encoding, and static code analysis.
Repudiation	Secure authentication, secure logging and auditing, digital signatures, log file analysis, and digital forensics technology.
Information Disclosure	Sensitive data encoding techniques include hashing and encryption, isolation, permissions, proper access controls, and authorization.
Denial of Service	Load balance, congestion control, redundancy, failover, QoS, bandwidth throttle, hardening and limiting allowed requests, limiting the amount of resources needed for handling a not-authorized connection, firewalls, and malware detection methods.
Elevation of Privilege	Least privilege authorization, antivirus, system patching and updating, auditing, strong password policies and enforcement, multi-factor authentication, user inputs sensitization, privileged account protections, access controls, group or role membership, privilege ownership, managing identity life cycle, hardening system components through configuration changes, closing ports, removing unnecessary access.

Table 3.3 Countermeasures for ML-based NIDS STRIDE Model Threats

3.8. Discussion

We implemented threat modeling to identify the security threats of employing machine learning technologies for network anomaly detection. 24 threats against the ML-based NIDS framework were identified using the STRIDE threat modeling. Overall, the more common identified threats were tampering, spoofing, information disclosure, and denial of service, each of which can be exploited to significantly endanger the ML-based NIDS. The least common threats were repudiation and elevation of privilege. An attack tree was implemented for possible threats associated with machine learning vulnerabilities. The attack tree model revealed 22 possible threats. We demonstrated that an adversary can realize specific malicious goals by exploiting threats at different components in the system. The results of the Attack Tree and STRIDE approaches are meaningful, sufficiently broad, and easily understandable to system designers in order to develop security hardening measures. The suggested countermeasures can help to mitigate some of the identified threats. However, these solutions come with a trade-off in performance. When assessing the security implications of employing machine learning technologies for NIDS, the proposed threat models can be taken as a starting point for a more profound risk assessment. We found that not all threats are relevant or can pose high risks.

Previous research on ML-based NIDS security primarily focused on developing adversarial attacks tailored to these systems, assessing their resilience, or implementing defenses against such attacks [9]. However, in this chapter, we identified 22 threats that target the machine learning detection model using attack tree modeling, and 24 threats by the STRIDE approach.

Threat modeling is crucial for real-world deployment of ML-based NIDS as it helps to reduce risk exposure and validate the effectiveness of security controls.

We developed threat models for ML-based NIDS that offer guidance for resilient real-world deployment. These models aid in identifying threats, designing defensive countermeasures, and securing systems from various security risks. While these models address the current threats, ongoing threat modeling processes will be crucial for identifying emerging threats and designing effective solutions to enhance ML-based NIDS security.

3.9. Conclusion

In this chapter, we explored the potential threats and vulnerabilities associated with employing machine learning approaches in NIDS, and devise countermeasures to enhance their security and trustworthiness. We provided a comprehensive overview of threat modeling for ML-based NIDS using Attack Tree and STRIDE approaches. By systematically identifying, categorizing, and assessing potential security threats, the models offer valuable insights into the vulnerabilities inherent in machine learning algorithms and the risks posed by inadequate security measures. In total, 46 threats were identified, necessitating mitigation for deploying a resilient ML-based NIDS.

Furthermore, we highlighted the practical utility of these threat models by demonstrating their applicability in considering potential attacks on each component of the detection system when developing hardening measures. This approach facilitates a more holistic approach to security enhancement in ML-based NIDS deployments.

Overall, the proposed threat models represent a significant contribution to the field, providing a foundational framework for future research aimed at addressing ML-based NIDS security threats and enhancing risk assessments.

As we have identified 46 potential threats through our comprehensive threat modeling in this chapter, it becomes evident that understanding and addressing evasion attacks in the context of ML-based NIDS is of paramount importance. This chapter has laid the foundation for recognizing the vulnerabilities and risks associated with machine learning algorithms in NIDS. To delve deeper into adversarial evasion attacks and explore potential solutions, we will proceed to the next chapter, where we conduct a systematic review to highlight these attacks in the literature.

Chapter 4. Adversarial Machine Learning in NIDS Domain: A Systematic Review

4.1. Summary

Despite their impressive accuracy of in detecting malicious traffic, ML-based NIDS are susceptible to adversarial attacks which encompass various strategies, with poisoning and evasion being among the most prevalent. Poisoning attacks inject malicious data into the training dataset to compromise the model's integrity, while evasion attacks manipulate input data during inference to bypass detection. Based on our threat modeling conducted in the previous Chapter 3, we have identified evasion attacks as a significant threat for ML-based NIDS. These attacks are highly probable as they don't require direct access to training data or internal privileges. Given that adversaries in network cyber attacks are mostly external agents, evasion tactics are commonly employed. Therefore, in this chapter, we focus on presenting a comprehensive overview of the current state-of-the-art in adversarial evasion attacks against ML-based NIDS. Drawing upon a structured analysis of the literature, we synthesize and categorize the existing body of work into generating tailored adversarial examples for ML-based NIDSs, evaluating ML-based NIDS resilience, and developing defensive mechanisms. We present the existing methodologies, discussing their strengths, limitations, and potential avenues for improvement. Overall, this chapter offers a comprehensive overview and assessment of the body of knowledge, identifies gaps in current research, and provides insights for future works.

4.2. Introduction

Threat of adversarial attacks is increasingly investigated in machine learning models across various domains, such as images and text recognition. However, the specific challenges posed by such attacks in the context of NIDS have not received adequate attention. ML-based NIDS operate in real-time which requires prompt detection of adversarial attacks to prevent security breaches. Failures can lead to severe consequences, such as data breaches and reputational damage which demands a robust and effective defense. Additionally, network environments are dynamic, with evolving threats necessitating adaptive adversarial defense strategies. Furthermore, resource constraints demand lightweight and efficient defense mechanisms for effective operation in ML-based NIDS. The lack of research addressing these challenges in the NIDS domain motivates us to conduct a comprehensive overview to fill this gap.

In this chapter, we explore adversarial evasion attacks against ML-based NIDS. We start with an overview of various techniques for generating adversarial attacks, encompassing a spectrum of approaches ranging from reinforcement learning to generative adversarial networks and genetic algorithms. Subsequently, we pivot to evaluating the resilience of ML-based NIDS against generic adversarial attacks. Then, we explore different defense mechanisms such as adversarial training, ensemble learning, and feature reduction. We analyze the literature methodologies highlighting strengths, and weaknesses. We also provide meta-data analysis to identify research trends and patterns and derive key findings. We tackle relevant research questions concerning the drawbacks of current methodologies and suggestions for enhancements. This review serves as a foundational resource offering insights into the current state of research and opportunities for further advancement. The main research question this chapter attempts to answer:
What adversarial evasion attacks are employed for ML-based NIDS, what mitigation strategies exist, and what are the limitations and research improvement considerations?

Contributions: The contributions of this chapter are as follows:

- **C1:** To provide a comprehensive overview of the sate-of-art of adversarial evasion attacks in the ML-based NIDS domain.
- **C2:** To synthesis, classify and identify key characteristics in the literature.
- **C3:** To present literature methodologies and highlight strengthens weakness, considerations for improvements
- **C4:** To provide meta-analysis, identify research trends and patterns, and derive key findings
- **C5:** To identify research gaps and provide recommendations for future work.

Organization: The rest of this chapter is structured as follows: Sec 4.3, discusses previous related surveys. Sec 4.4, describes the methodology of conducting this review. Sec 4.5 introduces novel techniques for generating adversarial attacks on ML-based NIDS. Sec 4.6 evaluates ML-based NIDS models' resilience to adversarial attacks. Sec 4.7 covers countermeasures for safeguarding ML-based NIDS against adversarial attacks. Sec 4.8, discusses surveyed studies, remarkable findings, and research question answers. Lastly, Sec 4.9 provides a conclusion for this chapter.

4.3. Related Work

In other related works, Buczak and Guven [29] discussed the complexity and challenges of using machine learning and data mining approaches for network intrusion detection; however, the threats of adversarial attacks were outside the scope of their study. Biggio and Roli [28] provided a detailed overview on the evolution of adversarial learning in the domains of computer vision and cybersecurity; nevertheless, this problem in the area of network anomaly detection was not covered. Qiu et al. [129] presented a general review on adversarial machine learning attacks

against a wide range of deep learning-based AI applications with a concise consideration of some security applications such as intrusion detection, malware detection, and cloud security. Ibitoye et al. [71] provided a survey on some of the adversarial attacks against machine learning applications in network security, including intrusion detection, spam filtering, and malware detection. Zhang et al. [179] addressed adversarial attacks as the drawback of applying deep learning solutions into mobile and wireless networking without covering security applications like network intrusion detection. Martins et al. [93] provided a limited review on some researches have applied the concepts of adversarial machine learning into malware and network intrusion detection contexts. Overall, these studies have narrow scopes, domain-specific focuses, and superficial exploration of adversarial attacks in network intrusion detection, highlighting the urgent need for more comprehensive research. Therefore, in this chapter, we conduct a systematic review to provide a thorough summary of existing research, identify gaps and inconsistencies, and facilitate knowledge translation.

4.4. Methodology

This chapter summarizes and investigates the state-of-the-art comprehensively to draw general conclusions and prelude further research. To elaborate this review, we followed the framework proposed by Kitchenham [80] which involves three main stages: planning, conducting, and reporting. A review protocol was established for the planning stage that formulates the following elements: research questions, search strategy, study selection criteria, and data extraction strategy.

4.4.1. Research Questions

This chapter investigates the threat of adversarial examples towards ML-based NIDS and how this problem was addressed in the literature. Accordingly, this chapter tries to answer the following sub-research questions:

- **RQ1.** What are the drawbacks of the proposed techniques for generating adversarial attacks against ML-based NIDS?
- **RQ2.** What are the requirements for developing effective and practical techniques for generating adversarial attacks against ML-based NIDS?

  RQ3. What are the common limitations in assessing ML-based NIDS resilience's to adversarial attacks, and how can be addressed?
- **RQ4.** What are the proposed countermeasures to mitigate the adversarial attacks against ML-based NIDS, their strengthens and limitations?
- **RQ5.** What are the considerations for developing effective defenses against adversarial attacks on ML-based NIDS?

The answers to these questions are provided in the discussion Section 4.8.2.

4.4.2. Search Strategy

We selected the following research keywords "adversarial machine learning" "OR" "Adversarial Attack" "AND" "network intrusion detection". The search on five selected scholarly databases: Scopus, Google Scholar, arXiv, WebofScience, and ResearchGate. It yielded 230 papers.

4.4.3. Inclusion and Exclusion Criteria

The retrieved research results underwent a comprehensive filtering process based on formulated inclusion and exclusion criteria, encompassing all types of publications from workshops, symposiums, conferences, or journals. Papers not written in English, inaccessible, or not addressing adversarial machine learning attacks in the network intrusion detection domain were excluded. Furthermore, papers applying adversarial machine learning in domains such as malware, medical images, 3D, time-series, and logs data were omitted. The main inclusion criteria focused on papers addressing adversarial examples against different DNN architectures designed for NIDS, employing supervised, semi-supervised, or unsupervised approaches commonly implemented for NIDS, while excluding reinforcement learning-based ones. Additionally, studies needed to address the problem in traditional networks, IoT, SDN, or WSN networks, excluding other types of networks such as industrial control systems (ICS) or cyber-physical systems (CSPs). In total, 112 papers were included as primary studies, which were then analyzed and categorized based on their application of adversarial machine learning into three groups in subsequent sections. 4.5, 4.6, and 4.7.

4.4.4. Data Extraction

To address the research questions, we conducted a data extraction procedure. This involved identifying a list of characteristics to extract from each study, aiming to gather relevant information for addressing our research questions and identifying classification criteria for categorizing the literature. The description of each characteristic is provided in the table 4.1.

Characteristic	**Description**
Year	Year of the publication.
Dataset	Dataset used for experimental evaluation.
Model	ML algorithms used for experimental evaluation.
Defense	Countermeasures proposed/used against adversarial attacks.
Environment	Type of network traffic (traditional, SDN, wireless, IoT).
Setting	Assumed adversary knowledge (white/black/grey-box).
Attack	Method for generating adversarial examples.
Results	Outcomes of the experimental evaluation.
Purpose	Objective of the study (e.g., propose new adversarial attacks, evaluate ML-based NIDS resilience, introduce new defense or assess the existing mechanisms.

Table 4.1 Literature Classification Characteristics

4.5. Generating Adversarial Attacks for ML-based NIDS

In this section, we explore the creation of adversarial attacks targeting and tailoring for ML-based NIDS. We provide an overview of various techniques used to generate these adversarial examples, we categorize these approaches into Reinforcement Learning Attacks, Generative Adversarial Networks Attacks, Surrogate Model Attacks, Genetic Algorithms Attacks, Constrained Generic Attacks, Certain Features Manipulation Attacks, and combinations of these approaches.

4.5.1. Reinforcement Learning Attacks

Several studies have introduced reinforcement learning RL-based frameworks for generating adversarial network traffic. Hore et al. [68] developed Deep PackGen, achieving a 66.4 success rate in disguising malicious packets as benign ones using deep reinforcement learning. Apruzzese et al. [23] proposed an RL-based framework employing 2DQN and Sarsa algorithms to generate realistic adversarial examples, focusing on small perturbations to achieve high evasion rates with limited queries Wu et al. [171] proposed a RL-based framework automatically adds perturbations, updating adversarial traces based on feedback from the detector and an action sequence modifying spatial and temporal traffic features.

Tan et al. [153] proposed evasion technique involves a Kalman filter-based algorithm, predefined packet mutation operators, and Strength Enhanced Deep Q-learning (SE-DQN) for operator selection, but it's only applicable to non-payload NIDSs. Wang et al. [165] introduced a framework involves the agent generating adversarial flow by interacting with the detection model, learning optimal perturbation strategies within a 14-action space to modify transport layer properties without impacting the application layer's malicious function.

While these RL-based frameworks demonstrate high evasion rates against detection models in controlled settings, their real-world effectiveness is less clear. RL-based approaches often require significant computational resources for training and execution. This could limit their practicality for widespread use by attackers, especially against high-throughput network environments.

Many RL-based attacks assume the adversary has full knowledge of the target detection model and access to its outputs for feedback. In real-world scenarios, attackers may not have such privileged access. This limitation makes it more difficult for them to create effective adversarial traffic.

4.5.2. Generative Adversarial Networks Attacks

Usama et al. [158] utilized GAN to generate adversarial examples, altering only non-functional properties of the traffic, and employed adversarial training to enhance models resilience. Hassan et al. [65] utilized tabular GANs to synthesize adversarial DDoS samples to evaluate models' resilience. Chauhan and Heydari [35] proposed using GANs to generate polymorphic DDoS attacks, updating attack profile features and merging them with previous adversarial examples to evade NIDS detection while maintaining a low false-positive rate.

Chen et al. [37] proposed an Anti-Intrusion Detection AutoEncoder (AIDAE) framework generates adversarial examples by learning the distribution of normal features and generating new random features to bypass detection without targeted IDS feedback. It considers the correlation between discrete and continuous features and utilizes both an autoencoder and a GAN for crafting adversarial examples. Shieh et al. [144] utilized Wasserstein Generative Adversarial Networks with Gradient Penalty (GP-WGAN) to generate DDoS adversarial traffic, and introduced an adversarial GAN intrusion detection system (AG-IDS) with dual discriminators for adversrail flow detection. . Cheng et al. [40] proposed Attack-GAN generates adversarial traffic at the packet level using Sequence Generative Adversarial Nets with policy gradient, but it's limited to generating fixed-length packets.

Zolbayar et al. [183] proposed a GAN-based algorithm for generating practical adversarial flow, learning the model's decision behavior without internal knowledge, and introducing active learning to enhance success rates and reduce training examples in the framework. Lin et al. [87] introduced IDSGAN, a Wasserstein GAN-based framework that generates adversarial traffic while preserving the distribution of the original traffic. Functional features representing basic attack characteristics remain unchanged, while non-functional features are modified. Yan et al. [173] proposed DOS-WGAN, which uses Wasserstein GANs to synthesize DoS traffic characteristics from benign traffic probability distribution.

Han et al. [62] proposed approach automatically mutates original traffic using GAN to generate adversarial examples for non-payload features and PSO to approximate these examples to the misclassification boundary. Shu et al. [146] ntroduced a Generative Adversarial Active Learning (Gen-AAL) algorithm, which combines GANs with active learning to generate adversarial attacks, requiring only a limited number of queries to the targeted model for labeled instances.

GANs exhibit remarkable flexibility and versatility in generating diverse and realistic adversarial examples tailored to different attack scenarios and objectives. They can also facilitate adaptive adversarial training which enhances model resilience against attacks. However, GANs' effectiveness is heavily dependent on large, high-quality datasets. Limited or biased data may yield unrealistic or ineffective adversarial examples, and reduce attack success rates.

Moreover, GANs training is resource-intensive, requiring powerful hardware and significant time investment. This can hinder their scalability and practicality in real-time or large-scale systems. Additionally, adversarial examples crafted by GANs often lack transferability across different models because of model-specific features and dataset differences. This limits their generalizability and applicability across varied architectures and training datasets.

4.5.3. Surrogate Model Attacks

He et al. [67] proposed the Liuer Mihou attack, a packet-based adversarial evasion technique, which generates attacks against a surrogate model using predefined mutation operations. The results indicate that existing feature-level defensive mechanisms like Feature Squeezing and Mag-

Net are ineffective in defending against this attack. Qiu et al. [128] employs model extraction to replicate the targeted model, saliency maps to identify significant features, and gradient-based methods like FGSM to generate adversarial examples.. Guo et al. [61] proposed a method that trains a substitute model with a similar decision boundary to the target model and extends the BIM to generate adversarial traffic based on it.

Surrogate model attacks involve replicating the behavior of a target ML-based NIDS by extracting its parameters or behavior and creating a surrogate model. Adversarial examples are subsequently generated based on this surrogate model. Despite advantages like reduced resource demands and stealthiness, their effectiveness heavily relies on the availability of diverse training data and assumes adversaries possess detailed knowledge about the target NIDS. These factors pose limitations on the attack's success rate and practicality in real-world scenarios where such information may not be readily accessible.

4.5.4. Genetic Algorithms Attacks

Mogg et al. [101] employed GA to generate optimal attack features from benign samples mimicking real attacks and using the model's output as feedback to compute fitness.

Genetic algorithms hold promise for crafting adversarial examples against ML-based NIDS by efficiently exploring solution spaces. However, their computational demands and potential lack of generalization impact their practicality and effectiveness in real-world deployment scenarios.

4.5.5. Constrained Generic Attacks

Mohammadian et al. [102] employed Jacobian Saliency Map to identify significant features and perturbation magnitudes for generating adversarial examples. Chernikova and Oprea [41] introduced a framework called FENCE (Feasible Evasion Attacks on Neural Networks in Constrained Environments) for generating adversarial examples in constrained domains with linear or non-linear feature dependencies, utilizing gradient-based methods and mathematical constraints to ensure compliance. However, it assumes a white-box scenario where both the feature set and detection model are known. Simonetto et al. [148] introduced two methods: Constrained Projected Descent (C-PGD), which integrates differentiable constraints into the loss function to guide PGD optimization, and Multi-Objective Evolutionary Adversarial Attack, employing a multi-objective search strategy to optimize perturbation distance, misclassification, and constraint fulfillment simultaneously. Despite ensuring compliance with domain constraints, these methods are feature-space attacks. Wang et al. [168] proposed the Constraint-Iteration Fast Gradient Sign Method (CIFGSM), capable of addressing network traffic complexity, feature correlations, and diverse feature types.

Teuffenbach et al. [155] proposed an approach that organizes features into groups, assigning weights to each group to indicate modifiability. They utilize the CW attack to optimize perturbation and feature weights based on feature space and perturbation budgets, restricting perturbations to accessible and independent features. Gómez et al. [57] proposed the Selective

and Iterative Gradient Sign Method (SIGSM), a method that selectively manipulates features in adversarial attack generation. Yang et al. [174] introduced a white-box adversarial attack method, universal adversarial sample generator (U-ASG), for autoencoder-based semi-supervised network anomaly detection, targeting changeable features with an optimization-based approach.

These attacks often rely on white-box scenarios, assuming complete access to model information. This limits their applicability in real-world settings. Many approaches involve computationally intensive processes and may struggle with generalization across diverse datasets and network architectures.

4.5.6. Certain Features Manipulation Attacks

Apruzzese and Colajanni [24] generated adversarial network flows by randomly altering four feature. This approach lacks of specificity in the alterations made, potentially limiting effectiveness and efficiency. Aiken and Scott-Hayward [6] introduced Hydra, a tool generating adversarial evasion attacks to bypass TCP-SYN DDoS attack detection, manipulating bidirectional traffic, packet rate, and payload size. However, it's limited to creating adversarial examples for TCP-SYN saturation attacks only.

Khamaiseh et al. [78] proposed a tool for generating adversarial evasion attacks to bypass detection of four saturation attacks (TCP-SYN, TCPSARFU, UDP, and ICMP) by perturbing three traffic features: IPv4 source address change rate, Ethernet source address change rate, and Packet rate. Apruzzese et al. [26] generated adversarial attacks by adding random values to three features: (i.e., exchanged_bytes, duration, and total_packets). Anthi et al. [19] proposed a method to generate adversarial examples for DoS attacks by perturbing top-ranked features from the InfoGain Ratio ranking method, but it's limited to DoS attacks and supervised-based NIDS models. Usama et al. [160] introduced a black-box adversarial attack requiring access to label outputs, utilizing mutual information for crafting adversarial perturbations and substitute model training for launching attacks. This reliance on label outputs for crafting attacks, potentially limiting applicability in real-world scenarios. Usama et al. [159] roposed an adversarial attack leveraging Mutual Information to extract important features from normal and DoS examples, then minimizing the distance between DoS example features using constrained $l1$ norm minimization.

4.5.7. Other Approaches

Alhajjar et al. [15] investigated using PSO, GA, and GANs to generate adversarial examples for network traffic. Sun et al. Sun et al. proposed the transFerable Adversarial Traffic (FAT), a black-box transferable adversarial traffic generation method, evading ML-based encrypted traffic classifiers by building proxy classifiers mimicking the target model and translating adversarial features into traffic. Zhang et al. [181] proposed BFAM, a Brute-force Black-box Method to Attack Machine Learning-Based systems, which operates in a black-box setup, requiring only

the model outputs and confidence scores. BFAM iteratively adds perturbations to features until the confidence score for the target label increases.

Peng et al. [121] proposed an enhanced boundary-based approach to generate adversarial DoS examples, perturbing both continuous and discrete features to optimize Mahalanobis distance and considering DoS traffic characteristics. Operating in a black-box setting, it utilizes query outputs and optimization methods. Abusnaina et al. [4] proposed FlowMerge, which merges a representative mask flow from a selected target class with the features of the original flow, using averaging or accumulating techniques. However, FlowMerge's testing was limited to a CNN model, and it assumes complete knowledge of the model and traffic features. Vitorino et al. [162] presented A2PM for generating targeted and untargeted adversarial examples in a grey-box setting, but it requires knowledge of the complete feature set and only preserves constraints on numeric and categorical features, omitting complex constraints on single and correlated multiple feature values.

Sharon et al. [143] introduced TANTRA, a timing-based adversarial network traffic reshaping attack that modifies malicious traffic using timestamp attributes to evade detection without altering the packet's content. Zhang et al. [180] proposed NIDSFM to generate adversarial examples preserving the latent spatial features of benign examples while maintaining malicious functionality. Kuppa et al. [83] proposed a black-box attack against DL-based anomaly detectors using manifold approximation and spherical adversarial subspaces, suitable for threshold-based detectors with unclear anomaly and normal class boundaries. Venturi et al. [161] introduced a time-based adversarial attack on sequential binary detectors, inserting delays in malicious samples to alter their arrangement in the test set, aiming to assess the model's dependence on temporal information for detection.

Some of these methods have limitations, such as being restricted to specific types of attacks like TCP-SYN saturation or certain DoS attacks. Additionally, techniques relying on specific knowledge about the target system or labeled data may not always be practical in real-world scenarios. Moreover, there's no guarantee of evasion success as detection systems may adapt, leading to an ongoing struggle between attackers and defenders. Furthermore, black-box attacks relying on substitute model training may encounter issues with the transferability of adversarial examples across different models or real-world environments.

Ref	Year	Network	Dataset	Setup	Attack	Model	Defense
[153]	2022	Traditional	CICIDS2018	GB	RL [153], FGSM, GAN, Random Policy	MLP, Adaboost, DT, LR, RF	DTA [153]
[23]	2021	Traditional,	CTU-13, UNB Botnet	GB	RL	RF, WnD	AT
[171]	2019	Traditional	CTU-13	BB	RL	DT, CNN	N/A
[68]	2023	Traditional	CICIDS2017-18	BB	RL	DT, RF, MLP, DNN, SVM	N/A
[165]	2021	Traditional,	CTU-13, ISOT	BB	RL	CNN-LSTM, XGBoost	N/A
[35]	2020	Traditional	CICIDS2017	BB	GAN [35]	DT, LR, NB, RF	AT
[65]	2022	Traditional	CICIDS2017	WB	tabular-GAN	RF, DT, LR, NB	N/A
[37]	2020	Traditional	CICIDS2017, NSL-KDD, UNSW-NB15	WB	AE, GAN	Adaboost, CNN, DT, KNN, LR, LSTM, RF	N/A
[144]	2021	Traditional	NSL-KDD	WB	WGAN-GP [144]	KNN, MLP, RF	GAN-AT
[40]	2021	Traditional	CTU-13	BB	Seq-GAN	DT, LR, MLP, SVM	N/A
[183]	2022	Traditional	NSL-KDD, CICIDS2017	WB, BB	GAN, Active Learning	DNN, LR, SVM, KNN, DT	N/A
[87]	2018	Traditional	NSL-KDD	BB	WGAN	DT, KNN, LR, MLP, NB, RF, SVM	N/A
[173]	2019	Traditional	KDDCup99	BB	WGAN	CNN	N/A
[62]	2020	Traditional, IoT	Kitsune, CICIDS2017	BB, GB	GAN, PSO	Kitsune, MLP, LR, DT, SVM, IF	FR
[158]	2019	Traditional	KDDCup99	BB	GAN	DNN, GAN, KNN, RF, SVM	GAN-AT
[146]	2020	Traditional	CICIDS2017	BB	GAN, Active Learning	GB-DT	N/A
[67]	2022	IoT	Kitsune, [67]	GB	Surrogate Model, Mutation Operations	Kitsune, SOM, RRCF, LOF, OCSVM, IF, Elliptical Envelope	Feature Squeezing, Mag-Net
[128]	2020	IoT	Kitsune	BB	IFGSM, Substitute Model, Saliency Maps	Kitsune	N/A
[61]	2021	Traditional	CICIDS2018, KDDCup99	BB	BIM, Substitute Model	CNN, KNN, MLP, SVM, Residual Network	N/A
[101]	2021	Traditional	NSL-KDD	BB	GA	DT	N/A
[148]	2021	Traditional	CTU-13	WB, GB	C-PGD, MoEvA2	MLP, RF	AT
[168]	2020	Traditional	NSL-KDD	WB	CIFGSM, IFGSM	DT, CNN, MLP	N/A
[155]	2020	Traditional	CICIDS2017, NSL-KDD	WB	Optimized- CW	AE, DBN, DNN	N/A
[57]	2021	ICS	Electra	WB	Selective and Iterative Gradient Sign Method	Dense Neural Network	N/A
[41]	2019	Traditional	CTU-13	WB	Gradient-based iteration	DNNs	N/A
[174]	2020	Traditional	KDDCup99	WB	U-ASG	Deterministic AE, VAE	N/A
[6]	2019	SDN	CICIDS2017, DARPA	BB	[6]	RF, SVM, LR, KNN	N/A
[78]	2020	SDN	simulated	BB	[78]	DNN, IF, KNN, NB, SVM	N/A
[24]	2018	Traditional	CTU-13, CICIDS2017, CICIDS2018, UNB Botnet	GB	[24]	RF, DT, AdaBoost, MLP, KNN, GB, LR, SVM	FR
[26]	2019	Traditional	CTU-13	BB	[26]	KNN, MLP, RF	AT, FR
[19]	2021	IoT	Testbed	WB	InfoGain Ratio [19]	RF, SVM, J48, DT, Bayesian Network	AT
[160]	2019	Tor	UNB-CIC Tor	BB	MI, Substitute Model	DNN, SVM	N/A
[159]	2019	Traditional	NSL-KDD	WB	MI, [159]	DNN, SVM	N/A
[15]	2020	Traditional	NSL-KDD, CICIDS2017	WB	PSO, GA, GAN	SVM, DT, NB, KNN, RF, MLP, GB, LR, LDA, QDA, BAG	N/A

Table 4.2 Generating Adversarial Attacks for NIDS Studies

Ref	Year	Network	Dataset	Setup	Attack	Model	Defense
[181]	2020	Traditional	ADFA-LD, DREBIN, NSL-KDD	BB	BFAM	LR, MLP, NB, RF	N/A
[121]	2019	Traditional	CICIDS2017, KDDCup99	BB	Improved Boundary	DNN	N/A
[4]	2019	SDN	[111]	WB	CW, DeepFool, ENM, MIM, PGD, FlowMerge	CNN	AT
[162]	2022	Traditional, IoT	CICIDS2017, IoT-23	GB[FS]	A2PM [162]	MLP, RF	AT
[136]	2021	Traditional	ICSX2016	WB	UAP	One-Dimensional CNN	N/A
[143]	2021	Traditional, IoT	CICIDS2017, Kitsune	BB	TANTRA [143], LSTM	Autoencoder, IF, Kitsune	AT
[180]	2022	Traditional	NSL-KDD, UNSW-NB15, CICDDoS2019	GB	NIDSFM	3L-IDS, LUNET, CNN-BiLSTM, SVM, LR, KNN, RF, DT	N/A
[161]	2022	Traditional	CTU-13	BB	time delays	LSTM	N/A
[83]	2019	Traditional	CICIDS2018	BB	Manifold Approximation Algorithm, spherial local subspaces	AE, BiGAN, DAGMM, GAN, IF, One-Class SVM	N/A
[150]	2023	Traditional	CICIDS2017, MTA	BB	FAT [150]	LR, DT, SVM, MLP, IF, LSTM, FS-Net	N/A
[102]	2023	Traditional	CICIDS2017, CICIDS2018, CICDDoS2019	WB	JSMA [102]	DNN, DT, RF, NB, LR	N/A

Table 4.3 Generating Adversarial Attacks for NIDS Studies

4.6. Evaluating ML-based NIDS Resilience to Adversarial Attacks

In this section, we comprehensively review existing literature on evaluating the resilience of ML-based NIDS, encompassing a range of adversarial attack techniques and their impacts on the NIDS performance.

Yang et al. [175] employed three black-box attacks for generating adversarial examples: one using the CW method on a substitute model similar to the victim model, another utilizing the ZOO method for gradient estimation, and the last employing a WGAN for crafting adversarial examples. Duy et al. [48] investigated the use of various GAN types, including: Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP), WGAN-GP with two timescale update rule (WGAN-GP TTUR), and AdvGAN to generate adversarial examples.

Vitorino et al. [163] assessed classifiers against constrained adversarial examples generated with A2PM. Alshahrani et al. [18] employed GAN-generated adversarial examples for evasion and poisoning attacks on LR and DT models, with DT models proving more vulnerable to evasion and LR models more impacted by poisoning. Merzouk et al. [97] assessed a Deep-RL model's performance against FGSM and BIM attacks in two setups: two-class and multi-class detection, testing both targeted and untargeted versions of the attacks.

Pujari et al. [127] examined the impact of FGSM, JSMA, and CW attacks on model performance, highlighting the influence of dataset feature counts on vulnerability to these attacks. Jmila and Khedher [77] evaluated the resilience of seven shallow ML-based classifiers against various adversarial attacks, finding differing impacts depending on the attack type,

with a trade-off between resilience and overall performance. Additionally, the authors explored using Gaussian data augmentation to enhance model resilience across different classification settings. Talty et al. [152] evaluated the performance of MLP, Classification Tree, RF, KNN, LR, Gradient Boosting models under the FGSM, PGD, and Boundary Attacks using the NSL-KDD dataset.

Kulikov and Platonov [82] assessed an LSTM model against FGSM and JSMA using the UNSW-NB15 dataset. Rashid et al. [132] evaluated the effects of JSMA, FGSM, BIM, DeepFool, and CW attacks on a DNN model, and found that the CW attack had the most significant impact on its performance. Jadidi et al. [73] demonstrated the impact of FGSM attacks on reducing an ANN classifier's performance across two datasets: BoT-IoT for IoT networks and Modbus for large-scale industrial IoT networks.

Mathews et al. [94] compared EN and TextAttack in deceiving an RNN-based DDoS detector, finding TextAttack achieved a 100 success rate, while EN achieved 67.63. Rigaki [133] generated adversarial examples against an MLP substitute model using FGSM and JSMA, then transferred them to attack other models (DT, SVM, RF, and Majority Voting). Results indicated all classifiers were impacted, with linear SVM being most affected and RF most resilient. Warzyński and Kołaczek [170] tested an FNN binary classifier's resilience against FGSM attack, resulting in significant performance degradation where all adversarial instances were misclassified as benign, yielding a False-Negative rate of zero.

Wang [169] assessed an MLP classifier's resilience against FGSM, JSMA, DeepFool, and CW attacks, observing that these attacks effectively reduced the classifier's AUC in the adversarial setting, with FGSM being the most effective and CW the least effective. Seven features targeted by the attacks were identified, suggesting the need for improved protection against exploitation by adversaries. Clements et al. [42] assessed the resilience of Kitsune, a lightweight DL-NIDS for IoT networks, against FGSM, JSMA, CW, and ENM adversarial attacks. Results demonstrated 100 success rates in availability attacks for CW and ENM, and all attacks achieved 100 success rates in integrity attacks.

Peng et al. [123] proposed an ENIDS framework to evaluate ML-based NIDS resilience against four adversarial attacks (PGD, MI-FGSM, L-BFGS, and SPSA), finding varying degrees of resilience among the models, with DNN performing the worst and MI-FGSM achieving the best performance. Ibitoye et al. [72] compared SNN and FNN resilience against FGSM, BIM, and PGD attacks, finding FNN outperforming SNN in accuracy but SNN demonstrating greater resilience. Feature normalization significantly lowered accuracy rates but enhanced resilience to adversarial examples for both models. Jeong et al. [75] evaluated AE and CNN performance against FGSM and JSMA attacks, revealing significant accuracy decreases. Huang et al. [69] assessed three port-scan attack detection models (MLP, CNN, LSTM) against FGSM, JSMA, and JSMA-RE attacks, with JSMA resulting in the most substantial performance decline, ranging from 14 to 42.

Martins et al. [92] evaluated classifiers' resilience against adversarial examples from four attack methods: FGSM, JSMA, DeepFool, and CW. DAE emerged as the most resilient classifier,

while RF exhibited the best performance, with just a 0.1 reduction in AUC between original and manipulated datasets. Piplai et al. [125] evaluated a GAN classifier's resilience against adversarial attacks using FGSM, finding that despite adversarial training, its performance degraded, with the lowest attack success rate reaching 41. This suggests adversarial training may not be effective in this context. Sriram et al. [149] compared ML and DL models for network intrusion detection in adversarial and non-adversarial environments, generating adversarial examples with FGSM and JSMA attacks.

Pacheco and Sun [117] evaluated DT, SVM, and RF classifiers against JSMA, FGSM, and CW attacks, finding FGSM most effective on BoT-IoT and CW most effective on UNSW-NB15. SVM was least resilient on both datasets, while RF was most resilient, and JSMA was least effective on both. Debicha et al. [45] analyzed FGSM, BIM, and PGD attacks on a DNN-NIDS model, finding FGSM reduced accuracy to 14.13, with BIM and PGD further reducing it to 8.85. Adversarial training improved resilience but resulted in a slight accuracy decrease on clean data. Maarouf et al. [89] evaluated encrypted traffic classification models' resilience against ZOO, PGD, and DeepFool attacks. While DL-based models outperformed conventional ML in an adversarial-free setup, their resilience varied depending on the attack type, with DeepFool being the most effective against both model types.

Fu et al. [52] tested model resilience against FGSM adversarial examples, with adversarial training enhancing resilience, particularly for LSTM models. However, this approach may reduce accuracy on normal examples by making decision boundaries more adaptable to adversarial examples, potentially undermining model judgment. Zhong et al. [182] proposed MACGAN, a framework to evaluate and enhance the resilience of ML-based NIDS. It comprises manual analysis of traffic features and a GAN for evading detection models, modifying only perturbable features.

The literature on evaluating ML-based NIDS against adversarial attacks reveals several areas for improvement. Studies often focus on specific attacks or datasets, limiting the generalizability of their findings. Additionally, there's limited exploration of the transferability of adversarial attacks across different datasets.

Ref	Year	Network	Dataset	Setup	Attack	Model	Defense
[48]	2023	SDN	CICIDS2018, InSDN	GB	WGAN-GP, WGAN-GP TTUR, AdvGAN	DT, LR, CNN, MLP, LSTM	AT
[163]	2023	IoT	IoT-23, Bot-IoT	GB	A2PM	RF, XGB, LGBM, IF	AT
[152]	2021	Traditional	NSL-KDD	WB	FGSM, PGD, Boundary	MLP, Classification Tree, RF, KNN, LR, GB	N/A
[82]	2021	Traditional	UNSW-NB15	WB	FGSM, JSMA	LSTM	N/A
[132]	2022	IoT	DS2OS	WB	JSMA, FGSM, BIM, DeepFool, CW	DNN	AT
[73]	2022	IoT	BoT-IoT, ModBus	WB	FGSM	DNN	AT
[77]	2022	Traditional	NSL-KDD, UNSW-NB15	WB, BB	FGSM, BIM, PGD, JSMA, CW, DeepFool, ZOO, Boundary, HSJ	Adaboost, Bagging, GB, LR, DT, RF, SVC, DNN	Gaussian Data Augmentation
[94]	2022	Traditional	KDD DDoS-2019 [142]	BB	EN, TextAttack	RNN	N/A
[133]	2017	Traditional	NSL-KDD	GB	JSMA	DT, MLP, RF, SVM, Majority Voting Ensemble	N/A
[170]	2018	Traditional	NSL-KDD	WB	FGSM	DNN	N/A
[169]	2018	Traditional	NSL-KDD	WB	CW, DeepFool, FGSM, JSMA	MLP	N/A
[175]	2019	Traditional	NSL-KDD	BB	WGAN, Substitute Model, ZOO	DNN	N/A
[42]	2019	IoT	Kitsune	WB	FGSM, JSMA, CW, ENM	Kitsune	N/A
[69]	2019	SDN	Theirs	WB	FGSM, JSMA, JSMA-RE	CNN, LSTM, MLP	N/A
[72]	2019	IoT	BoT-IoT	WB	BIM, FGSM, PGD	DNN, SNN	Feature Normalization
[92]	2019	Traditional	CICIDS2017, NSL-KDD	WB	CW, DeepFool, FGSM, JSMA	DT, DNN, DAE, NB, RF, SVM	AT
[123]	2019	Traditional	NSL-KDD	WB	L-BFGS, MIFGSM, PGD, SPSA	DNN, LR, RF, SVM	N/A
[75]	2019	Traditional	NSL-KDD	WB	FGSM, JSMA	AE, CNN	N/A
[149]	2020	Traditional	NSL-KDD	WB	FGSM, JSMA	Adaboost, CNN, DT, DNN, K-Means, LR, LSTM, NB, RF, SVM	N/A
[182]	2020	Traditional, IoT	CICIDS2017, Kitsune	BB	WGAN	IF, Kitsune, RBM, SAE, SVM	N/A
[117]	2021	Traditional, IoT	BoT-IoT, UNSW-NB15	WB	CW, FGSM, JSMA	DT, RF, SVM	N/A
[45]	2021	Traditional	NSL-KDD	WB	BIM, FGSM, PGD	DNN	AT
[52]	2021	Traditional, IoT	CICIDS2018	WB	FGSM	CNN, LSTM, Gated Recurrent Unit	AT
[89]	2021	Traditional	SCX VPN-NonVPN, NIMS	WB, BB	DeepFool, PGD, ZOO	CNN, DNN, KNN, Recurrent NN, C4.5	N/A
[125]	2020	Traditional	IEEE Big Data2019	WB	FGSM	GAN	AT
[127]	2022	Traditional	BoT-IoT, UNSW-NB15, CI-CIDS2018	WB	FGSM, JSMA, CW	MLP, DT, RF, SVM	N/A
[18]	2022	Traditional	CICIDS2017	WB	GAN	DT, LR	N/A
[97]	2022	Traditional	NSL-KDD	WB	FGSM, BIM	Deep-RL	N/A

Table 4.4 Evaluating NIDS Resilience to Adversarial Attacks Studies

4.7. Defending ML-based NIDS Against Adversarial Attacks

In this section, we explore several defense strategies for enhancing the robustness of ML-based NIDS, encompassing adversarial training, ensemble learning, and feature reduction techniques. Additionally, we address challenges associated with implementing and adapting these defenses.

4.7.1. Adversarial Training

Xiong et al. [172] proposed Adversarial Intrusion Detection Training Framework (AIDTF) includes an attacker model, a defender model, and a black-box trainer module. The attacker model produces adversarial samples to deceive the defender model, which distinguishes between real and fake samples. The generated samples are combined with those from the training set to train the NIDS, achieving a 99 recognition rate for attack samples. Abou Khamis and Matrawy [2] assessed the min-max formulation's efficacy as a defense mechanism by augmenting crafted inputs during model training and tested it against five white-box attacks, resulting in improved model accuracy rates, with an average increase from 39 to 92.

Abdelaty et al. [1] proposed GADoT, an adversarial training approach utilizing a GAN to generate DDoS samples for training. Results indicate that while a high accuracy NIDS may have over 60 undetected malicious flows under adversarial attacks, GADoT reduces this to 1.8 or less. Chaitou et al. [32] investigated the effect of resource allocation increases for GAN models on attackers and defenders, analyzing the training process sensitivity of GANs for adversarial attack generation and detection models to different resource parameters. Experiments assessed evasion capabilities with varying resource levels, including network layers, dataset distribution, and training epochs.

Grierson et al. [60] introduced the min-max method to enhance adversarial training, minimizing the maximum loss to decrease misclassification of adversarial examples while maintaining detection accuracy over clean data. Dyrmishi et al. [49] explored using unrealistic adversarial examples to bolster the detection model's resilience, achieving a high accuracy of 99.70 for botnet detection through adversarial training with PGD samples against those from the FENCE approach [41].

Benzaïd et al. [27] proposed a DL and SDN-based framework for application-layer DDoS self-protection, utilizing MLP for detection and employing adversarial training as a defense mechanism. Novaes et al. [112] introduced a detection and defense system utilizing GAN for DDoS attack detection in SDN, showing superior performance compared to MLP, CNN, and LSTM models.

Nugraha et al. [115] tested MLP and CNN-LSTM DL models with two adversarial datasets and found that including more adversary examples in the training dataset improved MLP model resilience. Peng et al. [122] presented an adversarial sample detector (ASD) mploying a BiGAN, where the generator learns the distribution of normal samples and an adversarial sample detection module calculates reconstruction and discriminator matching errors to detect and remove adversarial samples.

AbouKhamis et al. [3] explored the min-max approach, using the max approach to generate adversarial samples and the min approach as a defense mechanism to optimize the IDS during training. The paper also demonstrates that PCA-based feature reduction can improve the resilience of the detection model.

Some studies utilize different approaches for adversarial training like the min-max formulation and GANs which have shown promise in enhancing ML-NIDS robustness against adversarial

attacks. Other studies utilized multi-layered approaches combining different approaches which may offer comprehensive defense strategies.

However, challenges include the difficulty in generalizing to unseen attacks or real-world environments, computational overhead in complex methods like GANs, and addressing diverse adversarial attacks. Adversarial training remains computationally intensive and time-consuming, often leading to overfitting to specific attacks and reduced performance on clean data.

4.7.2. Ensemble Learning

Nguyen et al. [109] proposed an adversarial attacks detector using LightGBM, positioned before the NIDS to identify adversarial examples. Placed after preprocessing and feature extraction, this detector identifies and discards adversarial samples before they reach the main intrusion detection component. Shu et al. [147] introduced Omni, a method proposing an ensemble of models with hyperparameters distant from the attacker's model to counter non-adaptive, non-targeted, and white-box evasion attacks. However, it is limited to these specific attack types.

Nowroozi et al. [114] proposed SPRITZ-1.5C, an ensemble architecture comprising a two-class model (CNN), two one-class classifiers (autoencoder trained with benign and malicious examples), and a dense model. Apruzzese et al. [22] proposed AppCon, which utilizes ensemble learning and effectively mitigates over 75 of adversarial attacks without performance degradation in non-adversarial settings.

Ensemble learning offers robustness by combining multiple models, each capturing different aspects of the data or vulnerabilities to attacks, thereby enhancing system resilience. However, some of these methods introduce complexity which makes implementation and maintenance challenging. This can potentially hinder scalability.

Additionally, the computational overhead of ensemble methods and additional detection layers may impact real-time performance and scalability.

4.7.3. Feature Reduction

McCarthy et al. [95] investigated the effectiveness of feature selection for enhancing models' resilience, they analyzed the features that discriminate between the original and adversarial traffic. They applied RFEto remove the features with the largest absolute difference under the FGSM attack and retrained the detection model. The evaluation results indicated that the model achieved the highest accuracy rate when most features were used, but it rarely exceeded 0.60 under the attack. However, using feature removal, the model achieved an accuracy rate of 0.86 over the FGSM's adversarial examples with no drop in accuracy over the unperturbed examples.

Ganesan and Sarac [53] proposed using Recursive Feature Removal (RFR) to remove features targeted by adversarial attacks, creating multiple reduced feature sets. An ensemble of ML models, including DNN, SVM, LR, and RF, trained on these sets enhances resilience against

adversarial examples, detecting attacks missed by individual classifiers with complete feature sets.

While removing features that are most susceptible to attacks can enhance resilience, it may also result in the loss of important features that contribute to accurate detection of non-adversarial traffic. This could negatively impact the model's overall performance.

The proposed techniques involve additional computational costs, particularly when applied iteratively or in ensemble approaches. This overhead could be prohibitive in real-time NIDS applications.

4.7.4. Hybrid Approaches

Wang et al. [164] introduced Def-IDS, which combines a multi-class generative adversarial network (MGAN) for crafting mimic examples and a multi-source adversarial retraining (MAT) module integrating adversarial examples from various attacks (FGSM, DeepFool, JSMA, and BIM). Ganesan and Sarac [53] proposed using multiple smaller feature sets to train an ensemble of classifiers, which outperforms a single model trained with the complete set of features.

Randhawa et al. [131] proposed RELEVAGAN utilizes deep reinforcement learning to generate semantic-aware adversarial examples for training the discriminator of a GAN-based botnet detector. This approach accelerates generator convergence by focusing solely on evasive examples within semantic limits, addressing data imbalance, ensuring functionality, and reducing training time. Alahmed et al. [8] used a GAN-based defense to strengthen an RF model against ZOO attacks. Three setups were tested: using all 78 features, Recursive Features Elimination (RFE), and Principal Component Analysis (PCA). The PCA model achieved the highest F1 score of 0.77, while RFE scored the lowest at 0.64.

Schneider et al. [140] employed five resilience metrics to identify the most susceptible features to perturbation by adversarial example generation methods (i.e., PSO, GA, GAN, and PGD). These metrics include feature importance tests, visual analysis, distance measurements, feature dropout, and adversarially trained feature importance comparisons. They also tested the effectiveness of four defense mechanisms including mutable feature removal, adversarial training, ensemble methods, and outlier alarms. Zhang et al. [178] presented TIKI-TAKA, a framework testing three NIDS classifiers against five attack methods. They introduced three defense methods—adversarial training, query detection, and model voting ensembling.

These approaches suffer from the limitation of efficiently balancing defense effectiveness and computational overhead. Additionally, ensemble learning may complicate model interpretation and deployment, while deep reinforcement learning-based methods often demand substantial computational resources.

4.7.5. Other Approaches

Chen et al. [38] utilized a GAN model to create an adversarial dataset, integrating the perturbed samples with normal training data for adversarial training. They then proposed a DBN-LSTM

model for detecting adversarial examples. Khettaf and Bouzar-Benlabiod [79] presented NIDS-DEFEND framework detects decision-based adversarial examples with a two-layer approach. The first layer uses a statistical test to identify adversarial flows, while the second layer employs a model to pinpoint individual adversarial examples. A drawback is that the performance may vary depending on the chosen statistical test and classifier.

Nowroozi et al. [113] proposed a feature randomization approach where the training method of the target network (TN) involves selecting random features from the flatten layer of the source network (SN). They used an SVM model as the target, receiving these random features from the flatten layer of a CNN model as the source. A drawback is that effectiveness of randomization heavily relies on the selection of features and the randomness introduced. Furthermore, the performance might degrade if randomization interferes with learning meaningful representations.

Tcydenova et al. [154] introduced an XAI-based has two phases: initialization and detection. LIME extracts explanations from an SVM model for normal instances during initialization. In the detection phase, if traffic is classified as normal, the input explanation is compared with those from initialization. Yet, this approach is time-consuming and inefficient due to the rarity of attacks compared to normal traffic. This time-consuming nature of explanation extraction might limit scalability. Additionally, reliance on explanation-based detection may not generalize well to diverse attack scenarios.

Hashemi and Keller [64] presented Reconstruction from Partial Observation (RePO) employs denoising autoencoders and merges inputs with various random masks before inputting them into the model. Multiple random masks make the NIDS more resilient to adversarial attacks by introducing non-determinism.

Mohanty et al. [103] a Stacking Ensemble (SE) model combining DT, KNN, and RF predictions for better traffic classification accuracy. They added a two-layer auto-encoder for defense, with a detector to identify adversarial examples and a denoiser to remove noise from those examples. Wang et al. [167] proposed MANDA, a MANifold and Decision boundary-based, which detects adversarial examples by leveraging inconsistencies between manifold evaluation and model inference. It assesses model uncertainty on perturbations, identifying adversarial inputs when there's a disparity between the detection model's classification and manifold evaluation.

Jiang et al. [76] proposed the FGDM framework (Feature Grouping and Multi-model Fusion Detector), which detects adversarial examples by grouping features and combining predictions from multiple models with different structures. It maintains accuracy even without adversarial samples, performing well in both adversarial and non-adversarial scenarios.

Qureshi et al. [130] introduced RNN-ADV, a Random Neural Network-based NIDS trained with ABC algorithm, outperforms DNNs against JSMA attacks.

Debicha et al. [46] employed adversarial training and an ensemble of detection models, discovering that the ensemble exhibited greater resilience against transferable attacks compared to a single model.

Debicha et al. [44] proposed using multiple transfer learning-based adversarial detectors in a parallel NIDS. Each detector handles a portion of the data processed by the IDS. Combining

the decisions of these detectors showed improved detection of adversarial traffic compared to relying on a single detector.

Zakariyya et al. [177] introduced Robust, Effective and Resource Efficient FCNN (REDNN) model designed to withstand adversarial attacks and optimize resource usage, ideal for resource-constrained environments like IoT networks. REDNN balances accuracy with reduced training time and memory requirements.

Pawlicki et al. [120] proposed an adversarial detector using neural activations to spot attacks during inference. They collected neural activations from an ANN trained on part of the CICIDS2017 dataset during testing. These collected neural activations were used to train and test various classifiers.

These defense approaches exhibit weaknesses such as dependency on training data quality, computational overhead, vulnerability to new attacks, complexity, potential accuracy trade-offs, and limited generalization to diverse attacks and models.

Ref	Year	Network	Dataset	Setup	Attack	Model	Defense
[114]	2022	IoT	RIPE-Atlas, N-BaIoT	WB	I-FGSM, FGSM, JSMA, L-BFGS, PGD, BIM, DeepFool, CW	CNN-based Autoencoder	EL
[113]	2022	Traditional	UNSW-NB15	BB	JSMA, PGD, L-BFGS, I-FGSM, FGSM, DeepFool, BIM, CW	CNN, SVM	Feature Randomization
[44]	2022	Traditional	CICIDS2017, NSL-KDD	WB	FGSM, PGD, CW, DeepFool	DL	TAD [44]
[109]	2022	Traditional	CICIDS2017	WB	CW, DeepFool, FGSM, JSMA	DNN	EL (LightGBM)
[8]	2022	Traditional	CICIDS2017	BB	ZOO	RF	GAN-AT with FR
[79]	2022	Traditional	NSL-KDD	BB	FGSM, JSMA, CW2, BIM, DeepFool, Boundary, HSJ	Autoencoder with LSTM cells	Statistical test
[172]	2023	Traditional	NSL-KDD	BB	FGM, FGSM, PGD, JSMA	Adaboost, DNN, DT, GB, KNN, LR, NB, RF, SVM	AT [172]

Table 4.5 Defending NIDS Against Adversarial Attacks Studies

Ref	Year	Network	Dataset	Setup	Attack	Model	Defense
[38]	2023	SDN	CICDDoS2019	WB	GAN	GAN, CNN, LSTM, MLP, DBN-LSTM	DBN-LSTM model
[154]	2021	Traditional	NSL-KDD	WB	PGD	SVM	LIME (XAI approach)
[140]	2021	Traditional	NSL-KDD		PSO, GA, GAN, PGD	DT, RF, LR, LDA, QDA, MLP	Mutable FR, AT, EL, Outlier Alarms
[1]	2021	Traditional	custom-SYN, Scapy-SYN, CICIDS2017, UNB201X		GAN, FGSM, BFP		AT
[32]	2021	Traditional	NSL-KDD		GAN	GAN	AT
[60]	2021	Traditional	NSL-KDD	WB	FGSM, PGD	MLP	Min-Max AT
[177]	2022	IoT	N-BaIoT, Kitsune, WUSTL	WB	FGSM, PGD, Semantic, Random Noise	FCDNN	Robust, Effective and Resource Efficient FCNN (REDNN)
[147]	2022	Traditional	NSL-KDD, CICIDS2017, CICIDS2018	WB	FGSM, BIM, JSMA, DeepFool, CW		EL
[53]	2021	Traditional	KDDCup99, CICIDS, DARPA		Hydra [6]		Ensemble Classifiers with reduced features
[76]	2022	IoT	MedBIoT, IoTID	BB	[76]	LSTM, RNN, DT, FGDM	Feature Grouping and Multi-model fusion Detector
[167]	2022	Traditional	NSL-KDD	WB	FGSM, BIM, CW, [167]	MLP, LR, KNN, BNB, DT, SVM	MANDA [167]
[49]	2022	Botnet	CTU-13		FENCE [41], PGD		AT
[139]	2022	Traditional	NSL-KDD	WB	FGSM, PGD, DeepFool	MLP, DT, RF, linear SVM	AT
[103]	2022	Tor	CIC-Darknet-2020	WB, BB	FGSM, BIM, DeepFool, Boundary	Stacking Ensemble (RF, KNN, DT)	Two-staged Autoencoder
[158]	2019	Traditional	KDDCup99	BB	GAN	DNN, GAN, KNN, RF, SVM	GAN-AT
[178]	2020	Traditional	CICIDS2018	BB	Boundary, HSJA, NES, Opt, Pointwise	CNN, LSTM, MLP	Adversarial Query Detection, AT, EL
[2]	2020	Traditional	NSL-KDD, UNSW-NB15	WB	BIM, CW, DeepFool, FGSM, PGD	DNN, CNN, Recurrent NN	Min-Max Formulation
[3]	2019	Traditional	UNSW-NB15	WB	BCAS, BGAS, dFGSMS, rFGSMS	DNN	Min-Max Formulation, AT, FR
[130]	2020	Traditional	NSL-KDD	WB	JSMA	RNN-ADV trained with ABC, DNN	RNN-ADV trained with ABC
[22]	2020	Traditional	CTU-13	GB	[25]	Adaboost, DT, MLP, RF, WnD	Queries attempts restriction, EL
[64]	2020	Traditional	CICIDS2017	WB	[63]	DAE, Kitsune, BiGAN, DAGMM	RePO
[27]	2020	SDN	CICIDS2017	WB	FGSM	MLP	AT
[120]	2020	Traditional	CICIDS2017	WB	BIM, CW, FGSM, PGD	Adaboost, DNN, RF, SVM, KNN	Neurons Activation
[46]	2021	Traditional	NSL-KDD	WB	FGSM, PGD	DNN, DT, LDA, LR, RF, SVM	EL, Detect & Reject
[53]	2021	SDN	CICIDS2017, DARPA, KDDCup99	WB	Hydra	DNN, LR, RF, SVM	FR, EL
[95]	2021	Traditional	CICIDS2017	WB	FGSM	DNN	FR
[112]	2021	Traditional, SDN	CICIDS2019, Emulated Real SDN environment	WB	GAN	CNN, GAN, LSTM, MLP	GAN-AT
[115]	2021	SDN	Theirs, Emulated Real SDN environment	WB	Tabular GAN, Emulated Adversary SDN dataset	MLP, CNN-LSTM	AT
[164]	2021	Traditional	CICIDS2018	WB	BIM, DeepFool, FGSM, JSMA	DNN	Ensemble Retraining
[122]	2020	Traditional	NSL-KDD	WB	FGSM, MIFGSM, PGD	DNN	BiGAN-AT
[131]	2022	Traditional	CICIDS2017, CICIDS2018, ISCX-2014	WB	RFL	GAN	GAN-AT with RFL
[86]	2022	Traditional	CICIDS2017	WB	PGD, ZOO, FGSM, CW, JSMA	DT, LR, XGB, SVM, DNN, Ensemble of (DT, XGB, SVM), Ensemble of all	EL with AT

Table 4.6 Defending NIDS Against Adversarial Attacks Studies

Ref.	Dataset	Network	Year	Attacks Categories
[157]	KDD CUP 99	Traditional	1999	DoS, Probe, User 2 Root and Remote to User
[116]	NSL-KDD	Traditional	2009	DoS, Probe, User 2 Root and Remote to User
[56]	DARPA	Traditional	2009	DDoS, Malware, Spambots, Scans, Phishing
[54]	CTU-13	Traditional	2011	Botnet
[156]	Kyoto	Traditional	2015	Botnet
[106]	UNSW-NB15	Traditional	2015	Backdoors, Fuzzers, DoS, Generic, Shell code, Reconnaissance, Worms, Exploits, Analysis
[17]	WSN-DS	Wireless	2016	Greyhole, Blackhole, Scheduling, Flooding.
[111]	SDN traffic	SDN	2016	DDoS
[141]	CICIDS2017	Traditional	2017	DoS, DDoS, SSH-Patator, Web, PortScan, FTP-Patator, Bot.
[20]	Mirai	IoT	2017	Botnet
[141]	CICIDS2018	Traditional	2018	Bruteforce Web, DoS, DDoS, Botnet, Infilteration.
[81]	BoT-IoT	IoT	2018	DDoS, DoS, OS Service Scan, Keylogging, Data exfiltration
[99]	Kitsune	IoT	2018	Recon, Man in the Middle, DoS, Botnet Malware
[74]	IEEE BigData cup	Traditional	2019	N/A

Table 4.7 Overview of Used NIDS Datasets in the Literature

4.8. Discussion

In this section, we conduct a meta-data analysis to identify trends and patterns in research, extracting key insights. Subsequently, we address pertinent research questions regarding limitations in existing methodologies and propose suggestions for improvements.

4.8.1. Findings Analysis

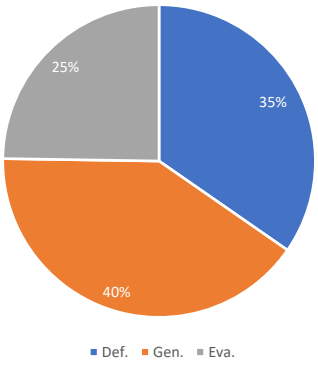

Figure 4.1 Distribution of Studies per Category

Figure 4.1, illustrates the distribution of studies across different categories: 40% focus on introducing novel techniques for generating adversarial attacks against ML-based NIDS, followed by 35% proposing defensive solutions or evaluating existing defense mechanisms for NIDS. Additionally, 25% of the studies are dedicated to assessing the resilience of ML-based NIDS against adversarial examples generated by general approaches.

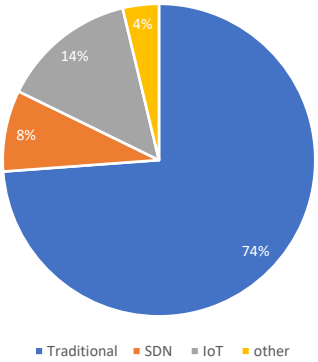

Figure 4.2 Distribution of Studies per Environment

Figure 4.2 reveals that the majority of studies 74% are centered on traditional networks, while fewer 14% address adversarial attacks in IoT networks. Given the increasing prevalence of IoT devices in various contexts like smart homes and buildings, there's a growing need for research to protect these networks from adversarial threats.

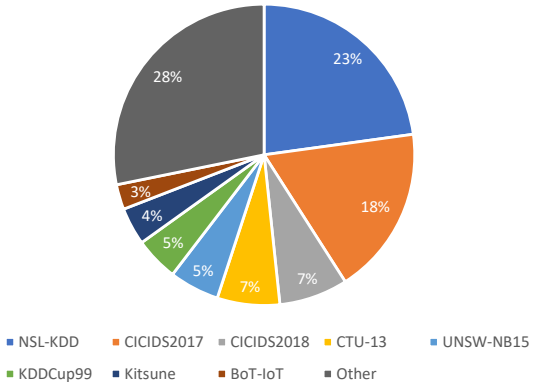

Figure 4.3 Distribution of Studies per Dataset

Figure 4.3, indicates that the NSL-KDD dataset, utilized by 28% of the surveyed studies, is the most popular. However, it's considered outdated and not representative of modern networks. Similarly, the KDDCup99 dataset, used by 5% of the studies, suffers from similar shortcomings, including redundancy issues that bias ML algorithms. Recent datasets reflecting real-world network complexity are needed for evaluating NIDS performance under adversarial attacks. Additionally, standard datasets for benchmarking purposes should be prioritized, as 5% of the studies either used uncommon datasets or emulated their own.

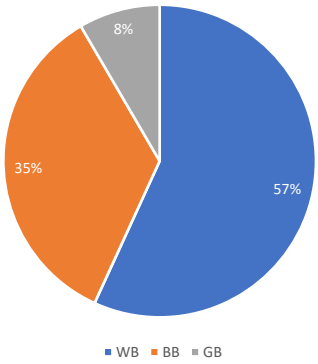

Figure 4.4 Distribution of Studies per Setting

Figure 4.4 demonstrates that a significant portion of the studies focused on white-box attacks 57%. Despite their potential impracticality in real-world scenarios, white-box attacks offer a controlled environment for assessing model vulnerabilities and refining defense strategies. By understanding how easily adversaries can manipulate model behavior with full access to internal parameters and architecture, robustness of ML-based NIDS can be iteratively improved. Conversely, experimental results from black-box studies revealed notable evasion rates, highlighting the substantial potential for adversaries to compromise NIDS without prior knowledge. Given their practicality in real-world scenarios, addressing the threat and defense against black-box attacks becomes imperative. A holistic approach that considers both white-box and black-box attack scenarios should be adapted to develop more resilient ML-based NIDS against adversarial threats in real-world environments.

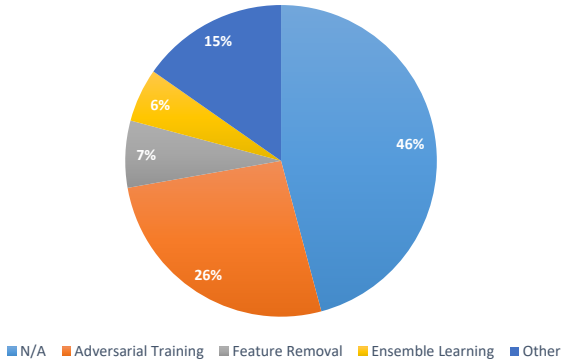

Figure 4.5 Distribution of Studies per Defense Mechanism

As can be seen in Figure 4.5, the proposed solutions, including adversarial training, feature removal, and ensemble learning, represent common approaches to mitigate adversarial attacks in the NIDS domain. However, there is a notable gap in exploring alternative defensive mechanisms from other domains, such as image recognition, which may offer valuable insights and strategies. The unique challenges of NIDS, such as the need for low-overhead methods due to real-time operation, necessitate careful consideration when adapting defensive mechanisms from other domains.

While generic countermeasures exist, they often fall short of addressing the diverse range of adversarial attacks effectively. Therefore, there's a pressing need to develop attack-agnostic defense approaches capable of handling the increasing variety of adversarial threats. This approach contrasts with attack-specific methods like adversarial training, which may only be effective against a narrow range of attacks.

Additionally, the effectiveness of most defensive techniques has primarily been tested on white-box attacks, overlooking challenges posed by black-box adversaries. Comprehensive assessment in both settings is crucial to enhance the applicability and resilience of defensive methods across a wide variety of scenarios.

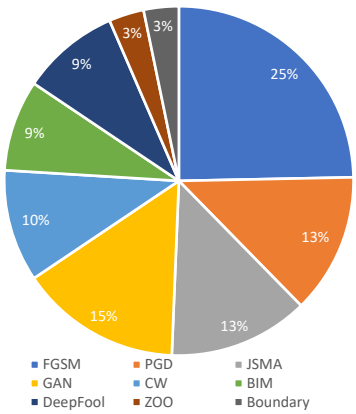

Figure 4.6 Most Utilized Adversarial Attacks

Figure 4.6 shows the most utilized generic approaches for crafting adversarial examples to evade NIDSs. FGSM perturbs input features in the direction of the gradient of the loss function to induce misclassification. Its strength lies in its simplicity and efficiency, making it a popular choice for crafting adversarial samples. However, FGSM attacks may lack robustness against defenses such as gradient masking or input transformation.

PGD and BIM iteratively perturb input features to maximize classification errors. Their strength lies in their versatility and effectiveness across various ML models. However, their iterative nature can be computationally expensive and require knowledge of the target model's architecture, limiting their applicability in real-world scenarios.

GANs generate realistic adversarial samples that are indistinguishable from legitimate traffic making them highly effective in evading ML-based NIDS. However, GANs require significant computational resources which limits their practicality for real-time attacks.

CW formulates the generation of adversarial samples as an optimization problem to minimize perturbation while maximizing misclassification. However, CW attacks are computationally intensive and require access to model gradients. DeepFool crafts adversarial samples by iteratively perturbing input features along the direction of the decision boundary. However, DeepFool attacks require multiple iterations.

ZOO performs black-box attacks by querying the target model without access to gradients. Its strength lies in its ability to bypass gradient-based defenses and adapt to different ML architectures. However, ZOO attacks may require a large number of queries and may be less efficient compared to white-box attacks.

Boundary attack generates adversarial samples by perturbing input features along the decision boundary. Its strength lies in its ability to bypass gradient-based defenses and generate diverse adversarial samples. However, boundary attacks may require extensive computational resources.

These attacks are mainly designed for deep learning models, with PGD, BIM, HSJ, JSMA, and FGSM shown to be the most effective attacks. Although these attacks can transfer through other models, their effectiveness can vary depending on the model's architecture and parameters.

4.8.2. Research Questions

In this section, we answer the research questions identified in Section 4.4.1.

RQ1. What are the drawbacks of the proposed techniques for generating adversarial attacks against ML-based NIDS?

Overall, the effectiveness of RL-based frameworks, GANs, surrogate model attacks, genetic algorithms, and other methods in crafting adversarial examples against ML-based NIDS varies in real-world scenarios due to computational demands, data requirements, and limitations in access to target system information. RL-based approaches require significant resources and assume full knowledge of the target system which limits their practicality. GANs offer flexibility but require large, high-quality datasets and are computationally intensive. Surrogate model attacks reduce resource demands but rely on detailed target system knowledge. Genetic algorithms efficiently explore solution spaces but struggle with generalization and computational demands. Many methods are limited to specific attack types or require detailed target system information. This makes these methods difficult to apply effectively in real-world settings.

The common drawbacks of the proposed techniques can be summarized as follows:

- Target System Knowledge: Most techniques assume the adversary has detailed knowledge of the target NIDS, such as its architecture, parameters, and behavior. This privileged access may not be available in real-world scenarios.

- Computational complexity: Many techniques, like RL-based frameworks, GANs, and genetic algorithms dcan be computationally expensive and time-consuming. This limits their application in real-time processing network environments. In practice, attackers require quick and efficient methods to generate adversarial traffic in real-time scenarios.

- Data Requirements: Techniques like GANs rely heavily on large, high-quality datasets to generate realistic adversarial examples. Limited or biased data can lead to ineffective attacks, reducing success rates. Similarly, surrogate model attacks require diverse training data for accurate replication of the target NIDS's behavior.

- Transferability Issues: Adversarial examples generated by GANs and other techniques may lack transferability across different models due to model-specific features and dataset differences. This limits their generalizability and applicability across varied architectures and training datasets.

- Limited Applicability: Some techniques are restricted to specific types of attacks or require labeled data, which may not always be practical in real-world scenarios. Additionally, white-box attacks relying on complete model information may not be feasible in real-world settings where such information is not readily accessible.

RQ2. What are the requirements for developing effective and practical techniques for generating adversarial attacks against ML-based NIDS?

- Black-box Setup: The attacks should be designed in a black-box setup, operate without detailed information about the ML-based NIDS architecture and parameters. This approach better reflects the limited information attackers typically have. White-box attacks, which assume detailed knowledge of the target NIDS, are often infeasible in real-world scenarios.

- Adaptation to Defense Mechanisms: The attacks must be tested against state-of-the-art defenses to demonstrate their ability to continually achieve evasion success in the presence of mitigation methods.

- Maintain Resources Constraints: The attacks should be efficient to deploy in real-time environments, considering computational resources and time constraints.

RQ3. What are the common limitations in assessing ML-based NIDS resilience's to adversarial attacks, and how can be addressed?

- Lack of Standardized Benchmarks Datasets: The absence of standardized benchmark datasets hinders easy and effective comparisons of NIDS models and attack techniques. Addressing this limitation involves incorporating contemporary and diverse datasets that accurately represent real-world network traffic. These datasets should also encompass a wide array of adversarial attack types to comprehensively test ML-based NIDS resilience. Incorporating standardized benchmarks streamlines evaluation comparisons. This ensures consistent assessments and enhances the reproducibility and generalizability of results.

- Lack of Standardized Evaluation Metrics: The absence of standardized evaluation metrics makes it difficult to comprehensively assess the effectiveness of ML-based NIDS against adversarial attacks. Employing a diverse set of metrics covering model accuracy, attack evasion rates, and transferability of attacks is essential. Defining these as standardized evaluation metrics ensures the reliability and generalizability of assessment results and facilitates easy comparison with other studies.

RQ4. What are the proposed countermeasures to mitigate the adversarial attacks against ML-based NIDS, their strengthens and limitations?

Tailored solutions specifically designed to protect ML-based NIDS against adversarial attacks are currently lacking. Instead, many studies defending NIDS models against adversarial machine learning attacks draw upon mitigation countermeasures from various domains, such as image recognition. These defensive mechanisms include adversarial training, feature reduction, and ensemble learning, each with its own set of strengths and limitations.

Adversarial training enhances models' resilience by exposing them to adversarial examples during training. This process increases their ability to withstand known attacks. However, they may still remain vulnerable to unseen attacks which limits their effectiveness against evolving threats.

Feature reduction aims to mitigate adversarial attacks by simplifying the input space. This reduces the number of potential vulnerabilities. While this can enhance model robustness, it often comes at the cost of decreased detection performance in non-adversarial scenarios. The simplified features may fail to capture relevant information effectively.

Ensemble learning, on the other hand, leverages the diversity of multiple models to improve overall accuracy, particularly on unperturbed examples. However, this approach may not fully address all attack scenarios and can significantly increase computational expenses and memory requirements.

These limitations highlight the necessity for continuous research to develop more comprehensive and robust defenses tailored explicitly for ML-based NIDS. Such defenses should address both known and unforeseen attacks, while also maintaining high performance over clean data and remaining computationally efficient.

RQ5. What are the considerations for developing effective defenses against adversarial attacks on ML-based NIDS?

Considering the following factors enables designing defenses that enhance ML-based NIDS resilience while maintaining high performance and scalability.

- Generalization and Adaptability: Defense approaches should generalize across attack types and adapt to unforeseen and evolving threats. While some strategies may effectively address specific attacks, their limited ability to generalize to diverse attacks undermines their generality and robustness. For example, although techniques like adversarial training prove effective against known threats, they render ML-based NIDS susceptible to unforeseen attacks.

- Computation and Resource Efficacy: Defense approaches should be designed to utilize resources efficiently. Some of the proposed techniques, especially those involving complex models like GANs and deep reinforcement learning, can be computationally intensive and require substantial resources. Implementing these defenses in real-time NIDS systems may be challenging.

- Scalability and Real-time Response: Defense approaches should avoid excessive computational overhead, memory requirements, and significant latency. For example, ensemble methods may introduce delays due to the additional processing time needed for aggregating predictions.

- Preservation of Performance: Defense approaches should maintain high accuracy on legitimate network traffic without introducing degradation in the ML-based NIDS's performance. For example, overly aggressive adversarial training techniques could lead to overfitting on adversarial examples and generate false alarms on benign traffic.

- Comprehensive Evaluation: Defense approaches should be thoroughly evaluated using standardized datasets and metrics across various attack scenarios to validate their effectiveness and reliability.

4.9. Conclusion

In this chapter, we provided a comprehensive overview of the landscape of adversarial evasion attacks against ML-based NIDS. We categorized the body of research into three main areas: generating tailored adversarial examples for ML-based NIDS, evaluating the models' resilience, and developing defensive mechanisms. Within each category, we discussed the proposed methodologies, strengths, limitations, and potential avenues for improvement. This chapter serves as a foundational resource providing an overview and assessment of the existing knowledge, identifying gaps in current research, and offering insights for future works.

In this review, we found that the generic evasion attacks were the most commonly employed techniques to assess the resilience of ML-based NIDS, as observed in Sec 4.8.1. These attacks are mainly designed for unconstrained domains (i.e., image processing), where the adversary can perturb any arbitrary amount of pixels, and the features can be amended independently. However, it's noteworthy that the applicability of these attacks for ML-based NIDS was not investigated in the literature. In the NIDS field, traffic data must adhere to specific domain constraints that restrict the features can be modified. This sets the stage for the next chapter, where we delve deeper into the realism versus performance trade-offs for adversarial examples against DL-based NIDS, and explore the practicality and viability of these attacks in real-world scenarios.

Chapter 5. Realism vs. Performance for Adversarial Examples Against DL-based NIDS

5.1. Summary

In the previous chapter, we found that the existing research utilizes the generic methods for generating adversarial examples to evaluate the resilience of DL-based NIDS against adversarial attacks, Section 4.6. However, these methods were primarily developed for unconstrained domains, such as image recognition, and their suitability for the network traffic domain has not been investigated. In this chapter, we evaluate whether the examples generated by these attacks adhere to network traffic constraints to determine their real threat level and potential impact on the viability of DL-based NIDSs for real-world deployment. We first implement the main adversarial attacks selected from the literature (FGSM, BIM, PGD, NewtonFool, CW, DeepFool, EN, Boundary, HSJ, ZOO) for two different datasets (WSN-DS and BoT-IoT) and we compare their relative performance. We then analyze the perturbations generated by these attacks and use a set of invalidation metrics that imply invalid network traffic, to establish a notion of "attack unrealism". We offer a contrasting analysis of the attacks' performance and realism, and provide a discussion on the practicality and feasibility of these attacks in real-world scenarios. We conclude that, for these datasets, some of these attacks are performant but not realistic.

5.2. Introduction

Recent research uses standard techniques to evaluate DL-based NIDS against adversarial traffic, though these were initially designed for image processing, where adversaries can alter any number of pixels and features independently [9, 10]. For NIDS, the traffic data must preserve some domain constraints that restrict how features can be perturbed, e.g., interdependence between the values of several features, features with fixed values, features with a limited range of values [14]. Therefore, this chapter attempts to answer the following research question:

To what extent do the outputs from generic adversarial attacks align with the characteristics of real network traffic, and how does evaluating the realism and feasibility of these attacks contribute to assessing the suitability of ML-based NIDS for real-world deployments?

We assess the performance of these attacks along with their realism in terms of the compliance of their outputs with the traffic domain constraints. The assessment was conducted for the attacks in two setups, targeted and untargeted, against multi-classification DL-based NIDSs. We use the *Unrealism Index* metric, which is an average of percentages of void adversarial examples and features perturbation to measure the unrealism of the attack.

Contributions: The contributions of this chapter are as follows:

- **C1:** To implement a range of prominent adversarial attacks (e.g., FGSM, BIM, PGD, NewtonFool, CW, DeepFool, EN, Boundary, HSJ, ZOO) in two setups targeted and untargeted on distinct datasets representing different types of network traffic (WSN-DS and BoT-IoT).
- **C2:** To quantify the alterations each attack introduces to the original network traffic and assess their inconsistency with actual network behavior.
- **C3:** To introduce a new Unrealism Index that quantifies "attack unrealism" by evaluating how much the manipulated traffic deviates from genuine network traffic.
- **C4:** To provide an assessment for the attacks' performance vs realism.
- **C5:** To provide a discussion of attacks' validity and feasibility in real-world scenarios.

Organization: The rest of this chapter is structured as follows: Sec 5.3, presents a brief description for network traffic constraints. Sec 5.4 explores the limitations of the previous related works and the gaps we do address. Sec 5.5 describes the setup of our experiments, evaluation metrics, and validation metrics. Sec 5.6 presents the results and analysis. Sec 5.7, provides a discussion of the validity and feasibility of using these attacks in real-world scenarios. Lastly, Sec 5.8 presents our conclusion.

5.3. Network Traffic Constraints

In unconstrained domains such as (e.g., image recognition), the features are independent and can be perturbed arbitrarily. However, network traffic features are constrained by some characteristics such as [14]:

- Every feature can have a continuous, categorical, or binary value.
- The values of some features can be highly interdependent and correlated.
- The values of some features can be constant and unmodifiable.

The binary feature can take either 1 or 0, the categorical feature takes a value that belongs to one category at once, and the numeric feature can only take a value within the allowed range. Additionally, some features may be highly interdependent or fixed and unchangeable. For instance, some features are linearly related, and others are immutable, such as protocol type or connection flag. The adversarial perturbations must maintain the above constraints to generate valid and functional flow.

5.4. Literature Drawbacks

In this section, we discuss some drawbacks of the previous studies that employed generic adversarial evasion techniques to prove their effectiveness in evading and degrading the performance of DL-based NIDS models.

Yang et al. [175] used three black-box attacks—substitute model, WGANs, and ZOO—to mislead a DNN classifier into misclassifying attack traces as normal traffic.

Warzyński and Kołaczek [170] demonstrated that FGSM fully compromised a DNN classifier on the NSL-KDD dataset, confirming its applicability to network traffic. Clements et al. [42] highlighted the vulnerability of Kitsune, a lightweight DL-NIDS, to FGSM, CW, and ENM attacks on the Mirai dataset. Wang [169] compared the effectiveness of FGSM, DeepFool, and CW attacks on an MLP classifier using NSL-KDD.

Peng et al. [123] observed performance drops in DNN, SVM, RF, and LR classifiers against various attacks on NSL-KDD. Ibitoye et al. [72] found that while DNNs were more accurate, SNNs were more resilient to FGSM, BIM, and PGD attacks on BoT-IoT. Jeong et al. [75] evaluated Autoencoder and CNN resilience to FGSM on NSL-KDD.

Huang et al. [69] assessed the impact of FGSM on MLP, CNN, and LSTM models for SDN environments. Martins et al. [92] reported performance deterioration in DT, RF, SVM, NB, NN, and DAE classifiers under FGSM, DeepFool, and CW attacks. Sriram et al. [149] analyzed the effect of FGSM on various classifiers using NSL-KDD.

Piplai et al. [125] used GANs defensively against adversarial examples and to address class imbalance, but found them defeated by FGSM. Debicha et al. [45] reported significant performance degradation in DNN models under FGSM, BIM, and PGD attacks. Maarouf et al. [89] concluded that DL models are generally more robust to adversarial examples than traditional ML models.

Pacheco and Sun [117] confirmed that FGSM and CW attacks reduced the performance of DT, SVM, and RF classifiers on BoT-IoT and UNSW-NB15 datasets. Fu et al. [52] evaluated CNN, LSTM, and GRU robustness to FGSM on CICIDS2018. Merzouk et al. [96] focused on FGSM, BIM, DeepFool, and CW attacks against a binary MLP classifier using NSL-KDD, though their scope was limited to a few white-box attacks.

The previous studies focused on compromising DL-based NIDS using generic evasion adversarial attacks and have stated the vulnerability of detection models to these attacks and regarded them as significant threats. Although these attacks achieve high evasion rates, the studies did not verify the realism or practicality of the generated adversarial traffic for real-world scenarios.

The literature overall and importantly has three significant flaws. First, they did not consider the necessity of maintaining traffic domain constraints in generating the adversarial flow for preserving the validity and functionality of attack traces. Second, they assume the adversary can freely perturb any amount of features that can break the semantic links among the interdependent features. Lastly, they assume a white-box threat model, where the adversary

has access to the parameters of the targeted model, which is not commonly feasible in real-world scenarios.

To fill this gap, we verified the realism of adversarial flow and its compliance with network domain constraints. Furthermore, we validated the outputs of widely used white-box and black-box evasion attacks in targeted and untargeted setups over two recent traffic datasets representing different contemporary networking contexts. Table 5.1 demonstrates the attacks used by the literature to assess the resilience of a wide spectrum of conventional and DL-based NIDS and whether domain constraints verification was conducted for produced adversarial network traffic or not.

Ref. \ Attacks	White-Box							Black-Box			C. V.?
	BIM	CW	DeepFool	FGSM	NewtonFool	PGD	ZOO	HSJ	EN	Boundary	
[170]				✓							
[169]			✓	✓							
[175]							✓				
[42]		✓		✓					✓		
[69]				✓							
[72]	✓			✓		✓					
[92]		✓	✓	✓							
[123]						✓					
[96]	✓	✓	✓	✓							
[75]				✓							
[149]				✓							
[117]		✓		✓							
[45]	✓			✓		✓					
[52]				✓							
[89]			✓			✓	✓				
Ours	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 5.1 Comparison with Related Works

5.5. Experimental Setup

In this section, we provide detailed information about the implementation of the experiments. We describe the used datasets, dataset preprocessing procedure, attack implementation, and targeted model architecture. Then, we explain the used metrics to evaluate the performance of the detection model, and realism of the crafted adversarial traffic by measuring its compliance with network domain constraints.

5.5.1. Datasets

We utilized two recent datasets that represent advanced networking environments a wireless sensor network (WSN-DS) and an IoT network (BoT-IoT). **WSN-DS** contains 374,661 records representing normal traffic and four different types of DoS attacks, namely: flooding, TDMA, grayhole, and blackhole [17]. **BoT-IoT** compromises a approximately 3.6 million records in its proposed scaled version of the best 10 features [81]. These features were selected by comparing Entropy and Correlation scores. Ideal features have high Entropy (indicating less redundancy) and low Correlation (indicating less similarity to others). The scores were normalized, and Correlation values were inverted so that higher scores in both measures indicate more independent and informative features [81]. the dataset represents normal IoT network traffic and various

attacks that include DDoS, DoS, Keylogging, Data exfiltration, OS and Service Scan. Table 5.2 outlines the features used for the experiments, with detailed descriptions of the datasets' features provided in Table B.1 and Table B.2.

For the training and testing split, we used `scikit-learn`'s `train_test_split` function to randomly divide the WSN-DS dataset into 80% for training and 20% for testing, with the `random_state` parameter set to a default value. For BoT-IoT, we utilized the pre-split training and testing datasets provided by the original authors.

For training our supervised multi-classifier FFDNN, we used the labels in these datasets as the target variable. WSN-DS labels include blackhole, grayhole, flooding, scheduling, and normal traffic, while BoT-IoT labels differentiate between DDoS, DoS, theft, reconnaissance, and normal traffic, as shown in Table 5.2.

Features	**WSN-DS**	**BoT-IoT**
Binary	is_CH, JOIN_S, Join_R	N/A
Numeric	Who_CH, Dist To CH, Consumed_Energy, ADV_S, ADV_R, SCH_S, SCH_R, Rank, DATA_S, DATA_R, Data_Sent_To_BS, dist_CH_To_BS, send_code,	Seq, state_number, Stddev, Min, Max, Srate, Drate, N_IN_Conn_P_SrcIP, N_IN_Conn_P_DstIP
Categorical	N/A	Proto
Label	Blackhole, Grayhole, Flooding, Scheduling, or Normal.	DDoS, DoS, Theft, Reconnaissance, and Normal

Table 5.2 Used Features in the Datasets

5.5.2. Dataset Preprocessing

Data preprocessing is a crucial stage to convert raw inputs into an understandable format by the ML algorithm. The first step in this stage is **One-Hot Encoding** which converts categorical features into numeric ones. We applied this step to the BoT-IoT dataset. However, there was no need for one-hot encoding for the WSN-Ds as the dataset does not contain categorical features. The second step is standardization, in which all numeric features are converted into a standard scale. **Min-Max Normalization** was used to transform the features into a scale between 0 and 1. This step is essential as the dataset has feature values with different scales drawn from different distributions or tainted by outliers. Furthermore, normalization prevents the features with large values from dominating others which causes imbalanced results. This method converts the maximum value into 1, the minimum value into 0, and the other values into decimals between 0 and 1. It is calculated via the following equation:

$$X(norm) = \frac{X - X(min)}{X(max) - X(min)} \tag{5.1}$$

where X denotes the feature value, X_{min} the minimum feature value, X_{max} the maximum feature value.

One-hot encoding and min-max normalization are widely used for ML-based NIDS compared to other methods due to their specific advantages in handling categorical and numerical data[138]. One-hot encoding is superior to other encoding methods, like label encoding, because it prevents

models from misinterpreting categorical variables as ordinal data with inherent numerical relationships [5]. This is crucial for NIDS, where features like protocol types or service names do not have a natural order. In contrast, min-max normalization is preferred over techniques like standardization (z-score normalization) because it scales numerical features to a fixed range, ensuring that all features contribute equally during model training [138, 5]. This is especially important in NIDS, where features may vary significantly in magnitude and scaling all features to the same range helps improve the performance and stability of the model. Both methods ensure that the data is appropriately prepared for machine learning algorithms, enhancing the model's ability to learn and generalize from the data effectively.

5.5.3. Adversarial Attacks & Target Model Implementation

The Adversarial Robustness Toolbox (ART) library[110] was used to generate the adversarial examples using the examined approaches and with the default parameters. The target model was a Feed-Forward Deep Neural Network (FF-DNN) implemented using the Keras library with a TensorFlow backend. The architecture of the model and the training parameters are demonstrated in Table 5.3. The implementation details for the experiment can be found in the code repository[1].

Parameter	**Value**
No. of hidden layers	3
Layer 1	128 neurons
Layer 2	64 neurons
Layer 3	32 neurons
Dropout	0.25
Optimizer	ADAM
Activation function	ReLU and Sigmoid
Learning rate	0.01
Epoch	100
Batch Size	64

Table 5.3 Feed-Forward DNN Model Parameters

5.5.4. Evaluation Metrics

We selected Evasion Rate (ER) as the primary metric to evaluate the performance of the attack. ER refers to the proportion of perturbed attack instances misclassified as benign by the detection model. The higher achieved ER by the approach indicates a more performant attack.

$$ER = \frac{Misclassified\ Attacks\ Records}{Total\ Attacks\ Records} \times 100 \tag{5.2}$$

To measure the realism of the attack, we consider three metrics.

- Approaches producing adversarial examples by manipulating all the features are unlikely to lead to realistic attacks; the adversary cannot have control over all of the traffic features to change them in a fine-grained manner. Furthermore, such massive manipulation breaks

[1]The repository is available at: https://mega.nz/folder/U7lhARbY#0gZ8kghKYXrrDg_nvfPH7A

the semantic links between the correlated features. We introduce the metric PF measuring the percentage of perturbed features:

$$PF = \frac{Average\ of\ Perturbed\ Features}{Total\ Features} \times 100 \tag{5.3}$$

- Adversarial examples that do not comply with the network domain constraints given in Section 5.3, e.g. by introducing out-range values to the continuous features, assigning non-binary values to the binary features, and triggering multiple categories at once for categorical features, are unlikely to correspond to realistic traffic [14]. We introduce a generic metric VAE_b measuring the percentage of these void adversarial examples:

$$VAE_c = \frac{Attacks\ Records\ violating\ b}{Total\ Attack\ Records} \times 100 \tag{5.4}$$

  We consider in the following VAE_{oor}, VAE_{nb} and VAE_{mc} for the constraints out-range, non-binary and multi-categories, respectively.

- The Unrealism Index (UI) is calculated by averaging the metrics above that are relevant to a particular dataset.

$$UI_{WSN-DS} = \frac{PF + VAE_{or} + VAE_{nb}}{3} \tag{5.5}$$

$$UI_{BoT-IoT} = \frac{PF + VAE_{or} + VAE_{mc}}{3} \tag{5.6}$$

5.6. Experimental Results & Analysis

In this section, we report and analyze the outcomes of executing the attacks in targeted (T) and untargeted (U) setups over the two datasets.

Table 5.4 shows assessment results of attacks performance and unrealism over the two datasets, using the metrics introduced in the previous section, presented in decreasing order based on Evasion Rate. Figures 5.1 and 5.2 demonstrate the Evasion Rate and Unrealism Index of the attacks over the WSN-DS and BoT-IoT datasets, respectively.

Figures 5.3 and 5.4 demonstrate the average percentages of void adversarial examples for each validation metric over the two datasets for white-box and black-box attacks, respectively. The suffixes (-T) and (-U) were added to the attack names to refer to the attacks in targeted and untargeted setups, respectively.

Lastly, Table 5.5 displays the validation metrics that were violated by the attacks over the two datasets. The results were presented in descending order based on the number of violated validation metrics scored by the attacks over the two datasets.

5.6.1. Attacks Performance

As reported in Table 5.4 the model over the BoT-IoT dataset recorded less ER of 0.06 on the clean attack instances compared to the WSN-DS model which scored an ER of 2 as shown in Table 5.4. This difference can be justified by the imbalance of normal and attacks data distribution between the two datasets. The attack instances are the majority of the BoT-IoT dataset records with a percentage of 99%, while they are the minority in the WSN-DS dataset with a percentage of 9.2%. However, both models were able to detect the clean attack traces with high accuracy.

From Table 5.4, we can see the variation in the attack effectiveness in terms of ER over the two datasets. Overall, we can observe that the performance of each attack depend on the dataset type. What stands out in Figures 5.1 and 5.2 is that the attacks overall achieved higher evasion rates over the WSN-DS compared to the BoT-IoT. The attacks over WSN-DS achieved ERs between 100-1.44 and 61.65-0.01 over BoT-IoT.

The white-box and the black-box attacks performed better in both setups, targeted and untargeted, over the WSN-DS dataset compared to the BoT-IoT.

Two reasons can justify that; first, the proportion of benign traffic instances constitutes about 91% of the WSN-DS dataset, which enriches learning the characteristics of normal flow behavior by the targeted attacks.

The second reason can be attributed to the number and datatype in a dataset. The WSN-DS dataset consists of binary and continuous features represented in numbers. The attacks could introduce any arbitrary numbers to these fields with the possibility of generating successful evasive examples. However, the BoT-IoT dataset contains continuous features and a categorical feature (*proto*) as shown in Table 5.2. A categorical feature can take one value from a finite set of possible values. The proto feature can be a value from a set of five values i.e., *arp, tcp, udp, icmp, and ipv6-icmp.* After one-hot encoding, this feature is represented in a binary vector in which only the corresponding category is assigned to 1, and the others are zeros. The attacks spread their perturbations to all features and introduce arbitrary numbers to fields belonging to a categorical feature that must be zeros, and only one of them can be 1. Such massive perturbation results in corrupted examples that cannot evade the detection model and are easily detected. This explains the terrible performance of the attacks over the BoT-IoT dataset.

It is apparent from Table 5.4 that the BIM and PGD attacks are the top performing white-box attacks over the two datasets with ERs of 65 and 12 in the targeted and untargeted setups, respectively.

The results of our experiments support the findings of previous research that has demonstrated that multi-step (iterative) perturbation strategies such as PGD and BIM are among the strongest attacks compared to the single-step attack (e.g., FGSM) [85].

The multi-step adversarial perturbation generation is an extension of the single-step method in which it iteratively adds a perturbation that follows the sign of the gradient with respect to the current adversarial example of the original input [85]. The PGD attack is similar to BIM. The differences are that PGD adds more iterations and uses random initialization. Because of

that, they had the same effect on the detection model as shown in Table 5.4. Although other studies support the same finding of us [77, 45], the BIM attack was reported as performing better than the PGG in [72].

What is striking in Table 5.4 is that the black-box attacks performed better than the white-box, with the EN being the best.

Although the EN is optimized to limit total perturbation across feature-space inputs, it minimizes the number of perturbed features. Therefore, the high effectiveness of this attack can be attributed to its ability to produce adversarial examples with minimal perturbation. As a consequence, the resulting examples become very close to the original examples and can successfully fool the detection model.

The Boundary and HSJ reported closer ERs. That can be justified by the fact that they are both from the same family of decision-based attacks, with HSJ is an extension of the Boundary attack.

	Attack	Setup	WSN-DS					BoT-IoT					Avg.	
			%ER	%PF	%VAE		UI_{WSN-DS}	%ER	%PF	%VAE		$UI_{BoT-IoT}$	ER	UI
					VAE_{or}	VAE_{nb}				VAE_{or}	VAE_{mc}			
-	Clean	-	2	0	0	0	0	0.06	0	0	0	0	1.03	0
White-box	BIM	T	84.59	92.69	100	98.79	97.16	45.43	85.27	98.38	93.73	92.46	65.01	94.81
	PGD	T	84.59	92.69	100	98.79	97.16	45.43	85.27	98.38	93.73	92.46	65.01	94.81
	CW2	T	36.36	44.94	0	34.36	26.43	0.89	33.53	0.47	0.83	11.61	18.63	19.02
	FGSM	T	12.51	100	100	100	100	3.85	100	100	100	100	8.18	100
	CW∞	T	2	98.5	0	97.92	65.47	0.06	99.8	4.03	99.94	67.92	1.03	66.7
	NewtonFool	U	22.2	90.63	85.04	89.59	88.42	4.76	90.33	90.47	90.43	90.41	13.48	89.42
	BIM	U	19.77	91.38	92.9	92.9	92.39	3.83	89.93	94.05	94.05	92.68	11.8	92.54
	PGD	U	19.77	91.38	92.9	92.9	92.39	3.83	89.93	94.05	94.05	92.68	11.8	92.54
	CW2	U	20.26	70.44	0	63.3	44.58	0.01	55.13	1.11	32.45	29.56	10.14	37.07
	DeepFool	U	7.02	93.13	87.61	92.88	91.21	0.35	94.33	94.04	94.05	94.14	3.69	92.68
	FGSM	U	4.81	94.25	92.9	92.88	93.34	0.5	95.8	94.05	94.05	94.63	2.66	93.99
	CW∞	U	4.22	22	0	3.94	8.65	0.03	43.27	1.85	15.98	20.37	2.13	14.51
Black-box	EN	T	85.02	50.06	0	42.59	30.88	61.65	49.93	0	43.49	31.14	73.34	31.01
	HSJ	T	100	92.69	0	98	63.56	23.32	47.2	4.14	23.31	24.88	61.66	44.22
	Boundary	T	91.79	95.31	0	98	64.44	20.38	48.73	1.3	24	24.68	56.09	44.56
	ZOO	T	1.41	1.81	0.59	0.59	1	0.01	1.07	0.04	0.04	0.38	0.71	0.69
	HSJ	U	28.1	85.88	0	99.16	61.68	0.48	86.07	58.72	99.74	81.51	14.29	71.6
	Boundary	U	26.89	91.25	0	99.99	63.75	0.98	93.93	53.4	100	82.44	13.94	73.1
	EN	U	10.36	34.44	0	5.77	13.4	1.01	41.2	0	14.33	18.51	5.69	15.96
	ZOO	U	6.86	13.06	13.89	13.96	13.64	0.15	50.07	55.1	52.94	52.7	3.51	33.17

Table 5.4 Attacks Assessment on WSN-DS & BoT-IoT Datasets

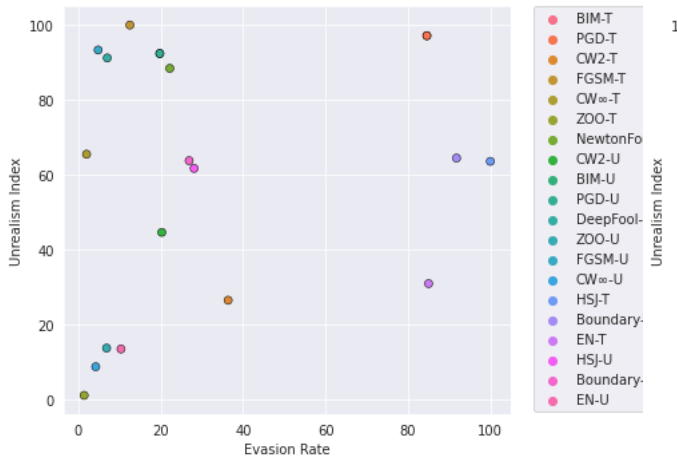

Figure 5.1 Evasion Rate vs. Unrealism Index over WSN-DS

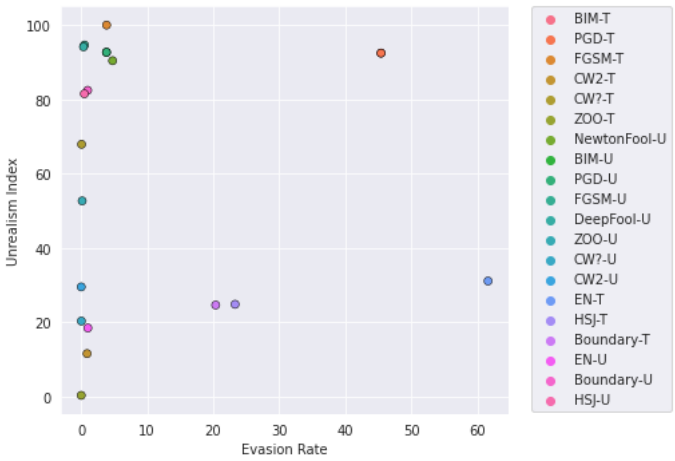

Figure 5.2 Evasion Rate vs. Unrealism Index over BoT-IoT

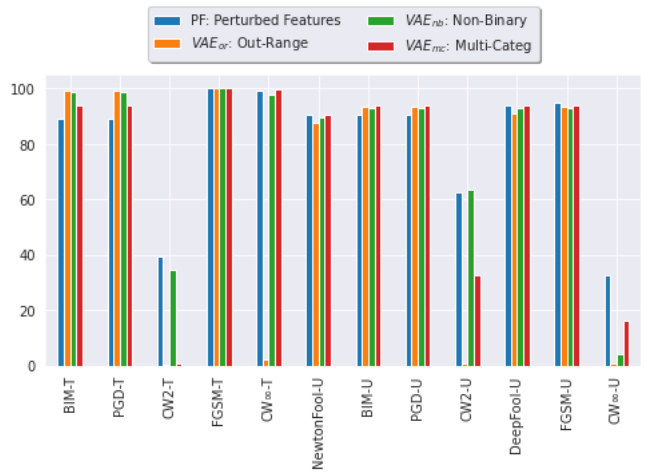

Figure 5.3 White-box Attack Unrealism Metrics Percentages

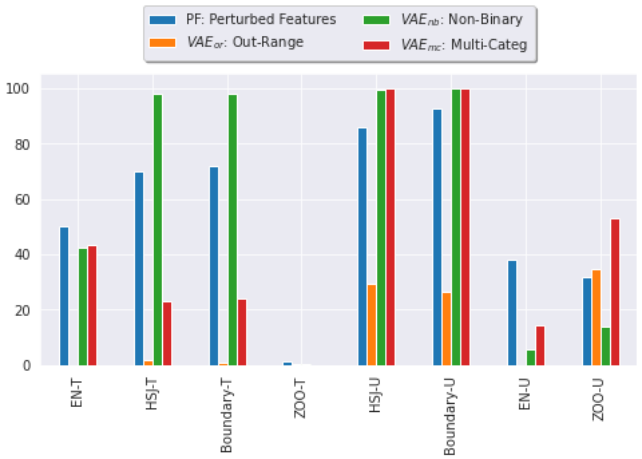

Figure 5.4 Black-box Attack Unrealism Metrics Percentages

5.6.2. Attacks Unrealism

In Figure 5.1, the data points are more widely dispersed, indicating significant variability in the unrealism index even among attacks with similar evasion rates. This suggests that while some attacks achieve high evasion rates, they also generate highly unrealistic traffic, which could make them easier to detect. Conversely, in Figure 5.2, the attacks are more clustered, especially for lower evasion rates, indicating that the unrealism index is more consistent in this dataset. This consistency might imply that, for BoT-IoT, attacks that succeed in evading detection are more uniformly realistic, potentially posing a greater challenge for NIDS models. Overall, these figures highlight the importance of considering both the effectiveness and the realism of adversarial attacks when evaluating the resilience of NIDS models.

Percentage of Features Perturbation: As shown in Figure 5.3 the majority of white-box attacks manipulated on average above 85% of the features over the two datasets including BIM, PGD, CW, and NewtoonFool which were the best performing. However, CW2-T and CW∞-U altered around 35% of the features. Though, as shown in Table 5.4 they reported low ERs of 18.63 and 2.13, respectively. Most of the black-box attacks manipulated around 62% of the feature, as can be observed from Figure 5.4. Although the ZOO-T, ZOO-U, and EN-U attacks were the least, they scored low ERs between 0.71-5.69 (Table 5.4).

From these results, it is clear that these attacks altered a vast amount of the traffic features by over 50%. The unrealism of the attack can be attributed to the infeasibility of accessing and controlling such amount of features by the adversary in real-world scenarios. Furthermore, massive features modification will break the semantic links among the features, invalidating traffic traces realism.

Out-Range Values: It is apparent from Figure 5.3 that the white-box attacks except CW2 and CW∞ generate above 90% adversarial examples that hold out-range values. The CW2 and CW∞ produce between 0.24-2.02 void adversarial examples. However, their ERs range between 18.63-1.03, as can be seen in Table 5.4. The targeted black-box attacks produced on average

less than 2.07% adversarial examples with out-range values, and 23% in the untargeted setup, as shown in Figure 5.4.

Each feature in traffic can take a value within a limited range of possible values. For instance, the *Rank* feature in WSN-DS has originally a range of values between [0,99] which scaled to [0,1] using min-max normalization for model training. However, the BIM-T attack, best performing, introduced values [-0.3,0.57] for that feature which do not comply with the [0,1] range.

Non-Binary Values: As can be seen in Figure 5.3, the majority of white-box attacks introduced non-binary values to the binary features with percentages of void adversarial examples above 90%, except CW2 and CW∞-U. In the back-box attacks, Boundary and HSJ generated above 98% void adversarial examples. The ZOO attack produced the lowest percentage of void adversarial examples. However, it achieved averaged ERs between 0.71 and 3.51 in targeted and untargeted setups, respectively.

The WSN-DS dataset contains three binary features: *Is_CH, JOIN_S, JOIN_R* which can take only 0 or 1, as shown in Table 5.2. However, the BIM-T attack introduced values between [-0.1,1.3], [-0.3,1.1], [-0.3,1.09] for those features, respectively. For instance, the *Join_S* feature in the WSN-DS dataset denotes whether the join request was sent from the sensor node to the head of the cluster, which can be True or False, i.e., a flag value of 1 or 0. Assigning a decimal or negative value to this feature makes no sense.

Multi-Category Belonging Values: It is apparent from Figure 5.3, that the majority of white-box attacks triggered multiple categories at once for the categorical feature for above 90% of adversarial examples, except CW2 and CW∞-U. In the untargeted back-box attacks, Boundary and HSJ generated almost 100% void adversarial examples. The other attacks produced less than 50%.

The BoT-IoT dataset includes a categorical feature *proto* as shown in Table 5.2. A categorical feature contains a limited number of possible values. The *proto* feature has five values i.e., *arp, tcp, udp, icmp, and ipv6-icmp*. After one-hot encoding, this feature is mapped into a binary vector containing either 0 or 1. Here, only the associated category is assigned to 1 and the others to 0. However, the attacks spread their perturbations overall of the fields that belong to the encoded categorical feature, which triggers multiple categories at once.

5.7. Discussion

In this section, we summarize the key findings and place our findings in the context of the literature that has employed the examined attacks to assess the resilience of ML-based NIDS to adversarial evasive examples. We discuss the literature from two points of view: First, the compatibility of the generated adversarial traffic with domain constraints of network traffic; Second, how likely the adversary can utilize these attacks for real-world scenarios.

5.7.1. Attacks Unrealism

Previous studies on assessing DL-based NIDS resilience to adversarial attacks ignored the compliance of generated examples with network domain constraints [170, 169, 175, 42, 69, 72, 92, 123, 75, 149, 117, 45, 52, 89]. Our findings note that the outputs of these attacks are void and unrealistic as they do not obey traffic data restrictions.

As can be seen in Table 5.4, the top performant techniques violated the validation metrics for realistic adversarial attacks. The attacks vary in the percentages of unrealistic adversarial examples they produce. Some approaches generated less unrealistic adversarial examples. However, they were the least performing attacks. On the other hand, the highest effective attacks created the highest percentages of unrealistic adversarial examples.

We found that all of the attacks introduce non-binary values to binary features and trigger multiple categories at once to categorical features, as demonstrated in Table 5.5. Most of the attacks violate all of the metrics explained in sections 6.6.5 and 5.3 over the two datasets as shown in Tables 5.5.

Although some of the adversarial examples generation methods can be theoretically successful, no attack maintains all of the domain constraints. These techniques can not lead to practical and realistic attacks as they violate the network domain constraints. They break the semantic links among the features due to the high percentage of perturbed features, as shown in Table 5.4.

These findings indicate that these attacks result in void data that cannot represent practical and realistic packets that can be delivered over the network. Therefore, they cannot be used to prove the resilience of DL-based NIDS to adversarial evasive flow in a real-world setup.

	Attack	Setup	WSN-DS			BoT-IoT		
			A	B	C	A	B	D
White-box	BIM	T	✓	✓	✓	✓	✓	✓
	FGSM	T	✓	✓	✓	✓	✓	✓
	PGD	T	✓	✓	✓	✓	✓	✓
	CW∞	T	✓		✓	✓	✓	✓
	CW2	T			✓		✓	✓
	BIM	U	✓	✓	✓	✓	✓	✓
	DeepFool	U	✓	✓	✓	✓	✓	✓
	FGSM	U	✓	✓	✓	✓	✓	✓
	NewtonFool	U	✓	✓	✓	✓	✓	✓
	PGD	U	✓	✓	✓	✓	✓	✓
	CW2	U	✓		✓	✓	✓	✓
	CW∞	U			✓		✓	✓
Black-box	Boundary	T	✓		✓		✓	✓
	ZOO	T		✓	✓		✓	✓
	HSJ	T	✓		✓		✓	✓
	EN	T	✓		✓			✓
	ZOO	U		✓	✓	✓	✓	✓
	Boundary	U	✓		✓	✓	✓	✓
	HSJ	U	✓		✓	✓	✓	✓
	EN	U			✓			✓

A=%of Perturbed Features over 50% B=Out-Range Values
C=Non-Binary Values D=Multi-Class Values

Table 5.5 Attacks Unrealism Metrics on WSN-DS & BoT-IoT Datasets

5.7.2. Attacks Infeasibility

Similar to the literature, the implemented attacks work with feature vectors extracted from preprocessed raw network traffic in the form of tabular CSV files. Such attacks are known as feature-space attacks, in which perturbations are applied directly to the inputs of the detection model. The data processor component in the NIDS parses raw packets to extract important features analyzed by the detection engine to classify the passing traffic using pre-constructed ML models. To implement such attacks, the adversary either has to know what features are parsed on the other side or control the channel of transforming the raw traffic to the preprocessed ML model inputs. Acquiring capabilities over the feature set or the preprocessing pipeline by the adversary is unlikely feasible for real-world scenarios. Differently, the problem-space attacks involve manipulating the actual packets and producing new adversarial ones. The difficulty of these attacks lies in perturbing the original raw input that corresponds to the adversarial feature vector. Although they are challenging to implement, they are feasibly realistic as the adversary can have the capability to craft the packet contents compared to knowing the feature set or controlling the preprocessing procedure.

A common drawback in literature is they assume white-box attack scenarios where the adversary can access everything related to the target system and have complete knowledge of the model architecture, parameters, hyperparameters, weights, and configurations. Hence, the adversary can directly craft adversarial examples by computing or approximating the model gradients [170, 169, 42, 69, 72, 92, 123, 96, 75, 149, 117, 45, 52, 125]. To gain such knowledge, the adversary must access the model source code. However, the source code can be unobservable for a commercial NIDS or securely stored on a different machine for in-house NIDS [21]; hence, these attacks are unlikely feasible. In real-world scenarios, the white-box attack assumption is unlikely common as the adversary in most cases an outsider.

Furthermore, they did not consider that the required perturbations for generating the adversarial examples do not directly correlate to modifying the actual network packets; hence, their incapability for end-to-end attacks was not taken into account. The authors applied the generic adversarial examples generation methods to the statistical features collected from the packet metadata. Moreover, they did not demonstrate how the adversarial raw packets can be generated. These attacks are known as feature-space attacks that transform the original feature vector as an into a new perturbed feature vector. Although such attacks can be theoretically successful, they operate at the preprocessed traffic data level, not at the packet level; therefore, the adversary cannot use them for real-world scenarios. On the other hand, the problem-space attacks perturb the raw packets to produce new functional adversarial ones that can result in realistic end-to-end attacks.

5.8. Conclusion

In this chapter, we validated the compliance of the generated adversarial examples with network domain constraints of network traffic and discussed the feasibility of utilizing these examples

for real-world attacks. We assessed the outputs of seven white-box and four black-box attacks widely used in the literature, and they were implemented in different settings: targeted and untargeted. Furthermore, we incorporated a wireless sensor network traffic representing a different networking environment that has not been investigated and an IoT network traffic.

We demonstrated the effect of adversarial evasion attacks on the performance of a DL-based NIDS. Overall, the attacks vary in their performance, and some attacks achieved remarkable Evasion Rates. However, they result in void adversarial examples that do not comply with the network traffic domain constraints. The examined attacks introduce arbitrary and unrealistic perturbations such as non-binary values to the binary features that only accept 0 or 1, out-range values to the numeric features that have a fixed range of possible values, or trigger multiple categories at once to the categorical features. Furthermore, some of the attacks manipulate more than half of the traffic features; controlling such amount of features in a fine-grained manner is unlikely feasible to the adversary and eventually breaks the semantic links between the features. Based on these Unrealism metrics, we concluded that although these attacks can be performant, they are impractical and unrealistic for DL/ML-based NIDSs.

For realistic adversarial evasion attacks setups, we derived the following considerations: crafting valid examples that comply with network traffic domain constraints, using black-box threat models, and maintaining minimal knowledge and capabilities over the target model.

Assessing ML-based NIDS against even unrealistic adversarial attacks offers crucial insights into the limits of ML-based NIDS models. By analyzing these attacks, we can devise better detection mechanisms for real-world attacks and prepare NIDS for future, more sophisticated threats and extreme conditions. This sets the stage for the next chapter, where we evaluate the resilience of a wide range of ML-based NIDS models and detect adversarial attacks.

Chapter 6. Resilience Evaluation and Detection of Adversarial Attacks in ML-based NIDS

6.1. Summary

Based on our systematic review in Chapter 4, we found a lack of comprehensive evaluations encompassing diverse models, attack types, and defense strategies using contemporary network traffic data. This hinders the verification of findings for real-world scenarios. The absence of standardized metrics complicates comparisons of ML-based NIDS resilience. Additionally, there is no lightweight solution that effectively detects and classifies adversarial traffic with high accuracy on both clean and perturbed data, demonstrating efficiency across recent datasets, diverse attacks, and defenses. Therefore, in this chapter, we conduct a comprehensive evaluation, including 15 detection models, 8 adversarial attacks, and 3 defense approaches. We introduce the Resilience Index as a metric for evaluating the resilience of ML-based NIDS models against adversarial attacks. We analyze attacks performance and stability across different models and defense methods. We investigate ensemble learning and adversarial training as defense mechanisms against adversarial examples and compare their strengths and weaknesses. Subsequently, we introduce the Adversarial-Resilient NIDS, a framework featuring an adversarial attack detector aimed at identifying such attacks. Utilizing an adversarially trained ensemble (ATE), the detector excels in accurately identifying eight types of adversarial attacks. Through rigorous experimental evaluations conducted over the NF-UQ-NIDS dataset, a recent standardized network traffic dataset that accumulated from various environments including traditional and IoT networks featuring twenty types of cyber attacks, the ATE detector showcases remarkable efficacy in comparative analyses with state-of-the-art defenses. It significantly diminishes the attack success rate from 0.41 to a mere 0.03, while achieving an exceptional overall performance rating of 0.98. Furthermore, the proposed framework incorporates an adversarial attack classifier designed to determine the type of employed adversarial attack after its detection, exhibiting a classification accuracy of 0.97.

6.2. Introduction

Adversarial examples feature transferability property refers to the ability to fool multiple models, even those with different internal architectures [151]. This allows attackers to use a surrogate model to create adversarial traffic that deceives one detector and then use the same samples to

attack another model without knowing its internals [151]. Detecting and assessing the resilience of ML-based NIDS against adversarial examples are crucial for network security. These examples exploit ML model vulnerabilities, causing misclassifications and potentially allowing malicious traffic to go undetected. [66, 10]. The aim of this chapter is to answer the following research questions:

1. **What metric can effectively evaluate the resilience of ML-based NIDS models against adversarial attacks, and how do the models rank accordingly?** (Section 6.7.3)

   Employing resilient ML-based models against adversarial attacks for NIDSs requires the identification of an effective evaluation metric to quantify their resilience. This question aims to determine such metric and provide models' resilience ranking.

2. **Which attacks maintain consistent performance across different model architectures, and how are they impacted by various defense strategies?** (Section 6.7.4 and 6.8)

   Tailoring countermeasures to universally effective attacks can optimize defense efficiency. Additionally, understanding how defenses impact the attacks is crucial for evaluating their efficacy and limitations. This question seeks to identify the consistent performing attacks for better defense prioritization, and find the best defense for each attack type.

3. **Can ensemble methods improve the resilience of ML-based NIDS against transferable adversarial example?** (Section 6.8.1)

   Ensemble methods combine multiple NIDS models to make collective decisions. This question examines whether ensemble techniques can mitigate the transferability of adversarial examples by leveraging diverse models with different vulnerabilities and decision-making processes.

4. **To what extent does adversarial training enhance the resilience of models against adversarial attacks, and does it result in degradation in their performance?** (Section 6.8.2)

   Adversarial training retrains models with adversarial examples to enhance resilience, yet excessive exposure to adversarial perturbations may cause models to prioritize recognizing them, potentially hindering generalization. This question examines adversarial training's impact on model resilience against attacks and overall performance.

5. **Can integrating adversarial training with ensembling techniques improve the detection of adversarial examples against ML-based NIDS?** (Section 6.8.3)

   Adversarial training strengthens a model against specific attacks, while ensemble methods enhance performance by using diverse models. Combining these approaches can leverage their complementary nature. This question examines whether such integration improves models resilience and overall performance.

Contributions: The contributions of this chapter are as follows:

- **C1:** To provide a comprehensive evaluation for adversarial attacks' performance, detection models' resilience, and defenses methods' effectiveness.
- **C2:** To introduce the Resilience Index, a standardized measure for evaluating the resilience of ML-based NIDSs.
- **C3:** To analyze the performance consistency of attacks across different detection models and defense mechanisms.
- **C4:** To evaluate the state-of-the-art defenses, ensemble learning and adversarial training, and compare their strengths and weaknesses
- **C5:** To present the Adversarial-Resilient NIDS, a framework for effectively and efficiently detecting and classifying adversarial attacks.
- **C6:** To identify the most effective defense strategy for each attack type.

Organization: The rest of this chapter is structured as follows: Sec 6.3 explores the limitations of the previous related works and the gaps we do address. Sec 6.4 describes our proposed Resilience Index. Sec 6.5 introduces the AR-NIDS framework. Sec 6.6, details the experimental setup. Sec 6.7, presents the experimental results. Sec 6.8, evaluates the proposed AR-NIDS framework. Sec 6.9, demonstrates the impact of defense strategies on adversarial attacks efficacy. Sec 6.10, discusses our results' implications, highlighting the strengths and limitations of attacks, models and defenses. Finally, Sec 6.11 concludes with a summary for the chapter.

6.3. Literature Drawbacks

The literature employs two major defensive mechanisms to increase the resilience of DL-based NIDS towards evasion adversarial attacks: adversarial training and ensemble learning. Adversarial training involves training the model on both clean and adversarially crafted examples to improve its resilience against thee examples [132, 92, 45, 52, 125, 140, 1, 178]. On the other hand, ensemble learning enhances prediction accuracy by combining diverse models and compensating for individual weaknesses. In adversarial detection, this integration of multiple models with different strategies can hinder adversarial attacks from exploiting vulnerabilities in a single model [133, 46].

There are several notable gaps in the existing research on defense mechanisms against adversarial attacks for NIDS. Firstly, the reliance on outdated datasets like NSL-KDD raises concerns about the generalizability of findings to current network traffic and the evolving threat landscape [45, 140, 133, 46]. Using more recent and diverse datasets is essential for understanding defense mechanisms' effectiveness in real-world scenarios. Additionally, most studies use only one dataset, which limits the generalizability of findings [132, 52, 125, 140, 158, 178, 133, 46, 95]. Another gap is the limited evaluation of defense mechanisms against various types of adversarial attacks. Many studies focus on specific attack methods, which does not provide a comprehensive

view of a defense mechanism's resilience [52, 125, 158, 115, 95]. Evaluating defense strategies against a broader spectrum of attacks is crucial for identifying strengths and weaknesses across different threat scenarios. Additionally, most evaluations are conducted under white-box attack scenarios, overlooking the importance of assessing defense mechanisms against black-box attacks where attackers have limited knowledge. Real-world attackers often operate under such constraints, making it imperative to evaluate the resilience of defense mechanisms in more realistic scenarios.

Previous studies have not provided solutions for detecting and classifying adversarial traffic that ensure both resilience against adversarial attacks and sustained high performance on clean data. Furthermore, there is a lack of thorough evaluations across various adversarial attack generation techniques. To address these gaps, we propose the Adversarial-Resilient NIDS framework. This framework employs a multi-layered approach, including an Intrusions and Adversarial Detector and an Adversarial Attacks Classifier, to effectively detect and classify adversarial attacks. We evaluate its effectiveness using the NF-UQ-NIDS dataset [137], which includes recent network traffic data from traditional and IoT networks and features twenty types of cyber attacks. Our comprehensive assessment covers 8 types of adversarial attacks, including white-box and black-box scenarios, and 15 different machine learning methods for NIDS models.

Table 6.1 provides a comparative overview of employed defense mechanisms against specific white-box and black-box adversarial evasion attacks. These mechanisms as discussed above include: Adversarial Training (AT), Ensemble Learning (EL), and our approach Adversarially Trained Ensemble (ATE).

	Ref	[132]	[73]	[133]	[92]	[45]	[52]	[125]	[140]	[1]	[32]	[147]	[139]	[158]	[178]	[27]	[46]	[95]	[112]	[115]	[122]	[86]	[114]	[109]	[8]	[35]	[144]	[62]	[158]	[4]	Ours
⊙	JSMA	✓		✓	✓							✓										✓	✓	✓							✓
	CW	✓			✓							✓										✓	✓	✓						✓	✓
	DeepFool	✓			✓							✓	✓										✓	✓						✓	✓
	FGSM	✓	✓		✓	✓	✓	✓		✓		✓	✓			✓	✓	✓			✓	✓	✓	✓							✓
	PGD					✓			✓				✓				✓				✓	✓	✓							✓	✓
	ZOO																					✓			✓						✓
	HSJ														✓																✓
⊕	EL			✓					✓			✓			✓		✓					✓	✓	✓				✓			✓
	AT	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	✓	✓	✓			✓	✓	✓		✓	✓	✓
	ATE																														✓

Table 6.1 Employed Attacks and Defense Comparison [⊙=Attacks, ⊕=Defense]

6.4. Resilience Index

In this section, we introduce the Resilience Index (RI) to evaluate ML-based NIDS models' resilience against adversarial attacks. RI combines key indicators: balanced accuracy (BA_{adv}, macro F1 score ($F1_{adv}$), and Attack Success Rate (ASR). BA_{adv} and $F1_{adv}$ measure detection accuracy, while ASR gauges vulnerability. RI provides a unified metric that balances detection performance and adversarial robustness. Here's a breakdown of the mathematical reasoning behind the formulation:

- **BA_{adv}:** The average of the true positive rate (sensitivity) and the true negative rate (specificity). It is particularly useful when dealing with imbalanced datasets, as it provides a more accurate reflection of the model's performance across different classes. In the

context of adversarial attacks, $\mathrm{BA}_{\mathrm{adv}}$ represents the model's ability to correctly classify both normal and adversarially perturbed samples.

$$\mathrm{BA}_{\mathrm{adv}} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}\right)$$

- **$\mathrm{F1}_{\mathbf{adv}}$:** The harmonic mean of precision and recall. It balances the trade-off between false positives and false negatives. In adversarial settings, $\mathrm{F1}_{\mathrm{adv}}$ indicates the model's effectiveness in correctly identifying adversarial samples while minimizing misclassifications.

$$\mathrm{F1}_{\mathrm{adv}} = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Where:

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

- **ASR:** The proportion of adversarial samples that successfully evade detection. A higher ASR indicates that the adversary is more successful in fooling the NIDS. Therefore, ASR is a negative indicator of resilience: the lower the ASR, the more resilient the system.

$$\mathrm{ASR} = \frac{\text{Number of Successful Adversarial Samples}}{\text{Total Number of Adversarial Samples}}$$

- **RI:** is formulated by combining the metrics as:

$$\mathrm{RI} = \frac{(\mathrm{BA}_{adv} + \mathrm{F1}_{adv} - \mathrm{ASR}) + 1}{3}$$

The formula is designed to balance these three metrics into a single value (RI), which ranges between 0 and 1:

Best Case (RI = 1): When When BA_adv = 1, F1_adv = 1, ASR = 0

Substituting these values:

$$\mathrm{RI} = \frac{(1+1-0)+1}{3} = \frac{3}{3} = 1$$

This represents the scenario where the model performs perfectly against adversarial attacks.

Worst Case (RI = 0): When BA_adv = 0, F1_adv = 0, ASR = 1

Substituting these values:

$$\mathrm{RI} = \frac{(0+0-1)+1}{3} = \frac{0}{3} = 0$$

This indicates that the model fails to defend against adversarial attacks, resulting in a poor resilience score.

- **Interpretation of Each Term:**
  - BA_{adv} and $\mathrm{F1}_{adv}$ are positively associated with resilience; higher values are better.
  - ASR is negatively associated with resilience; higher values indicate a weaker system.
- **Normalization:**
  - The addition of 1 ensures that the numerator is positive even if $\mathrm{BA}_{adv} + \mathrm{F1}_{adv} -$ ASR is slightly negative.
  - Dividing by 3 averages the three components, normalizing the index to a consistent scale.
- **Mathematical Justification:** The Resilience Index can be viewed as a weighted sum of the key performance metrics, normalized to a fixed scale. Here's the breakdown:
  - **Weighted Sum:** The formula $\mathrm{BA}_{adv} + \mathrm{F1}_{adv} -$ ASR combines the metrics linearly, reflecting the contribution of each to the overall resilience.
  - **Normalization and Shift:** Adding 1 ensures that the index is positive, avoiding negative values that could be non-intuitive. Dividing by 3 scales the result to a more interpretable range.
  - **Interpretability:** The index simplifies interpretation by combining three critical aspects of resilience into a single score. This allows for easy comparison across different models or configurations.

6.5. Adversarial-Resilient Network Intrusion Detection System (AR-NIDS)

In this section, we present the Adversarial-Resilient NIDS, a layered framework for detecting and classifying adversarial attacks.

Figure 6.1 illustrates the two main layers of our framework: the Adversarially Trained Ensemble Detector and the Adversarial Attacks Classifier. The Adversarially Trained Ensemble (ATE) detector combines the power of adversarial training and ensemble learning. Adversarial training exposes the model to adversarial examples during training to enhance its resilience. Ensemble learning leverages the diversity of multiple models, achieving superior performance compared to individual models. The ATE detector is a multiclassifier trained with the LGBM algorithm on a dataset of clean and perturbed data, labeled as normal, attack, or adversarial. To optimize performance, ensemble techniques like bagging and boosting are used. Bagging trains multiple models on different data subsets and aggregates their predictions, while boosting iteratively improves models by focusing on misclassified instances, creating a resilient final model. In operation, the ATE detector categorizes incoming traffic: normal traffic passes through, while abnormal traffic triggers alerts, blocks, or activates security protocols. Adversarial traffic is routed to the Adversarial Attack Classifier, which uses an LGBM classifier to identify the specific attack type. This enhances threat understanding and enabling targeted defense.

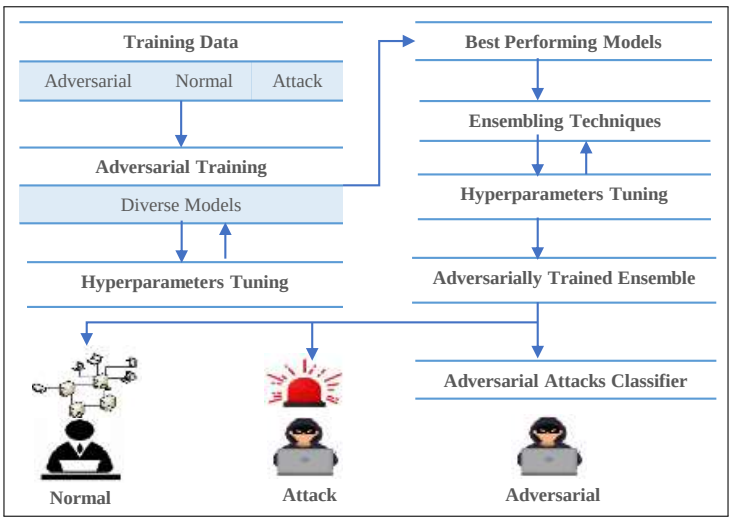

Figure 6.1 Adversarial-Resilient NIDS

6.6. Experimental Setup

6.6.1. Adversary Model

We assume the adversary's goal is to subvert the network detection model by generating malicious traffic that evades detection. He exploits vulnerabilities in the model's decision-making process, crafting adversarial examples that cause misclassification while maintaining malicious intent, such as data exfiltration, launching attacks, or compromising network services. Using advanced techniques like gradient-based or optimization-based attacks, the adversary designs examples specifically to fool the detection model. He can have a complete knowledge of the model's architecture, parameters, and training data, acting as either an insider or an external threat with access to the model's source code, data, or a substitute model that provides insights into the detection model's behavior.

6.6.2. Dataset

Numerous publicly accessible datasets have been used to develop and assess ML-based NIDS. However, variations in feature sets among these datasets make it challenging to reliably compare ML models across different scenarios. This hinders their generalization across diverse network environments and attack scenarios. This gap prompted Sarhan et al. [137] to present standardized NIDS datasets with consistent, practical features based on NetFlow. We utilized the NF-UQ-NIDS dataset [137], which contains 11,994,893 records: 9,208,048 benign (76.77%) and 2,786,845 attack records (23.23%), and includes twenty distinct attack categories. The original dataset contains nearly 12 million records, which can be computationally intensive to process and analyze. Sampling a representative subset allows for more efficient experimentation, reducing the computational resources and time required while still maintaining the integrity and diversity of the data. We selected a representative subset of 6,338,509 records, with 81.8% normal and 18.2% attack records. The detailed description of the dataset's features is demonstrated in Table B.3.

Table 6.2 summarizes the prevalence of each type of traffic. The dataset was stratified into training (80%) and testing (20%) subsets based on class labels. To streamline the task into a binary classification problem for network anomaly detection, all attack types were unified under a single class label, 'Attack'. Table 6.3 shows the distribution of samples in the training and testing datasets.

Flow Type	**Number of Samples**	**Ratio %**
Benign	5,184,112	81.7727
DDoS	151,866	2.3942
password	151,866	2.3942
injection	151,866	2.3942
xss	151,866	2.3942
DoS	151,866	2.3942
Reconnaissance	151,866	2.3942
scanning	151,866	2.3942
Brute Force	27,721	0.4372
Infiltration	26,131	0.4121
Bot	16,829	0.2655
Exploits	6,591	0.1039
Fuzzers	4,462	0.0704
Backdoor	3,856	0.0608
Generic	2,268	0.0358
mitm	1,723	0.0272
ransomware	784	0.0124
Theft	533	0.0084
Shellcode	312	0.0049
Analysis	90	0.0014
Worms	35	0.0006
Total	6,338,509	100

Table 6.2 Statistics of Used NF-UQ-NIDS Dataset

Samples	**Train**	**Test**
Normal	4,147,289	1,036,823
Attack	923,518	230,879
Adversarial	820,903	205,224

Table 6.3 Samples Distribution in Train and Test Data

6.6.3. Dataset Preprocessing

The features were transformed using Min-Max normalization, rescaling them within the range of 0 to 1. This normalization addresses issues of non-uniform distributions and outliers, preventing largevalued features from dominating and causing imbalanced results. Mathematically, Min-Max scaling is represented as:

$$X(norm) = \frac{X - X(min)}{X(max) - X(min)} \tag{6.1}$$

6.6.4. Adversarial Attacks & Models Implementation

The Adversarial Robustness Toolbox (ART) library[110] was used to generate adversarial examples within a targeted attack scenario, using default parameters. Eight adversarial attack techniques—FGSM, PGD, HSJ, JSMA, DeepFool, CW2, ZOO, and CW∞ —were employed to create adversarial instances of malicious network traffic samples. These methods generated

102,612 adversarial examples for training and 25,623 for testing by manipulating true attack samples in the datasets. Table 6.3 shows the distribution of samples in the training and testing datasets.

The Feed-Forward Deep Neural Network (FF-DNN) was built using Keras with TensorFlow backend. Bayesian Optimization was used for hyperparameter tuning to achieve optimal accuracy. Bayesian Optimization is a method for finding the best value of a function with expensive evaluations. It uses a probabilistic model, like a Gaussian Process, to predict the function's behavior and guide where to sample next. This helps efficiently find the optimum with fewer evaluations [51]. Additionally, fifteen conventional Machine Learning (ML) algorithms were selected due to their widespread use in developing ML-based NIDS. These algorithms include Random Forest (RF), Decision Tree (DT), Extra Trees (ET), Logistic Regression (LR), eXtreme Gradient Boosting (XGBoost), Catboost, Gradient Boosting Decision Tree (GBDT), Light Gradient Boosting Machine (LGBM), Adaptive Boosting (Adaboost), Multilayer Perceptron (MLP), Multinomial Naive Bayes, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Stochastic Gradient Descent (SGD). For reproducibility, classifiers were implemented with default parameters as outlined in Table 6.4. This study focuses on assessing the transfer and impact of adversarial examples on the overall accuracy of these models. It compares their performance under normal and adversarial conditions. The implementation details for the experiment can be found in the code repository[1].

Model	**Parameters**
FF-DNN	Layers: [Dense(128, activation=relu), Dropout, Dense(64, activation=relu), Dropout, Dense(2, activation=softmax)], Optimizer: Adam
RF	n_estimators=100, criterion='gini', max_depth=None
DT	criterion='gini', splitter='best', max_depth=None
ET	c n_estimators=100, criterion='gini', max_depth=None
LR	penalty='l2', C=1.0, solver='lbfgs'
XGBoost	objective='binary:logistic', n_estimators=100, max_depth=3
Adaboost	base_estimator=None, n_estimators=50, learning_rate=1.0
NB	alpha=1.0, fit_prior=True, class_prior=None
LDA	solver='svd', shrinkage=None, priors=None
QDA	priors=None, reg_param=0.0, store_covariance=False, stor_covariances=None
SGD	loss='hinge', penalty='l2', alpha=0.0001
GBDT	n_estimators=100, learning_rate=0.1, max_depth=3, mi_samples_split=2
Catboost	iterations=100, depth=3, learning_rate=0.03, boosting_type= 'Ordered'
LGBM	n_estimators=100, learning_rate=0.1, max_depth=-1, boosting_type= 'gbdt'
MLP	hidden_layer_sizes= (100,) activation= 'relu' solver= 'adam' learning_rate_init= 0.001

Table 6.4 Parameters of ML-based NIDS Models

The assessment of the transferability property of adversarial attacks was designed as follows, see Figure. 6.2: 1) randomly split the dataset into training and testing sets, 2) train a range of binary supervised classification algorithms, 3) generate malicious adversarial traffic against the FF-DNN model in a white-box setup, 4) evaluate the attack success rate of the generated adversarial samples in 3 over the trained models in 2 in a black-box setup.

[1]The repository is available at: https://mega.nz/folder/U7lhARbY#0gZ8kghKYXrrDg_nvfPH7A

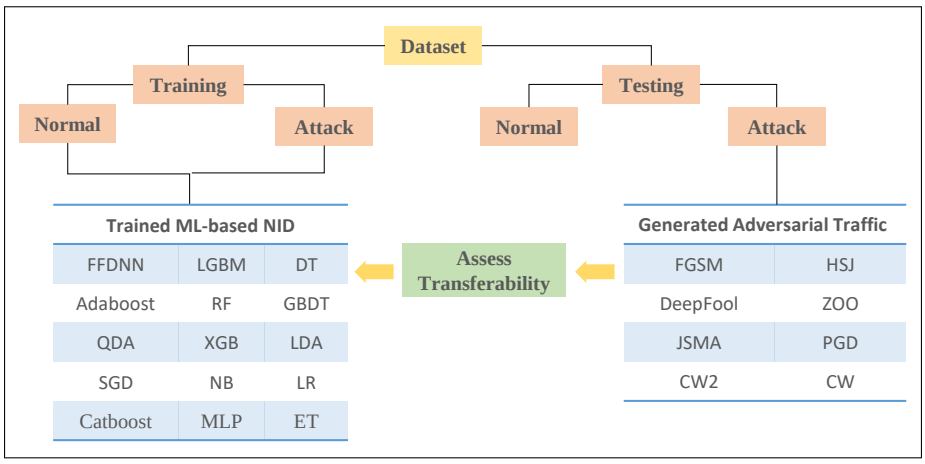

Figure 6.2 ML-NIDS Resilience to Adversarial Attacks

6.6.5. Evaluation Metrics

Models Performance

The confusion matrix helps assess a binary model's performance, detailing true positives (correctly classified attacks), false positives (normal instances misclassified as attacks), true negatives (correctly identified normal instances), and false negatives (attacks classified as normal). These four terms are used to derive the following evaluation metrics:

Metric	Definition	Formula
Accuracy	Ratio of correctly classified instances.	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Ratio of correctly classified positive instances out of all instances classified as positive.	$\frac{TP}{(TP+FP)}$
Recall	Ratio of actual positive cases correctly identified.	$\frac{TP}{TP+FN}$
F-1	Harmonic mean of precision and recall.	$\frac{2\cdot(Precision\times Recall)}{Precision+Recall}$

Table 6.5 Performance Metrics for Binary Classification

In evaluating imbalanced datasets, particularly in network anomaly detection scenarios where 'attack' instances are rare compared to 'normal' instances, standard accuracy metrics can be misleading. Instead, we prioritize metrics like balanced accuracy, recall, precision, and F1-score. Balanced accuracy offers a more reliable measure of the model's overall performance across all classes, regardless of their frequencies. In assessing the impact of adversarial examples on NIDS models, we prioritize attack class recall and normal class precision. Attack class recall measures the classifier's ability to capture most attack instances. High attack class recall indicates effective detection of most attack instances, despite adversarial manipulations. Normal class precision reflects the classifier's accuracy in identifying normal instances. Lower precision indicates misclassification of attacks or adversarial instances as normal, indicating evasion of detection. Finally, we favor the F1-score as it combines precision and recall, offering a balanced view of overall performance. Specifically, we use the macro F1-score, averaging scores for each

class without weighting by frequencies, ensuring equal contribution from each class for a reliable measure of performance.

Adversarial Attacks Performance

- Attack Success Rate (ASR): Raito of adversarial examples that are able to fool the model and cause it to make an incorrect prediction. A lower success rate indicates that the model is more resilient against adversarial examples, as fewer examples are able to fool the model.

$$ASR = \frac{number\ of\ successful\ AEs}{total\ number\ of\ AEs} \tag{6.2}$$

6.7. Resilience Evaluation of ML-based NIDS to Adversarial Attacks

In this section, we delve into a comprehensive assessment of the resilience of ML-based NIDS in the face of transferable adversarial attacks. This evaluation encompasses two crucial perspectives: the models' performance when exposed to clean test data and their behavior under the influence of adversarial test data.

6.7.1. Models Performance Over Clean Test Data

Table 6.6 displays the performance evaluation results of the models under an adversarial-free environment and an adversarial setting. Firstly, balanced accuracy is useful for imbalanced data, equally weighting normal and attack instances, providing an unbiased assessment of the model's ability to detect attacks. Ensemble and Decision Tree-Based methods (Catboost, DT, ET, RF, XGBoost) achieve high scores (0.98), while linear (SGD, LR, LDA) and probabilistic models (NB, QDA) perform below 0.83, with QDA at 0.66. Secondly, the macro F1 score, the average of F1 scores for each class, reflects overall performance balance across classes. Ensemble and Decision Tree-Based methods achieve high F1-scores of 0.98, while MLP, FFDNN, GBDT at 0.95. Linear and probabilistic models score lower, between 0.76 and 0.46.

Overall, the excellent performance of ensemble tree-based methods can be attributed to their ability to mitigate overfitting, capture complex relationships in network data, and maintain resilience to noise and outliers, offering resilience and flexibility by aggregating predictions from multiple base learners. On the other hand, Linear models like SGD, LR, LDA, and probabilistic models like NB and QDA exhibit limitations in capturing complex, nonlinear relationships in data. This limitation can hinder their ability to distinguish between normal and attack instances, especially in high-dimensional and nonlinear datasets.

Model	Balanced Accuracy			Precision (Normal)			Recall (Attack)			Macro F1-score		
	clean	adversarial	Drop	clean	adversarial	Drop	clean	adversarial	Drop	clean	adversarial	Drop
QDA	0.66	0.48	0.18	0.98	0.68	0.3	0.97	0.61	0.36	0.46	0.43	0.03
FFDNN	0.95	0.78	0.17	0.98	0.84	0.14	0.92	0.57	0.35	0.95	0.81	0.14
RF	0.98	0.82	0.16	0.99	0.87	0.12	0.97	0.65	0.32	0.99	0.86	0.13
Catboost	0.98	0.84	0.14	0.99	0.88	0.11	0.97	0.69	0.28	0.98	0.87	0.11
DT	0.98	0.84	0.14	0.99	0.88	0.11	0.97	0.69	0.28	0.98	0.87	0.11
ET	0.98	0.84	0.14	0.99	0.88	0.11	0.97	0.68	0.29	0.98	0.87	0.11
XGB	0.98	0.86	0.12	0.99	0.9	0.09	0.97	0.74	0.23	0.98	0.89	0.09
Adaboost	0.93	0.84	0.09	0.98	0.9	0.08	0.93	0.75	0.18	0.89	0.85	0.04
GBDT	0.96	0.87	0.09	0.99	0.91	0.08	0.94	0.76	0.18	0.95	0.89	0.06
LR	0.81	0.75	0.06	0.94	0.86	0.08	0.78	0.67	0.11	0.75	0.75	0
MLP	0.95	0.89	0.06	0.98	0.92	0.06	0.91	0.8	0.11	0.96	0.91	0.05
LGBM	0.96	0.91	0.05	1	0.95	0.05	0.98	0.87	0.11	0.92	0.9	0.02
SGD	0.83	0.78	0.05	0.95	0.88	0.07	0.82	0.73	0.09	0.76	0.77	-0.01
LDA	0.69	0.66	0.03	0.88	0.78	0.1	0.43	0.36	0.07	0.72	0.68	0.04
NB	0.7	0.67	0.03	0.89	0.8	0.09	0.54	0.48	0.06	0.69	0.68	0.01
Average	0.89	0.79	0.1	0.97	0.86	0.11	0.87	0.67	0.2	0.86	0.8	0.06

Table 6.6 Baseline Models Performance over Clean vs Adversarial Data

6.7.2. Models Performance: Clean vs. Adversarial

Creating adversarial instances usually requires knowledge of the target ML model's architecture, but these examples often deceive other models as well. In this section, we generate adversarial samples using eight techniques against the FFDNN model, add them to the test data with class labels as 1, matching the attack instances, and then evaluate the models' performance compared to baseline performance on clean data only.

As can be seen in Table 6.6, the models show a significant performance drop on adversarial data compared to clean data. The average balanced accuracy decreases from 0.89 to 0.79. QDA, FFDNN, and RF see substantial drops (0.16 to 0.18), while Catboost, DT, ET, and XGB experience decreases of 0.12 to 0.14. In contrast, LR, MLP, LGBM, SGD, LDA, and NB show minimal drops of 0.03 to 0.06.

The average recall for the attack class drops significantly by 20%, from 0.87 on clean data to 0.67 on adversarial data. This indicates a decreased ability to correctly identify attacks, leading to misclassification of perturbed instances as normal. The recall metric highlights the model's resilience to manipulation attempts intended to deceive it into misclassifying attacks as normal. For QDA, FFDNN, and RF models, recall decreases significantly, with reductions ranging from 0.36 to 0.32, indicating high vulnerability to adversarial manipulations. The ET, CatBoost, DT, XGB, Adaboost, and GBDT models experience moderate recall declines, with losses ranging from 0.29 to 0.18. In contrast, the LR, MLP, LGBM, SGD, LDA, and NB models show only minor decreases in recall, with reductions ranging from 0.06 to 0.11, demonstrating better resilience against such manipulations.

As can be seen in Table 6.6, the average precision of the normal class decreases by 11%, from 0.97 on clean data to 0.86 on adversarial data. This signifies a reduction in the model's accuracy in correctly identifying normal instances. The decrease suggests that more instances that are actually attacks or anomalies are incorrectly classified as normal, attributed to adversarial manipulations. QDA shows the highest vulnerability with a decrease of over 30%. Adaboost, GBDT, LGBM, LR, MLP, RF, SGD, and XGB show decreases of around 10% or less, indicating

greater resilience. In contrast, Catboost, DT, ET, FFDNN, LDA, and NB display notable decreases of over 10% in precision.

6.7.3. Models Adversarial Resilience

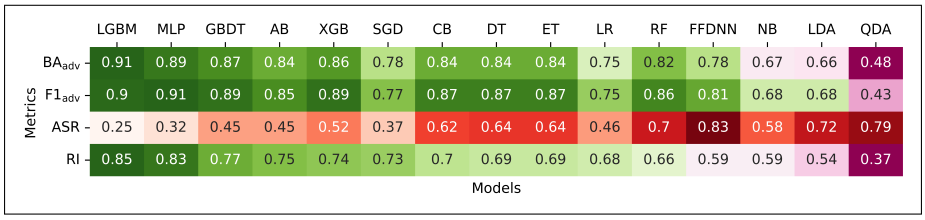

Figure 6.3 Baseline Models Resilience Index

We use our Resilience Index (RI) introduced in Section 6.4 to evaluate the resilience of ML-based NIDS models against adversarial attacks. RI combines key performance indicators: balanced accuracy (BA_{adv}), macro F1 score ($F1_{adv}$), and Attack Success Rate (ASR), providing a holistic view of the model's resilience. BA_{adv} and $F1_{adv}$ assess detection accuracy and resilience under adversarial conditions, while ASR measures vulnerability to evasion. RI offers a unified metric balancing detection accuracy and resilience against adversarial manipulation, aiding in the selection of the most effective NIDS models.

Top Performing Models (High RI): Models like LGBM, MLP, and GBDT exhibit high RI values due to their complex architectures and ability to capture intricate patterns. These top-performing models are promising for NIDS deployment, effectively detecting intrusions even with adversarial attacks. Techniques such as feature engineering, regularization, or ensembling may further enhance their resilience.

Mid-Range Models (Moderate RI): Models like Catboost, DT, ET, RF, and LR have moderate RI values. While they perform reasonably well, they may have higher ASR compared to top-performing models. Decision tree-based models are sensitive to small input perturbations and prone to overfitting, leading to poor generalization and vulnerability to adversarial attacks. LR's linearity limits its ability to capture complex, non-linear relationships, making it susceptible to outliers. Deploying these models for NIDS requires additional defenses or complementary models to enhance their resilience.

Lowest Performing Models (Low RI): Models like NB, LDA, and QDA achieve the lowest RI values among the baseline models. These models are all based on strong assumptions about the underlying data distribution, which may not hold true in the presence of adversarial manipulations. They may struggle to adapt to complex, non-linear relationships in the data, leading to reduced resilience. Deploying them in NIDS systems may lead to suboptimal threat detection and higher risk of successful adversarial evasion.

6.7.4. Adversarial Attacks Performance

We utilize various statistical metrics to assess the performance and stability of attacks across diverse models. The mean success rate offers an average performance measure, with higher values

indicating greater overall success. The median success rate, less affected by extreme values, reveals effectiveness; a high median suggests reliability, while a low median indicates variability. Standard deviation gauges success rate variability around the mean: higher values denote more variability, lower values signify consistency. The range, showing the spread between highest and lowest success rates, indicates variability: wider ranges suggest greater variability, narrower ranges imply more consistent performance. Collectively, these metrics offer a comprehensive analysis of attack effectiveness and stability across models.

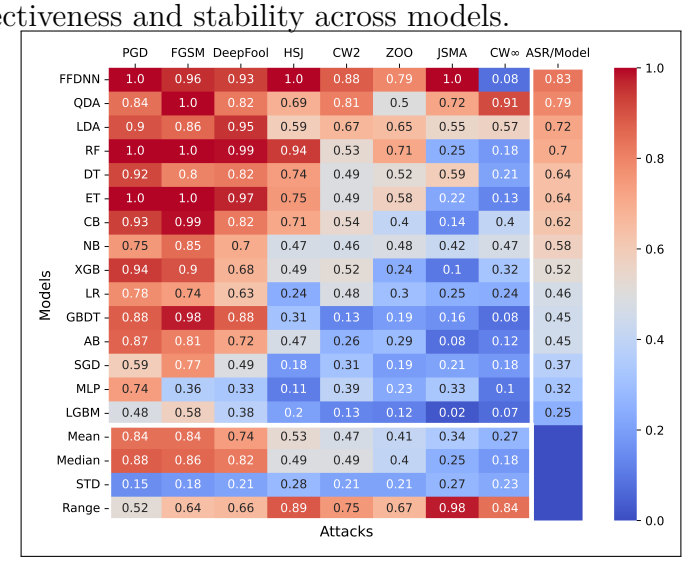

Figure 6.4 Attack Success Rates Against Baseline Models

In Figure 6.4, the PGD, FGSM, and DeepFool attacks show median success rates between 0.88 and 0.82, demonstrating high performance consistency. PGD's iterative approach refines perturbations based on loss function gradients, effectively overcoming defenses and exploiting common model weaknesses. FGSM uses target model gradients to efficiently craft perturbations by computing loss function gradients with respect to input data, maximizing loss and causing misclassification. DeepFool's iterative method calculates minimal perturbations to linearize decision boundaries, creating adversarial examples close to the original input. Its model-agnostic nature allows it to be applied across various models and architectures without needing specific modifications.

HSJ and CW2 exhibit moderate effectiveness with average success rates of 0.53 and 0.47, respectively, with HSJ showing less consistency. HSJ's heuristic search strategy explores the input space for perturbations but can be inconsistent due to not always finding optimal solutions. CW attacks, optimized for minimal perturbations, may not transfer well if models are less sensitive to small changes. ZOO's success rate of 0.41 indicates lower effectiveness. As a black-box technique, it estimates gradients numerically rather than analytically, leading to less finely tuned adversarial examples that don't exploit the model's vulnerabilities as effectively. JSMA and CW∞ demonstrate the least effectiveness and low consistency, with median success rates of 0.25 and 0.18, respectively. JSMA, notably inconsistent like HSJ, relies on saliency maps from model gradients, which vary significantly across models due to differing decision boundaries and feature sensitivities. Thus, perturbations for one model may not deceive another effectively.

6.8. Evaluation of Adversarial-Resilient NIDS Framework

In this section, we explore ensemble learning and adversarial training as defenses against adversarial examples, examining their limitations and efficacy. We also present our solution: an adversarially trained ensemble. This approach combines the benefits of both techniques, demonstrating superior performance and resilience against adversarial attacks.

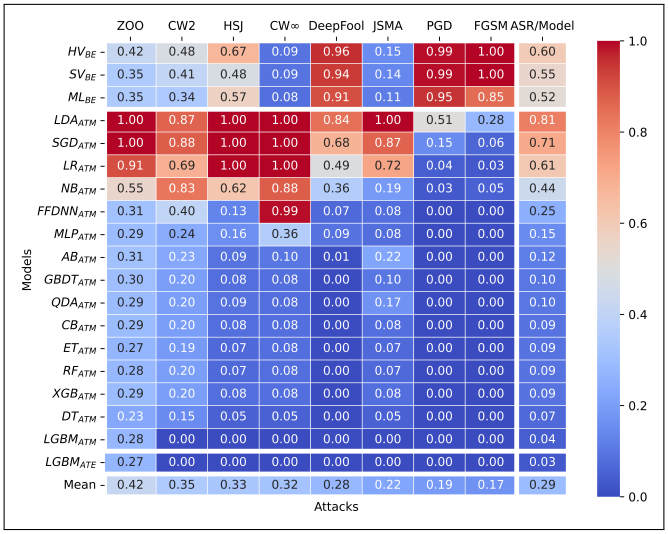

Figure 6.5 Attack Success Rates Against Defense Models

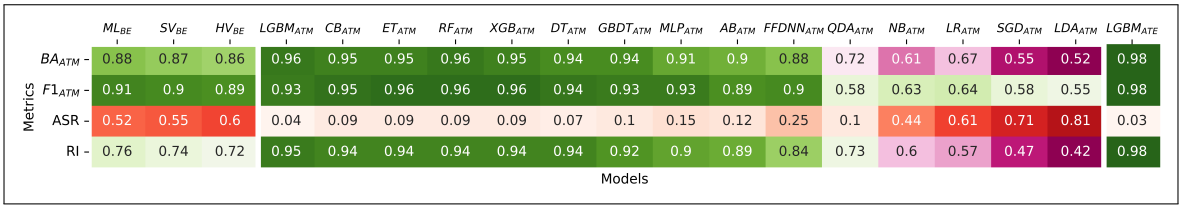

Figure 6.6 Defense Models Resilience Index

6.8.1. Baseline Ensemble (BE)

Ensemble learning combines multiple models to improve predictive performance by leveraging their diverse strengths. In adversarial detection, it may enhance accuracy by aggregating multiple models with different detection strategies, making it harder for adversaries to exploit single-model vulnerabilities.

We implemented three ensemble techniques to assess their effectiveness against adversarial examples: hard voting (HV), soft voting (SV), and meta-learning (ML). In hard voting, the final prediction is based on the majority vote among models. In soft voting, class probabilities are averaged or weighted, and the class with the highest probability is chosen. Meta-learning trains a model on the outputs or features of base models to enhance performance by leveraging their diversity. We selected RF, CB, DT, ET, XGB, GBDT, LGBM, FFDNN, and MLP based on their high balanced accuracy rates over clean data, as presented in Table 6.6.

While all ensemble models achieved a balanced accuracy of 0.98 on clean data, their average balanced accuracy on adversarial data was 0.87, as depicted in Figure 6.6, lower than individual models like LGBM and MLP, which scored 0.90, as illustrated in Figure 6.3. Meta-learning had the lowest attack success rate at 0.52, followed by soft voting at 0.55 and hard voting at 0.60 (Figure 6.6). However, individual models like LGBM, MLP, and GBDT had lower attack success rates (0.25 to 0.45). The ensemble models averaged an RI of 0.74, lower than individual models LGBM (0.85), MLP (0.83), and GBDT (0.77).

Overall, despite high balanced accuracy on clean datasets, ensemble methods do not improve resilience against transferable adversarial examples. They often show lower accuracy and higher attack success rates compared to individual models like LGBM, MLP, and GBDT, falling behind in effectiveness and resilience as measured by RI. Thus, ensemble methods may not significantly enhance resilience to adversarial examples in NIDS compared to individual models.

6.8.2. Adversarially Trained Models (ATM)

Adversarial training involves training a model on both clean and adversarial examples to enhance resilience against attacks. Exposure to adversarial examples helps the model focus on generalizable features and ignore irrelevant ones. However, excessive exposure may cause the model to prioritize recognizing adversarial perturbations, decreasing performance on clean examples and risking overfitting to specific perturbations, which can hinder generalization. We retrained the baseline models on the same training dataset, augmented with adversarial examples. The models were trained as multiclassifiers to distinguish between normal, attack, and adversarial instances. Table 6.3 shows the class distribution in the training dataset.

Although adversarial training was initially expected to decrease overall performance, the majority of adversarially trained models actually exhibit higher scores across all metrics. They achieve an average balanced accuracy of 0.82 and an F1 score of 0.82, compared to 0.79 and 0.80 for baseline models. Attack success rates drop by 44.46%, from 0.56 to 0.25. Using the RI as a comprehensive metric, adversarially trained models score 0.8, outperforming baseline models at 0.68. Overall, adversarial training slightly enhanced performance and significantly reduced attack success rates, suggesting it effectively improves model resilience without compromising performance.

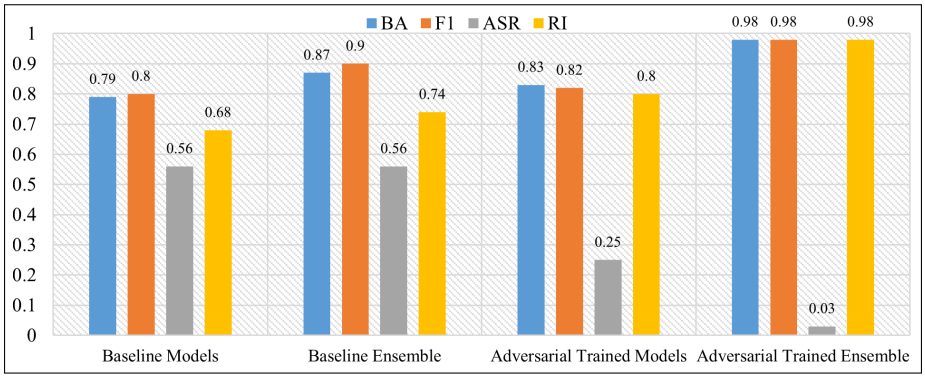

Figure 6.7 Comparison Baseline Models vs. Defense Models

6.8.3. Adversarially Trained Ensemble (ATE)

Adversarial training shows potential in detecting adversarial examples and can be enhanced by integrating other defense techniques. This combined approach strengthens the model's ability to identify and counter adversarial attacks.We integrate adversarial training with ensembling to increase overall resilience. This approach diversifies model perspectives, boosting resilience against attacks by leveraging varied data interpretations. Ensemble redundancy enhances security by reducing the risk of all models falling for the same adversarial example. Leveraging the ensemble's varied strengths improves performance and generalization across clean and adversarial examples, providing a multi-layered defense.

Based on the results in Figure 6.6, LGBM showed the highest RI score of 0.95 among adversarially trained models. We chose LGBM for constructing a homogeneous ensemble using bagging and boosting techniques. Bagging trains multiple models on different subsets of the training data and aggregates their predictions. Boosting iteratively trains models, focusing on misclassified instances to improve performance, resulting in a strong final model.

As shown in Figure 6.6, our proposed Adversarial Trained Ensemble (ATE) model with LGBM base demonstrates superior performance. It significantly reduces the attack success rate to 0.03, compared to 0.56 for baseline ensembles and 0.25 for adversarially trained models. Additionally, it achieves the highest balanced accuracy and F1 score of 0.98, outperforming all other models across performance metrics.

As illustrated in Figure 6.6, our ATE achieves an outstanding RI of 0.98, outperforming all other model groups. The RI increases from 0.68 for baseline models to 0.74 for baseline ensembles, then to 0.80 for adversarially trained models, and peaks at 0.98 for the ATE. These results highlight the ATE's potential for effectively detecting and mitigating adversarial examples, reducing their transferability.

Attack	Precision	Recall	F1-Score	Support
CW2	0.94	0.94	0.94	25653
CW∞	0.92	0.95	0.93	25653
DeepFool	1.00	1.00	1.00	25653
FGSM	1.00	1.00	1.00	25653
HSJ	0.96	0.93	0.95	25653
JSMA	0.95	0.94	0.94	25653
PGD	1.00	1.00	1.00	25653
ZOO	1.00	1.00	1.00	25653
Accuracy			0.97	205224
Macro Avg	0.97	0.97	0.97	205224
Weighted Avg	0.97	0.97	0.97	205224

Table 6.7 Adversarial Attacks Classifier Performance

6.8.4. Performance of Adversarial Attacks Classifier

The Adversarial Attacks Classifier using an LGBM model achieved an overall accuracy of 0.97. Table 6.7 shows the classification report, highlighting varying success in classifying different attacks. Attacks like DeepFool, FGSM, PGD, and ZOO achieved an F1 score of 1.00, indicating

strong performance in correctly classifying these instances. In contrast, CW2, CW∞, HSJ, and JSMA had F1 scores ranging from 0.93 to 0.95.

6.9. Impact of Defense Strategies on Adversarial Attacks Efficacy

Figure 6.8 reveals distinct patterns in the effectiveness of adversarial attacks on different defensive models. We can categorize the attacks into two primary groups based on their success rates and the impact of defense mechanisms.

Group 1, including PGD, FGSM, DeepFool, HSJ, and CW2, shows a marked reduction in success with adversarial training, with success rates dropping to 0.05 for PGD and 0.03 for FGSM. However, ensemble learning is counterproductive, increasing success rates to 0.98 for PGD and 0.95 for FGSM. This suggests adversarial training is highly effective for this group, while ensemble learning may introduce exploitable vulnerabilities.

Group 2, comprising ZOO, JSMA, and CW∞, responds differently. Ensemble learning effectively mitigates these attacks, reducing success rates significantly—for example, JSMA to 0.13 and CW∞ to 0.09. Conversely, adversarial training increases success rates for ZOO and CW∞ to 0.44 and 0.39, respectively. This suggests ensemble learning provides a more robust defense for these attacks, while adversarial training may inadvertently increase susceptibility.

The analysis highlights the varied effectiveness of defense mechanisms against adversarial attacks. Adversarial training significantly reduces success rates for Group 1 attacks but is less effective for Group 2, where ensemble learning performs better. Conversely, ensemble learning worsens resilience for Group 1 attacks. This underscores the need to tailor defense strategies to specific attacks, leveraging each method's strengths to improve overall model resilience.

Our proposed Adversarial Trained Ensemble enhances resilience against all types of adversarial attacks. By combining adversarial training with ensemble methods, we achieve dramatic reductions in attack success rates, with zero success rates for all attacks except ZOO, which is reduced to 0.27. This approach leverages both defense mechanisms' strengths, ensuring robust and comprehensive protection against adversarial attacks.

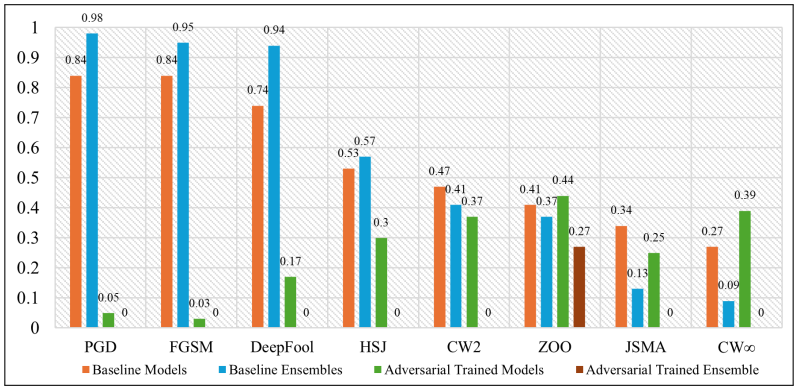

Figure 6.8 Impact of Defense Strategies on Adversarial Attacks Efficacy

6.10. Discussion

In this section, we discuss the implications of our results, highlighting the strengths and limitations of various attacks, models and defenses, and provides insights for future research and practical deployment.

Our analysis revealed that ensemble and decision tree-based methods (e.g., Catboost, DT, ET, RF, XGBoost) demonstrate superior performance on clean data due to their ability to capture complex relationships and mitigate overfitting. However, these models, along with others, exhibit significant performance degradation when subjected to adversarial attacks. The substantial drop in balanced accuracy and macro F1-score for models like QDA, FFDNN, and RF under adversarial conditions highlights the vulnerability of these approaches to adversarial manipulation. Interestingly, linear models (e.g., SGD, LR, LDA) and probabilistic models (e.g., NB, QDA) display minimal performance drops, which could be attributed to their simpler architectures. Despite their robustness to perturbations, these models struggle with the inherent complexity of network traffic data, reflected in their lower overall performance metrics.

Our evaluation of various adversarial attack techniques shows that iterative methods like PGD, FGSM, and DeepFool are highly effective and consistent across different models. These attacks exploit the models' vulnerabilities by refining perturbations, often resulting in successful evasion. The lower success rates of attacks like ZOO, JSMA, and CW∞ suggest variability in their ability to deceive models, pointing to potential areas for enhancing model robustness.

The exploration of ensemble learning and adversarial training as defense mechanisms reveals nuanced insights. Ensemble models, while achieving high balanced accuracy on clean data, do not significantly enhance resilience against adversarial attacks. In fact, they often show lower resilience compared to individual models like LGBM and MLP. This indicates that simply aggregating predictions from multiple models does not necessarily confer robustness to adversarial perturbations. Adversarial training, on the other hand, demonstrates a notable improvement in model resilience. The retrained models exhibit higher balanced accuracy and F1 scores on adversarial data, with a significant reduction in attack success rates. This suggests that adversarial training effectively strengthens the models' ability to generalize and recognize adversarial patterns. However, the potential risk of overfitting to specific perturbations warrants careful consideration and further research.

Our proposed Adversarially Trained Ensemble (ATE) model, combining adversarial training with ensembling, exhibits superior performance and resilience. The ATE model significantly reduces attack success rates and achieves the highest balanced accuracy and F1 scores across all evaluated models. This approach leverages the strengths of both defense mechanisms, providing a robust and comprehensive solution for detecting and mitigating adversarial examples.

The impact of defense strategies on adversarial attacks reveals distinct efficacy patterns. For Group 1 attacks (PGD, FGSM, DeepFool, HSJ, CW2), adversarial training significantly reduces success rates, highlighting its effectiveness. However, ensemble learning increases these attacks' success rates, potentially introducing vulnerabilities. Conversely, for Group 2 attacks (ZOO, JSMA, CW∞), ensemble learning effectively reduces success rates, while adversarial training

increases susceptibility. This divergence underscores the need for tailored defenses against specific attacks. The Adversarially Trained Ensemble (ATE) model stands out, demonstrating superior performance by leveraging both methods' strengths, achieving near-zero success rates across most attack types, and providing robust, comprehensive protection. This combined approach underscores the importance of multifaceted defense strategies in enhancing NIDS resilience against diverse adversarial threats.

For model design, the insights stress the importance of selecting appropriate models based on the trade-offs between performance on clean data and resilience to adversarial attacks. For defense strategies, the nuanced effectiveness of defense mechanisms against different attack types suggests that a one-size-fits-all approach is insufficient. Tailored defenses are necessary for different attack vectors. The effectiveness of the ATE model implies that integrating multiple defense strategies can offer comprehensive protection, reducing the success rates of various adversarial attacks.

While previous studies often focused on individual defense mechanisms or specific attacks, this study's strengths include a comprehensive evaluation of diverse models and attacks, detailed analysis of defense mechanisms, and the introduction of a novel ATE model. This model, which combines the strengths of different defense strategies, represents a significant and novel contribution not extensively explored in prior research.

6.11. Conclusion

In this chapter, we provided a comprehensive assessment of the performance of 8 adversarial attacks, the resilience of 15 different detection models, and the effectiveness of three defense methods. We introduced the Resilience Index as a metric for evaluating the resilience of ML-based NIDS models against adversarial attacks. This index combines performance metrics with the attack success rate to offer a comprehensive assessment of the model's resilience. Employing various statistical metrics, we analyzed attacks performance and stability across different models. Additionally, we investigated ensemble learning and adversarial training as defense mechanisms against adversarial examples and compared their strengths and weaknesses. Importantly, we introduced the adversarial-resilient NIDS, a multi-layered framework comprising an Intrusions and Adversarial Detector along with an Adversarial Attack Classifier. Leveraging an Adversarially Trained Ensemble, the detector demonstrates proficiency in accurately identifying eight types of adversarial attacks. Thorough experimental evaluations on the NF-UQ-NIDS dataset—a recent standardized network traffic dataset gathered from diverse environments including traditional and IoT networks featuring twenty types of cyber attacks—alongside comparative analyses with state-of-the-art defenses, the ATE detector demonstrates remarkable efficacy. It significantly reduces the attack success rate from 0.41 to just 0.03, while achieving an outstanding overall performance rating of 0.98. Furthermore, the framework incorporates an adversarial attack classifier, achieving a classification accuracy of 0.97 in determining the type of adversarial attack employed.

Chapter 7. Conclusion

7.1. Summary

In this chapter, we provide a discussion of the thesis and outline our future work. Finally, we highlight some challenges observed during our research and potential opportunities for further advancements.

7.2. Discussion

In this Ph.D. thesis, we introduced comprehensive threat models for ML-based NIDS and investigated the realism and impact of generic adversarial attacks on DL-based NIDS. Additionally, we conducted a comprehensive assessment encompassing the performance of attacks, the resilience of detection models, and the effectiveness of defense approaches. Finally, we presented the Adversarial-Resilient NIDS, a lightweight framework for detecting and classifying adversarial attacks.

7.2.1. Threat Modeling Analysis for ML-Based NIDS: Uncovering Hidden Risks

In chapter 3, we provided comprehensive threat models for ML-based NIDS. Our models enable proactive defense by anticipating and addressing potential threats before they occur. These models enable for the prioritization of efforts based on the severity and likelihood of different attacks by providing not only vulnerabilities identification but also risks quantification. This ensures efficient resource allocation and focused security measures. Overall, our models contribute to developing more robust ML-based NIDS by implementing the recommended countermeasures and working on areas for security improvements in the detection algorithms and system architectures.

Using STRIDE and Attack Tree methodologies, we provided a comprehensive security perspective, ensuring no threat vector is overlooked. The Attack Tree technique maps potential threats from ML algorithm vulnerabilities, while STRIDE examines data flow to identify additional technical threats. We identified and categorized 46 distinct threats, revealing vulnerabilities at various levels of ML algorithms and detection system data flows. Our models highlight how ML-based NIDS can be compromised and the diverse avenues for potential attacks. This facilitates the development of targeted and effective hardening measures. Our threat models

offer a foundational framework for future research on addressing security threats in ML-based NIDS.

Despite the thoroughness of our threat models, it is crucial to acknowledge that the dynamic nature of cyber threats necessitates continuous updating and validation of these models. As adversaries evolve their tactics and ML algorithms advance, new vulnerabilities may emerge, requiring ongoing research and adaptation of defense mechanisms. This iterative approach ensures that the threat models remain relevant and effective in safeguarding ML-based NIDS against an ever-changing landscape of cyber threats.

7.2.2. Adversarial Evasion Attacks on ML-based NIDS: Reviewing Current Knowledge

In chapter 4, we presented a more detailed and structured categorization of the literature than previously available by conducting a thorough analysis of 112 research studies in the field. We offered an extensive overview and evaluation of existing knowledge, highlighted gaps in current research, and recommended potential research directions and practical considerations for enhancing the resilience of NIDS against adversarial attacks.

we surveyed the up-to-date literature and synthesized the existing body of work into three main areas: generating tailored adversarial examples for ML-based NIDS, evaluating the resilience of these systems, and developing defensive mechanisms. We critically examined and categorized these studies, discussing their strengths and limitations, and identified potential avenues for improvement. Additionally, we conducted a meta-data analysis to uncover research trends and patterns, derived key findings, and addressed relevant research questions regarding the shortcomings of current approaches. Based on this analysis, we suggested enhancements and provided insights for future research and development in the field.

7.2.3. Evaluating the Realism of Adversarial Attacks: Bridging Theory and Reality

In chapter 5, we bridged the gap between theoretical adversarial attacks and real-world network scenarios, introducing the "Unrealism Index" to evaluate adversarial attacks' realism. The Unrealism Index is a novel metric to measure how adversarial manipulations diverge from real network traffic. We conducted a comprehensive evaluation of a wide range of adversarial example attacks across different datasets, providing a rich resource for understanding their performance and realism.

From our research, we now have a deeper understanding of the limitations and unrealistic aspects of current adversarial example generation methods in the realm of network traffic. We revealed the disconnect between the theoretical efficiency of common adversarial attacks and their practical execution against DL-based NIDS, due to their often unrealistic manipulations. Our work underscored the necessity for adversarial examples that are effective, realistic, and compliant with network traffic constraints. This new knowledge propels future research towards more realistic and practical threat models, emphasizing problem-space perturbations and black-box approaches that more accurately represent the capabilities of real-world adversaries.

Although our proposed Unrealism Index employs metrics to exclude unrealistic attacks, compliance with these metrics does not guarantee the complete realism of the adversarial examples. There may still be instances where adversarial traffic passes the validation metrics but does not accurately reflect real-world network conditions. This highlights the need for ongoing refinement of the Unrealism Index and the development of more sophisticated validation techniques that better capture the complexities and nuances of actual network traffic.

7.2.4. Enhancing ML-Based NIDS Resilience: Shielding the Shield

In chapter 6, we did several key areas of enhancement in the field. Through comprehensive experimental evaluations, we assessed the resilience of 15 different ML-based NIDS models against 9 types of adversarial attacks, including both white-box and black-box attacks. We used the NF-UQ-NIDS dataset features recent network traffic data from various traditional and IoT networks, and encompasses twenty types of cyber attacks. This comprehensive experimental setups ensures the generalizability and applicability of our findings to real-world scenarios. We provided the Resilience Index a standardized measure to compare the ML-based NIDS's resilience for informed deployment decisions. Additionally, We identified the most consistently performing attacks. Defense' efficiency can be optimized by designing targeted countermeasures to the attacks that exhibit generalized effectiveness across all NIDS models. We performed a comprehensive assessment of ensemble learning and adversarial training and highlighted their strengths and weaknesses. Finally, we introduced the Adversarial-Resilient NIDS framework, which employs a multi-layered approach to robustly detect and classify adversarial attacks. Our framework maintains both high performance and resilience.

Our analysis revealed key findings on the performance and resilience of machine learning models and defense strategies against adversarial attacks in NIDS. Ensemble and decision tree-based methods (e.g., Catboost, DT, ET, RF, XGBoost) excel in clean data scenarios due to their ability to capture complex relationships, yet they suffer significant performance degradation under adversarial conditions. In contrast, linear models (e.g., SGD, LR, LDA) and probabilistic models (e.g., NB, QDA) display minimal performance drops due to their simpler architectures but struggle with the inherent complexity of network traffic data. Iterative attack techniques like PGD, FGSM, and DeepFool consistently exploit model vulnerabilities, while attacks like ZOO, JSMA, and CW∞ show variable success rates. Ensemble learning does not significantly enhance resilience against adversarial attacks, often showing lower resilience compared to individual models. However, adversarial training notably improves resilience, although it risks overfitting to specific perturbations. Our proposed Adversarially Trained Ensemble (ATE) model, which combines adversarial training with ensembling, demonstrates superior performance and resilience, reducing attack success rates and achieving the highest balanced accuracy and F1 scores. Defense strategies show distinct efficacy patterns, with adversarial training significantly reducing success rates for Group 1 attacks (PGD, FGSM, DeepFool, HSJ, CW2), and ensemble learning effectively reducing success rates for Group 2 attacks (ZOO, JSMA, CW∞).

These findings imply that selecting appropriate models involves trade-offs between performance on clean data and resilience to adversarial attacks. Defense strategies must be tailored to specific attack vectors, as a one-size-fits-all approach is insufficient. The effectiveness of the ATE model suggests that integrating multiple defense strategies can provide comprehensive protection, reducing the success rates of various adversarial attacks. However, our Adversarial-Resilient framework, while covering covers eight types of adversarial attacks, this scope, though extensive, is not exhaustive. Other unseen or emerging adversarial methods not included in this study could evade detection highlighting the need for expansion of the detection model to address the unknown and evolving attacks. Our study's strengths include a comprehensive evaluation of diverse models and attacks, detailed analysis of defense mechanisms, and the introduction of the novel ATE model.

7.3. Future Work

In future work, we will integrate Explainable Artificial Intelligence (XAI) techniques to enhance the interpretability of the Adversarial-Resilient NIDS framework. XAI will provide insights into our framework's decision-making process and identify key features for adversarial detection. This integration increases transparency and aids in understanding adversarial attacks by clarifying why certain decisions are made. By using XAI to analyze feature importance and decision patterns, we can identify specific aspects of the framework that adversarial attacks exploit, thus helping us uncover our framework's vulnerabilities. This understanding allows us to diagnose weaknesses more effectively and implement targeted improvements to enhance the framework's robustness.

Additionally, we will use GANs to create diverse sophisticated and realistic adversarial examples to enhance the robustness and adaptability of our framework. This approach ensures better preparation and defense against a broader range of evolving attacks.

One of the most intriguing aspects of our future work is the comparative evaluation of supervised and unsupervised models in terms of their resilience against adversarial attacks. By assessing the strengths and weaknesses of each approach, we aim to gain valuable insights into their responses to various adversarial attacks. This comparative study will guide improvements in enhancing ML-based NIDS models' resilience, allowing us to develop hybrid or enhanced approaches that leverage the best features of both supervised and unsupervised models.

7.4. Challenges and Research Opportunities

Beyond our contributions in this thesis to enhancing ML-based NIDS against adversarial attacks, several challenges remain. Addressing these will pave the way for more resilient and efficient ML-based NIDS capable of mitigating evolving adversarial threats.

Adversarial defenses often require complex, computationally intensive techniques to achieve high detection accuracy. This can result in significant computational overhead. Research should focus on finding optimal solutions that maximize performance and minimize overhead. This

can include utilizing lightweight algorithms, implementing efficient data processing methods, optimizing model architectures, using approximation techniques, and employing selective feature extraction.

Standardized metrics are essential for providing a common framework to consistently evaluate and compare the effectiveness of adversarial attacks and defenses. The absence of such metrics makes it challenging to assess the effectiveness of different approaches and compare results across studies. Research should focus on defining and validating metrics that capture various aspects of adversarial robustness, including detection rate, false positive rate, impact on network performance, and resilience to different types of attacks.

Generating adversarial examples is time-consuming process. This underscores the critical need for standardized and comprehensive datasets to streamline the assessment of NIDS defenses. These datasets must encompass a wide range of existing adversarial attacks, including realistic and sophisticated ones. These datasets are crucial for rigorously testing the robustness of models against adversarial attacks and for training them to recognize and mitigate such threats. The absence of such datasets limits the ability to generalize findings. Standardized datasets would enable consistent evaluation and comparison of defense mechanisms.

Non-ML-based NIDS, which often rely on signature-based or heuristic methods, are still widely used. However, their vulnerability to ML-based adversarial attacks is not investigated. How can these traditional systems be compromised by adversarial techniques typically targeting ML-based NIDS, and in what ways do these attacks differ.

Existing literature evaluates the impact of adversarial attacks on ML-based classification models and how their resilience can be enhanced. However, there is a notable gap in research on anomaly detection approaches, their resilience to adversarial attacks, and potential defense mechanisms. Investigating these aspects is crucial to uncover the strengths of both classification and anomaly detection approaches, which can then be combined to create a comprehensive defense against adversarial attacks.

7.5. Conclusion

In conclusion, we have made significant contributions to the field of ML-based NIDS by conducting comprehensive threat modeling using STRIDE and Attack Tree techniques to identify and mitigate potential vulnerabilities. Our extensive survey and meta-data analysis have provided valuable insights into research trends and highlighted areas for future exploration. We investigated the vulnerability of DL-based NIDS to various adversarial attacks, assessing their effectiveness and realism, and established "attack unrealism" to evaluate the practicality of these attacks in real-world scenarios. The introduction of the Resilience Index offers a standardized measure for assessing ML-based NIDS resilience, facilitating informed deployment decisions. Furthermore, our development of the Adversarial-Resilient NIDS framework, featuring a lightweight adversarially trained ensemble, demonstrates exceptional efficacy in detecting adversarial attacks while maintaining high accuracy on clean data. These advancements collectively enhance the robustness and reliability of ML-based NIDS.

Appendix A. Experiments Reproducibility

A.1. Introduction

This chapter ensures the reproducibility of the experiments conducted in this study. It includes details about the computer configuration, software packages, and specific information for each experiment, including datasets, model parameters, and code repositories.

A.2. Computer Configuration

All experiments were conducted on a computer with the following configuration:

- **Processor:** Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz 2.40 GHz
- **RAM:** 8 GB
- **Graphics Card:** NVIDIA GeForce RTX 3080
- **Storage:** 500 GB SSD
- **Operating System:** Windows 10 Home

A.3. Packages

The following software packages and versions were used across all experiments:

- **Python:** 3.8.8
- **TensorFlow:** 2.11.0
- **Keras:** 1.3.2
- **Pandas:** 0.25.0
- **NumPy:** 1.21.6
- **ART (Adversarial Robustness Toolbox):** 1.17.1
- **Scikit-learn:** 1.0.2
- **Joblib:** 1.3.2

A.4. Experiment 1: Realism vs. Performance for AEs Against DL-based NIDS

A.4.1. Datasets

We used two datasets for this experiment:

- **WSN-DS**: Contains 374,661 records with normal traffic and four DoS attacks: flooding, TDMA, grayhole, and blackhole [17].
- **BoT-IoT**: Includes 3.6 million records with normal IoT traffic and attacks such as DDoS, DoS, Keylogging, Data exfiltration, OS, and Service Scan [81]. Available at BoT-IoT Dataset.
- Detailed feature descriptions are in Tables B.1 and B.2.
- WSN-DS was split into 80% training and 20% testing using `scikit-learn`'s `train_test_split`. BoT-IoT used pre-split datasets from the authors.
- Labels: WSN-DS includes blackhole, grayhole, flooding, scheduling, and normal traffic. BoT-IoT includes DDoS, DoS, theft, reconnaissance, and normal traffic (Table 5.2).

A.4.2. Model Parameters

The parameters for the models used in this experiment are detailed in Table A.1 below.

Parameter	Value
No. of hidden layers	3
Layer 1	128 neurons
Layer 2	64 neurons
Layer 3	32 neurons
Dropout	0.25
Optimizer	ADAM
Activation function	ReLU and Sigmoid
Learning rate	0.01
Epoch	100
Batch Size	64

Table A.1 Feed-Forward DNN Model Parameters

A.4.3. Code Repository

The code for this experiment is available at:

- **Repository:** https://mega.nz/folder/U7lhARbY#0gZ8kghKYXrrDg_nvfPH7A

A.5. Experiment 2: Evaluating and Detecting AEs in ML-based NIDS

A.5.1. Datasets

We used the NF-UQ-NIDS dataset for this experiment:

- Dataset Records: It has 11,994,893 records (9,208,048 benign, 2,786,845 attack).
- Features Description: See Table B.3
- Subset Used: 6,338,509 records (81.8% normal, 18.2% attack).
- Sampling Rationale: Reduces computational load while preserving data diversity.
- Classification: Unified attack types into a single 'Attack' class.
- Splits: Stratified into 80% training and 20% testing subsets.
- Traffic Statistics: See Table 6.2 and Table 6.3.

A.5.2. Model Parameters

The parameters for the models used in this experiment are detailed in the Table A.2 below.

Model	**Parameters**
FF-DNN	Layers: [Dense(128, activation=relu), Dropout, Dense(64, activation=relu), Dropout, Dense(2, activation=softmax)], Optimizer: Adam
RF	n_estimators=100, criterion='gini', max_depth=None
DT	criterion='gini', splitter='best', max_depth=None
ET	c n_estimators=100, criterion='gini', max_depth=None
LR	penalty='l2', C=1.0, solver='lbfgs'
XGBoost	objective='binary:logistic', n_estimators=100, max_depth=3
Adaboost	base_estimator=None, n_estimators=50, learning_rate=1.0
NB	alpha=1.0, fit_prior=True, class_prior=None
LDA	solver='svd', shrinkage=None, priors=None
QDA	priors=None, reg_param=0.0, store_covariance=False, stor_covariances=None
SGD	loss='hinge', penalty='l2', alpha=0.0001
GBDT	n_estimators=100, learning_rate=0.1, max_depth=3, mi_samples_split=2
Catboost	iterations=100, depth=3, learning_rate=0.03, boosting_type= 'Ordered'
LGBM	n_estimators=100, learning_rate=0.1, max_depth=-1, boosting_type= 'gbdt'
MLP	hidden_layer_sizes= (100,) activation= 'relu' solver= 'adam' learning_rate_init= 0.001

Table A.2 Parameters of ML-based NIDS Models

A.5.3. Code Repository

The code for this experiment is available at:

- **Repository:** https://mega.nz/folder/QjMGhArI#mZElWXo98eDAMTPXSEbfog

Appendix B. Datasets Description

B.1. WSN-DS Dataset

Feature	Description
Node ID	A unique ID to distinguish the sensor node in any round and at any stage.
Time	The current simulation time of the node.
Is CH?	A flag with value of 1 for Cluster Head, 0 for normal node.
Who CH?	The ID of the CH in the current round.
RSSI	Received Signal Strength Indication between the node and its CH in the current round.
Distance to CH	The distance between the node and its CH in the current round.
Max distance to CH	The maximum distance between the CH and the nodes within the cluster.
Average distance to CH	The average distance between nodes in the cluster to their CH.
Current energy	The current energy for the node in the current round.
Energy consumption	The amount of energy consumed in the previous round.
ADV_CH send	The number of advertise CH broadcast messages sent to the nodes.
ADV_CH receive	The number of advertise CH messages received from CHs.
Join_REQ send	The number of join request messages sent by the nodes to the CH.
Join_REQ receive	The number of join request messages received by the CH from the nodes.
ADV_SCH send	The number of advertise TDMA schedule broadcast messages sent to the nodes.
ADV_SCH receive	The number of TDMA schedule messages received from CHs.
Rank	The order of this node within the TDMA schedule.
Data sent	The number of data packets sent from a sensor to its CH.
Data received	The number of data packets received from CH.
Data sent to BS	The number of data packets sent to the Base Station (BS).
Distance CH to BS	The distance between the CH and the BS.
Send Code	The cluster sending code.
Attack Type	It is a class of five values: Blackhole, Grayhole, Flooding, Scheduling, or normal.

Table B.1 Description of WSN-DS Dataset

B.2. BoT-IoT Dataset

Feature	Description
pkSeqID	Row Identifier
Proto	Textual representation of transaction protocols within network flow.
Saddr	Source IP address
Sport	Source port number
Daddr	Destination IP address
Dport	Destination port number
Seq	Argus sequence number
Stddev	Standard deviation of aggregated records
N_IN_Conn_P_SrcIP	Number of inbound connections per source IP.
Min	Minimum duration of aggregated records
state_number	Numerical representation of feature state
Mean	Average duration of aggregated records
N_IN_Conn_P_DstIP	Number of inbound connections per destination IP.
Drate	Destination-to-source packets per second
Srate	Source-to-destination packets per second
Max	Maximum duration of aggregated records
Attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
Category	Traffic category
Subcategory	Traffic subcategory

Table B.2 Description of BoT-IoT Dataset

B.3. NF-UQ-NIDS Dataset

Feature	Description
IPV4_SRC_ADDR	IPv4 source address
IPV4_DST_ADDR	IPv4 destination address
L4_SRC_PORT	IPv4 source port number
L4_DST_PORT	IPv4 destination port number
PROTOCOL	IP protocol identifier byte
L7_PROTO	Layer 7 protocol (numeric)
IN_BYTES	Incoming number of bytes
OUT_BYTES	Outgoing number of bytes
IN_PKTS	Incoming number of packets
OUT_PKTS	Outgoing number of packets
FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds
TCP_FLAGS	Cumulative of all TCP flags
CLIENT_TCP_FLAGS	Cumulative of all client TCP flags
SERVER_TCP_FLAGS	Cumulative of all server TCP flags
DURATION_IN	Client to Server stream duration (msec)
DURATION_OUT	Server to Client stream duration (msec)
MIN_TTL	Min flow TTL
MAX_TTL	Max flow TTL
LONGEST_FLOW_PKT	Longest packet (bytes) of the flow
SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow
MIN_IP_PKT_LEN	Length of the smallest flow IP packet observed
MAX_IP_PKT_LEN	Length of the largest flow IP packet observed
SRC_TO_DST_SECOND_BYTES	Source to destination bytes per second
DST_TO_SRC_SECOND_BYTES	Destination to source bytes per second
RETRANSMITTED_IN_BYTES	Number of retransmitted TCP flow bytes (src to dst)
RETRANSMITTED_IN_PKTS	Number of retransmitted TCP flow packets (src to dst)
RETRANSMITTED_OUT_BYTES	Number of retransmitted TCP flow bytes (dst to src)
RETRANSMITTED_OUT_PKTS	Number of retransmitted TCP flow packets (dst to src)
SRC_TO_DST_AVG_THROUGHPUT	Source to destination average throughput (bps)
DST_TO_SRC_AVG_THROUGHPUT	Destination to source average throughput (bps)
NUM_PKTS_UP_TO_128_BYTES	Packets whose IP size $\leq$ 128 bytes
NUM_PKTS_128_TO_256_BYTES	Packets whose IP size $>$ 128 and $\leq$ 256 bytes
NUM_PKTS_256_TO_512_BYTES	Packets whose IP size $>$ 256 and $\leq$ 512 bytes
NUM_PKTS_512_TO_1024_BYTES	Packets whose IP size $>$ 512 and $\leq$ 1024 bytes
NUM_PKTS_1024_TO_1514_BYTES	Packets whose IP size $>$ 1024 and $\leq$ 1514 bytes
TCP_WIN_MAX_IN	Max TCP window size (src to dst)
TCP_WIN_MAX_OUT	Max TCP window size (dst to src)
ICMP_TYPE	ICMP type $\times$ 256 + ICMP code
ICMP_IPV4_TYPE	ICMP type
DNS_QUERY_ID	DNS query transaction ID
DNS_QUERY_TYPE	DNS query type (e.g., 1 = A, 2 = NS)
DNS_TTL_ANSWER	TTL of the first A record (if any)
FTP_COMMAND_RET_CODE	FTP client command return code

Table B.3 Description of NF-UQ-NIDS-v2 Dataset

References

[1] Abdelaty, M., Scott-Hayward, S., Doriguzzi-Corin, R., and Siracusa, D. (2021). Gadot: Gan-based adversarial training for robust ddos attack detection. In *2021 IEEE Conference on Communications and Network Security (CNS)*, pages 119–127. IEEE.

[2] Abou Khamis, R. and Matrawy, A. (2020). Evaluation of adversarial training on different types of neural networks in deep learning-based idss. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE.

[3] AbouKhamis, R., Shafiq, O., and Matrawy, A. (2019). Investigating resistance of deep learning-based ids against adversaries using min-max optimization. *arXiv*.

[4] Abusnaina, A., Khormali, A., Nyang, D. H., Yuksel, M., and Mohaisen, A. (2019). Examining the Robustness of Learning-Based DDoS Detection in Software Defined Networks. *2019 IEEE Conference on Dependable and Secure Computing, DSC 2019 - Proceedings*.

[5] Agarwal, S. (2013). Data mining: Data mining concepts and techniques. In *2013 international conference on machine intelligence and research advancement*, pages 203–207. IEEE.

[6] Aiken, J. and Scott-Hayward, S. (2019). Investigating adversarial attacks against network intrusion detection systems in sdns. In *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7. IEEE.

[7] Akkiraju, R., Sinha, V., Xu, A., Mahmud, J., Gundecha, P., Liu, Z., Liu, X., and Schumacher, J. (2020). Characterizing machine learning processes: A maturity framework. In *Business Process Management: 18th International Conference, BPM 2020, Seville, Spain, September 13–18, 2020, Proceedings 18*, pages 17–31. Springer.

[8] Alahmed, S., Alasad, Q., Hammood, M. M., Yuan, J.-S., and Alawad, M. (2022). Mitigation of black-box attacks on intrusion detection systems-based ml. *Computers*, 11(7):115.

[9] Alatwi, H. A. and Aldweesh, A. (2021). Adversarial black-box attacks against network intrusion detection systems: A survey. In *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0034–0040. IEEE.

[10] Alatwi, H. A. and Morisset, C. (2021). Adversarial machine learning in network intrusion detection domain: A systematic review. *arXiv preprint arXiv:2112.03315*, pages 0034–0040.

[11] Alatwi, H. A. and Morisset, C. (2022). Threat modeling for machine learning-based network intrusion detection systems. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE.

[12] Alatwi, H. A. and Morisset, C. (2023). Realism versus performance for adversarial examples against dl-based nids. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, pages 1549–1557.

[13] Aldweesh, A., Derhab, A., and Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124.

[14] Alhajjar, E., Maxwell, P., and Bastian, N. (2021). Adversarial machine learning in network intrusion detection systems. *Expert Systems with Applications*, 186:115782.

[15] Alhajjar, E., Maxwell, P., and Bastian, N. D. (2020). Adversarial machine learning in network intrusion detection systems. *arXiv preprint arXiv:2004.11898*.

[16] Allen, B., Rachelle Loyear, C., et al. (2017). Enterprise security risk management: Concepts and applications.

[17] Almomani, I., Al-Kasasbeh, B., and Al-Akhras, M. (2016). Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks. *Journal of Sensors*, 2016.

[18] Alshahrani, E., Alghazzawi, D., Alotaibi, R., and Rabie, O. (2022). Adversarial attacks against supervised machine learning based network intrusion detection systems. *Plos one*, 17(10):e0275971.

[19] Anthi, E., Williams, L., Javed, A., and Burnap, P. (2021). Hardening machine learning denial of service (dos) defences against adversarial attacks in iot smart home networks. *computers & security*, page 102352.

[20] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., et al. (2017). Understanding the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110.

[21] Apruzzese, G., Andreolini, M., Ferretti, L., Marchetti, M., and Colajanni, M. (2021). Modeling realistic adversarial attacks against network intrusion detection systems. *arXiv preprint arXiv:2106.09380*.

[22] Apruzzese, G., Andreolini, M., Marchetti, M., Colacino, V. G., and Russo, G. (2020a). AppCon: Mitigating evasion attacks to ML cyber detectors. *Symmetry*, 12(4).

[23] Apruzzese, G., Andreolini, M., Marchetti, M., Venturi, A., and Colajanni, M. (2020b). Deep reinforcement adversarial learning against botnet evasion attacks. *IEEE Transactions on Network and Service Management*, 17(4):1975–1987.

[24] Apruzzese, G. and Colajanni, M. (2018a). Evading botnet detectors based on flows and random forest with adversarial samples. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE.

[25] Apruzzese, G. and Colajanni, M. (2018b). Evading botnet detectors based on flows and random forest with adversarial samples. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE.

[26] Apruzzese, G., Colajanni, M., Ferretti, L., and Marchetti, M. (2019). Addressing adversarial attacks against security systems based on machine learning. In *2019 11th International Conference on Cyber Conflict (CyCon)*, volume 900, pages 1–18. IEEE.

[27] Benzaïd, C., Boukhalfa, M., and Taleb, T. (2020). Robust self-protection against application-layer (d) dos attacks in sdn environment. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

[28] Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

[29] Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176.

[30] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

[31] Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

[32] Chaitou, H., Robert, T., Leneutre, J., and Pautet, L. (2021). Assessing adversarial training effect on idss and gans. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 543–550. IEEE.

[33] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

[34] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2021). A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45.

[35] Chauhan, R. and Heydari, S. S. (2020). Polymorphic adversarial ddos attack on ids using gan. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6. IEEE.

[36] Chen, J., Jordan, M. I., and Wainwright, M. J. (2020a). Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE.

[37] Chen, J., Wu, D., Zhao, Y., Sharma, N., Blumenstein, M., and Yu, S. (2020b). Fooling intrusion detection systems using adversarially autoencoder. *Digital Communications and Networks*.

[38] Chen, L., Wang, Z., Huo, R., and Huang, T. (2023). An adversarial dbn-lstm method for detecting and defending against ddos attacks in sdn environments. *Algorithms*, 16(4):197.

[39] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). 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*, pages 15–26.

[40] Cheng, Q., Zhou, S., Shen, Y., Kong, D., and Wu, C. (2021). Packet-level adversarial network traffic crafting using sequence generative adversarial networks. *arXiv preprint arXiv:2103.04794*.

[41] Chernikova, A. and Oprea, A. (2019). Fence: Feasible evasion attacks on neural networks in constrained environments. *arXiv preprint arXiv:1909.10480*.

[42] Clements, J., Yang, Y., Sharma, A., Hu, H., and Lao, Y. (2019). Rallying adversarial techniques against deep learning for network security. *arXiv preprint arXiv:1903.11688*.

[43] Clements, J., Yang, Y., Sharma, A. A., Hu, H., and Lao, Y. (2021). Rallying adversarial techniques against deep learning for network security. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 01–08. IEEE.

[44] Debicha, I., Bauwens, R., Debatty, T., Dricot, J.-M., Kenaza, T., and Mees, W. (2023). Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems. *Future Generation Computer Systems*, 138:185–197.

[45] Debicha, I., Debatty, T., Dricot, J.-M., and Mees, W. (2021). Adversarial training for deep learning-based intrusion detection systems. *arXiv preprint arXiv:2104.09852*.

[46] Debicha, I., Debatty, T., Dricot, J.-M., Mees, W., and Kenaza, T. (2022). Detect & reject for transferability of black-box adversarial attacks against network intrusion detection systems. In *Advances in Cyber Security: Third International Conference, ACeS 2021, Penang, Malaysia, August 24–25, 2021, Revised Selected Papers*, pages 329–339. Springer.

[47] Dhall, D., Kaur, R., and Juneja, M. (2020). Machine learning: a review of the algorithms and its applications. *Proceedings of ICRIC 2019: Recent innovations in computing*, pages 47–63.

[48] Duy, P. T., Khoa, N. H., Do Hoang, H., Pham, V.-H., et al. (2023). Investigating on the robustness of flow-based intrusion detection system against adversarial samples using generative adversarial networks. *Journal of Information Security and Applications*, 74:103472.

[49] Dyrmishi, S., Ghamizi, S., Simonetto, T., Traon, Y. L., and Cordy, M. (2022). On the empirical effectiveness of unrealistic adversarial hardening against realistic adversarial attacks. *arXiv preprint arXiv:2202.03277*.

[50] El Naqa, I. and Murphy, M. J. (2015). *What is machine learning?* Springer.

[51] Frazier, P. I. (2018). Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pages 255–278. Informs.

[52] Fu, X., Zhou, N., Jiao, L., Li, H., and Zhang, J. (2021). The robust deep learning–based schemes for intrusion detection in internet of things environments. *Annals of Telecommunications*, pages 1–13.

[53] Ganesan, A. and Sarac, K. (2021). Mitigating evasion attacks on machine learning based nids systems in sdn. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 268–272. IEEE.

[54] Garcia, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *computers & security*, 45:100–123.

[55] Ge, Z., Song, Z., Ding, S. X., and Huang, B. (2017). Data mining and analytics in the process industry: The role of machine learning. *Ieee Access*, 5:20590–20616.

[56] Gharaibeh, C. S. U. M. (2009). *(2009) DARPA 2009 Intrusion Detection Dataset*. [Online].

[57] Gómez, Á. L. P., Maimó, L. F., Celdrán, A. H., Clemente, F. J. G., and Cleary, F. (2021). Crafting adversarial samples for anomaly detectors in industrial control systems. *Procedia Computer Science*, 184:573–580.

[58] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[59] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[60] Grierson, S., Thomson, C., Papadopoulos, P., and Buchanan, B. (2021). Min-max training: Adversarially robust learning models for network intrusion detection systems. In *2021 14th International Conference on Security of Information and Networks (SIN)*, volume 1, pages 1–8. IEEE.

[61] Guo, S., Zhao, J., Li, X., Duan, J., Mu, D., and Jing, X. (2021). A black-box attack method against machine-learning-based anomaly network flow detection models. *Security and Communication Networks*, 2021.

[62] Han, D., Wang, Z., Zhong, Y., Chen, W., Yang, J., Lu, S., Shi, X., and Yin, X. (2020). Practical traffic-space adversarial attacks on learning-based nidss. *arXiv preprint arXiv:2005.07519*.

[63] Hashemi, M. J., Cusack, G., and Keller, E. (2019). Towards evaluation of nidss in adversarial setting. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, pages 14–21.

[64] Hashemi, M. J. and Keller, E. (2020). Enhancing robustness against adversarial examples in network intrusion detection systems. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 37–43. IEEE.

[65] Hassan, A. A., Hussein, M. S., AboMoustafa, A. S., and Elmowafy, S. H. (2022). Synthesis of adversarial ddos attacks using tabular generative adversarial networks. *arXiv preprint arXiv:2212.14109*.

[66] He, K., Kim, D. D., and Asghar, M. R. (2023). Adversarial machine learning for network intrusion detection systems: a comprehensive survey. *IEEE Communications Surveys & Tutorials*.

[67] He, K., Kim, D. D., Sun, J., Yoo, J. D., Lee, Y. H., and Kim, H. K. (2022). Liuer mihou: A practical framework for generating and evaluating grey-box adversarial attacks against nids. *arXiv preprint arXiv:2204.06113*.

[68] Hore, S., Ghadermazi, J., Paudel, D., Shah, A., Das, T. K., and Bastian, N. D. (2023). Deep packgen: A deep reinforcement learning framework for adversarial network packet generation. *arXiv preprint arXiv:2305.11039*.

[69] Huang, C. H., Lee, T. H., huang Chang, L., Lin, J. R., and Horng, G. (2019). Adversarial attacks on SDN-based deep learning IDS system. *Lecture Notes in Electrical Engineering*, 513:181–191.

[70] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. (2011). Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58.

[71] Ibitoye, O., Abou-Khamis, R., Matrawy, A., and Shafiq, M. O. (2019a). The threat of adversarial attacks on machine learning in network security–a survey. *arXiv preprint arXiv:1911.02621*.

[72] Ibitoye, O., Shafiq, O., and Matrawy, A. (2019b). Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks. *arXiv*.

[73] Jadidi, Z., Pal, S., Selvakkumar, A., Chang, C.-C., Beheshti, M., Jolfaei, A., et al. (2022). Security of machine learning-based anomaly detection in cyber physical systems. *arXiv preprint arXiv:2206.05678*.

[74] Janusz, A. and Kałuza (2019). Ieee bigdata 2019 cup: suspicious network event recognition. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5881–5887. IEEE.

[75] Jeong, J., Kwon, S., Hong, M.-P., Kwak, J., and Shon, T. (2019). Adversarial attack-based security vulnerability verification using deep learning library for multimedia video surveillance. *Multimedia Tools and Applications*, pages 1–15.

[76] Jiang, H., Lin, J., and Kang, H. (2022). Fgmd: A robust detector against adversarial attacks in the iot network. *Future Generation Computer Systems*, 132:194–210.

[77] Jmila, H. and Khedher, M. I. (2022). Adversarial machine learning for network intrusion detection: A comparative study. *Computer Networks*, page 109073.

[78] Khamaiseh, S. Y., Alsmadi, I., and Al-Alai, A. (2020). Deceiving machine learning-based saturation attack detection systems in sdn. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 44–50. IEEE.

[79] Khettaf, D. and Bouzar-Benlabiod, L. (2022). Defending the defender: Detecting adversarial examples for network intrusion detection systems. *arXiv preprint arXiv:2203.01467*.

[80] Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

[81] Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796.

[82] Kulikov, D. and Platonov, V. (2021). Adversarial attacks on intrusion detection systems using the lstm classifier. *Automatic Control and Computer Sciences*, 55(8):1080–1086.

[83] Kuppa, A., Grzonkowski, S., Asghar, M. R., and Le-Khac, N.-A. (2019). Black box attacks on deep anomaly detectors. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–10.

[84] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

[85] Kurakin, A., Goodfellow, I. J., and Bengio, S. (2018). Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC.

[86] Lin, Y.-D., Pratama, J.-H., Sudyana, D., Lai, Y.-C., Hwang, R.-H., Lin, P.-C., Lin, H.-Y., Lee, W.-B., and Chiang, C.-K. (2022). Elat: Ensemble learning with adversarial training in defending against evaded intrusions. *Journal of Information Security and Applications*, 71:103348.

[87] Lin, Z., Shi, Y., and Xue, Z. (2018). Idsgan: Generative adversarial networks for attack generation against intrusion detection. *arXiv preprint arXiv:1809.02077*.

[88] Liu, H. and Lang, B. (2019). Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396.

[89] Maarouf, R., Sattar, D., and Matrawy, A. (2021). Evaluating resilience of encrypted traffic classification against adversarial evasion attacks. *arXiv preprint arXiv:2105.14564*.

[90] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

[91] Mahesh, B. (2020). Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9(1):381–386.

[92] Martins, N., Cruz, J. M., Cruz, T., and Abreu, P. H. (2019). Analyzing the footprint of classifiers in adversarial denial of service contexts. In *EPIA Conference on Artificial Intelligence*, pages 256–267. Springer.

[93] Martins, N., Cruz, J. M., Cruz, T., and Abreu, P. H. (2020). Adversarial machine learning applied to intrusion and malware scenarios: a systematic review. *IEEE Access*, 8:35403–35419.

[94] Mathews, J., Chatterjee, P., Banik, S., and Nance, C. (2022). A deep learning approach to create dns amplification attacks. *arXiv preprint arXiv:2206.14346*.

[95] McCarthy, A., Andriotis, P., Ghadafi, E., and Legg, P. (2021). Feature vulnerability and robustness assessment against adversarial machine learning attacks. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8. IEEE.

[96] Merzouk, M. A., Cuppens, F., Boulahia-Cuppens, N., and Yaich, R. (2020). A deeper analysis of adversarial examples in intrusion detection. In *International Conference on Risks and Security of Internet and Systems*, pages 67–84. Springer.

[97] Merzouk, M. A., Delas, J., Neal, C., Cuppens, F., Boulahia-Cuppens, N., and Yaich, R. (2022). Evading deep reinforcement learning-based network intrusion detection with adversarial attacks. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–6.

[98] Microsoft (2023). Threats - microsoft threat modeling tool - azure. *Threats - Microsoft Threat Modeling Tool - Azure | Microsoft Docs*.

[99] Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. (2018). Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*.

[100] Mishra, P., Varadharajan, V., Tupakula, U., and Pilli, E. S. (2018). A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE Communications Surveys & Tutorials*, 21(1):686–728.

[101] Mogg, R., Enoch, S. Y., and Kim, D. S. (2021). A framework for generating evasion attacks for machine learning based network intrusion detection systems. In *International Conference on Information Security Applications*, pages 51–63. Springer.

[102] Mohammadian, H., Ghorbani, A. A., and Lashkari, A. H. (2023). A gradient-based approach for adversarial attack on deep learning-based network intrusion detection systems. *Applied Soft Computing*, page 110173.

[103] Mohanty, H., Roudsari, A. H., and Lashkari, A. H. (2022). Robust stacking ensemble model for darknet traffic classification under adversarial settings. *Computers & Security*, page 102830.

[104] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

[105] Moustafa, N., Hu, J., and Slay, J. (2019). A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55.

[106] Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE.

[107] Myagmar, S., Lee, A. J., and Yurcik, W. (2005). Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)*, volume 2005, pages 1–8. Citeseer.

[108] Nagaraju, V., Fiondella, L., and Wandji, T. (2017). A survey of fault and attack tree modeling and analysis for cyber risk management. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6. IEEE.

[109] Nguyen, X.-H., Nguyen, X.-D., and Le, K.-H. (2022). Preventing adversarial attacks against deep learning-based intrusion detection system. In *International Conference on Information Security Practice and Experience*, pages 382–396. Springer.

[110] Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., et al. (2018). Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*.

[111] Niyaz, Q., Sun, W., and Javaid, A. Y. (2016). A deep learning based ddos detection system in software-defined networking (sdn). *arXiv preprint arXiv:1611.07400*.

[112] Novaes, M. P., Carvalho, L. F., Lloret, J., and Proença Jr, M. L. (2021). Adversarial deep learning approach detection and defense against ddos attacks in sdn environments. *Future Generation Computer Systems*, 125:156–167.

[113] Nowroozi, E., Mohammadi, M., Golmohammadi, P., Mekdad, Y., Conti, M., and Uluagac, S. (2022a). Resisting deep learning models against adversarial attack transferability via feature randomization. *arXiv preprint arXiv:2209.04930*.

[114] Nowroozi, E., Mohammadi, M., Savas, E., Conti, M., and Mekdad, Y. (2022b). Spritz-1.5 c: Employing deep ensemble learning for improving the security of computer networks against adversarial attacks. *arXiv preprint arXiv:2209.12195*.

[115] Nugraha, B., Kulkarni, N., and Gopikrishnan, A. (2021). Detecting adversarial ddos attacks in software-defined networking using deep learning techniques and adversarial training. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 448–454. IEEE.

[116] of New Brunswick, U. (2023). Nsl-kdd dataset. *https://www.unb.ca/cic/datasets/nsl.html*.

[117] Pacheco, Y. and Sun, W. (2021). Adversarial machine learning: A comparative study on contemporary intrusion detection datasets. In *ICISSP*, pages 160–171.

[118] Papernot, N., McDaniel, P., and Goodfellow, I. (2016a). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

[119] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

[120] Pawlicki, M., Choraś, M., and Kozik, R. (2020). Defending network intrusion detection systems against adversarial evasion attacks. *Future Generation Computer Systems*, 110:148–154.

[121] Peng, X., Huang, W., and Shi, Z. (2019a). Adversarial attack against dos intrusion detection: An improved boundary-based method. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1288–1295. IEEE.

[122] Peng, Y., Fu, G., Luo, Y., Hu, J., Li, B., and Yan, Q. (2020). Detecting adversarial examples for network intrusion detection system with gan. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, pages 6–10. IEEE.

[123] Peng, Y., Su, J., Shi, X., and Zhao, B. (2019b). Evaluating deep learning based network intrusion detection system in adversarial environment. *ICEIEC 2019 - Proceedings of 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication*, pages 61–66.

[124] Peng, Y., Su, J., Shi, X., and Zhao, B. (2019c). Evaluating deep learning based network intrusion detection system in adversarial environment. In *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pages 61–66. IEEE.

[125] Piplai, A., Chukkapalli, S. S. L., and Joshi, A. (2020). Nattack! adversarial attacks to bypass a gan based classifier trained to detect network intrusion. In *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 49–54. IEEE.

[126] Pitropakis, N., Panaousis, E., Giannetsos, T., Anastasiadis, E., and Loukas, G. (2019). A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199.

[127] Pujari, M., Pacheco, Y., Cherukuri, B., and Sun, W. (2022). A comparative study on the impact of adversarial machine learning attacks on contemporary intrusion detection datasets. *SN Computer Science*, 3(5):1–12.

[128] Qiu, H., Dong, T., Zhang, T., Lu, J., Memmi, G., and Qiu, M. (2020). Adversarial attacks against network intrusion detection in iot systems. *IEEE Internet of Things Journal*.

[129] Qiu, S., Liu, Q., Zhou, S., and Wu, C. (2019). Review of artificial intelligence adversarial attack and defense technologies. *Applied Sciences*, 9(5):909.

[130] Qureshi, A. U. H., Larijani, H., Mtetwa, N., Yousefi, M., and Javed, A. (2020). An Adversarial Attack Detection Paradigm with Swarm Optimization. *Proceedings of the International Joint Conference on Neural Networks*.

[131] Randhawa, R. H., Aslam, N., Alauthman, M., Khalid, M., and Rafiq, H. (2022). Deep reinforcement learning based evasion generative adversarial network for botnet detection. *arXiv preprint arXiv:2210.02840*.

[132] Rashid, M. M., Kamruzzaman, J., Hassan, M. M., Imam, T., Wibowo, S., Gordon, S., and Fortino, G. (2022). Adversarial training for deep learning-based cyberattack detection in iot-based smart city applications. *Computers & Security*, page 102783.

[133] Rigaki, M. (2017). Adversarial deep learning against intrusion detection classifiers. *https://www.diva-portal.org/smash/get/diva2:1116037/FULLTEXT01.pdf*.

[134] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167.

[135] Rosenberg, I., Shabtai, A., Elovici, Y., and Rokach, L. (2021). Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36.

[136] Sadeghzadeh, A. M., Shiravi, S., and Jalili, R. (2021). Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification. *IEEE Transactions on Network and Service Management*, 18(2):1962–1976.

[137] Sarhan, M., Layeghy, S., Moustafa, N., and Portmann, M. (2021). Netflow datasets for machine learning-based network intrusion detection systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10*, pages 117–135. Springer.

[138] Sarhan, M., Layeghy, S., and Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile networks and applications*, pages 1–14.

[139] Sauka, K., Shin, G.-Y., Kim, D.-W., and Han, M.-M. (2022). Adversarial robust and explainable network intrusion detection systems based on deep learning. *Applied Sciences*, 12(13):6451.

[140] Schneider, M., Aspinall, D., and Bastian, N. D. (2021). Evaluating model robustness to adversarial samples in network intrusion detection. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 3343–3352. IEEE.

[141] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp*, pages 108–116.

[142] Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. (2019). Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*, pages 1–8. IEEE.

[143] Sharon, Y., Berend, D., Liu, Y., Shabtai, A., and Elovici, Y. (2021). Tantra: Timing-based adversarial network traffic reshaping attack. *arXiv preprint arXiv:2103.06297*.

[144] Shieh, C.-S., Nguyen, T.-T., Lin, W.-W., Huang, Y.-L., Horng, M.-F., Lee, T.-F., and Miu, D. (2022). Detection of adversarial ddos attacks using generative adversarial networks with dual discriminators. *Symmetry*, 14(1):66.

[145] Shostack, A. (2014). *Threat modeling: Designing for security*. John Wiley & Sons.

[146] Shu, D., Leslie, N. O., Kamhoua, C. A., and Tucker, C. S. (2020). Generative adversarial attacks against intrusion detection systems using active learning. In *Proceedings of the 2nd ACM Workshop on Wireless Security and Machine Learning*, pages 1–6.

[147] Shu, R., Xia, T., Williams, L., and Menzies, T. (2022). Omni: automated ensemble with unexpected models against adversarial evasion attack. *Empirical Software Engineering*, 27(1):1–32.

[148] Simonetto, T., Dyrmishi, S., Ghamizi, S., Cordy, M., and Traon, Y. L. (2021). A unified framework for adversarial attack and defense in constrained feature space. *arXiv preprint arXiv:2112.01156*.

[149] Sriram, S., Simran, K., Vinayakumar, R., Akarsh, S., and Soman, K. P. (2020). Towards Evaluating the Robustness of Deep Intrusion Detection Models in Adversarial Environment. *Communications in Computer and Information Science*, 1208 CCIS(January):111–120.

[150] Sun, H., Peng, C., Sang, Y., Li, S., Zhang, Y., and Zhu, Y. (2023). Evading encrypted traffic classifiers by transferable adversarial traffic. In *Collaborative Computing: Networking, Applications and Worksharing: 18th EAI International Conference, CollaborateCom 2022, Hangzhou, China, October 15-16, 2022, Proceedings, Part II*, pages 153–173. Springer.

[151] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

[152] Talty, K., Stockdale, J., and Bastian, N. D. (2021). A sensitivity analysis of poisoning and evasion attacks in network intrusion detection system machine learning models. In *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*, pages 1011–1016. IEEE.

[153] Tan, S., Zhong, X., Tian, Z., and Dong, Q. (2022). Sneaking through security: Mutating live network traffic to evade learning based nids. *IEEE Transactions on Network and Service Management*.

[154] Tcydenova, E., Kim, T. W., Lee, C., and Park, J. H. (2021). Detection of adversarial attacks in ai-based intrusion detection systems using explainable ai. *HUMAN-CENTRIC COMPUTING AND INFORMATION SCIENCES*, 11.

[155] Teuffenbach, M., Piatkowska, E., and Smith, P. (2020). *Subverting Network Intrusion Detection: Crafting Adversarial Examples Accounting for Domain-Specific Constraints*, volume 12279 LNCS. Springer International Publishing.

[156] University, K. (2023). Traffic data from kyoto university's honeypots. *http://www.takakura.com/Kyoto_data/*.

[157] University of California, Irvine (2023). Kdd cup 1999 data. *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*.

[158] Usama, M., Asim, M., Latif, S., Qadir, J., et al. (2019a). Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pages 78–83. IEEE.

[159] Usama, M., Qadir, J., Al-Fuqaha, A., and Hamdi, M. (2019b). The adversarial machine learning conundrum: Can the insecurity of ml become the achilles' heel of cognitive networks? *IEEE Network*, 34(1):196–203.

[160] Usama, M., Qayyum, A., Qadir, J., and Al-Fuqaha, A. (2019c). Black-box adversarial machine learning attack on network traffic classification. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 84–89. IEEE.

[161] Venturi, A., Zanasi, C., Marchetti, M., and Colajanni, M. (2022). Robustness evaluation of network intrusion detection systems based on sequential machine learning. In *2022 IEEE 21st International Symposium on Network Computing and Applications (NCA)*, volume 21, pages 235–242. IEEE.

[162] Vitorino, J., Oliveira, N., and Praça, I. (2022). Adaptative perturbation patterns: Realistic adversarial learning for robust intrusion detection. *Future Internet*, 14(4):108.

[163] Vitorino, J., Praça, I., and Maia, E. (2023). Towards adversarial realism and robust learning for iot intrusion detection and classification. *Annals of Telecommunications*, pages 1–12.

[164] Wang, J., Pan, J., AlQerm, I., and Liu, Y. (2021a). Def-ids: An ensemble defense mechanism against adversarial attacks for deep learning-based network intrusion detection. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9. IEEE.

[165] Wang, J., Qixu, L., Di, W., Dong, Y., and Cui, X. (2021b). Crafting adversarial example to bypass flow-&ml-based botnet detector via rl. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 193–204.

[166] Wang, M., Yang, N., Gunasinghe, D. H., and Weng, N. (2023). On the robustness of ml-based network intrusion detection systems: An adversarial and distribution shift perspective. *Computers*, 12(10):209.

[167] Wang, N., Chen, Y., Xiao, Y., Hu, Y., Lou, W., and Hou, T. (2022). Manda: On adversarial example detection for network intrusion detection system. *IEEE Transactions on Dependable and Secure Computing*.

[168] Wang, Y., Wang, Y., Tong, E., Niu, W., and Liu, J. (2020). A c-ifgsm based adversarial approach for deep learning based intrusion detection. In *International Conference on Verification and Evaluation of Computer and Communication Systems*, pages 207–221. Springer.

[169] Wang, Z. (2018). Deep Learning-Based Intrusion Detection with Adversaries. *IEEE Access*, 6:38367–38384.

[170] Warzyński, A. and Kołaczek, G. (2018). Intrusion detection systems vulnerability on adversarial examples. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pages 1–4. IEEE.

[171] Wu, D., Fang, B., Wang, J., Liu, Q., and Cui, X. (2019). Evading machine learning botnet detection models via deep reinforcement learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

[172] Xiong, W. D., Luo, K. L., and Li, R. (2023). Aidtf: Adversarial training framework for network intrusion detection. *Computers & Security*, page 103141.

[173] Yan, Q., Wang, M., Huang, W., Luo, X., and Yu, F. R. (2019). Automatically synthesizing DoS attack traces using generative adversarial networks. *International Journal of Machine Learning and Cybernetics*, 10(12):3387–3396.

[174] Yang, C., Zhou, L., Wen, H., and Wu, Y. (2020). U-ASG: A universal method to perform adversarial attack on autoencoder based network anomaly detection systems. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2020*, pages 68–73.

[175] Yang, K., Liu, J., Zhang, C., and Fang, Y. (2019). Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems. *Proceedings - IEEE Military Communications Conference MILCOM*, 2019-October:559–564.

[176] Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.

[177] Zakariyya, I., Kalutarage, H., and Al-Kadri, M. O. (2022). Robust, effective and resource efficient deep neural network for intrusion detection in iot networks. In *Proceedings of the 8th ACM on Cyber-Physical System Security Workshop*, pages 41–51.

[178] Zhang, C., Costa-Pérez, X., and Patras, P. (2020a). Tiki-taka: Attacking and defending deep learning-based intrusion detection systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 27–39.

[179] Zhang, C., Patras, P., and Haddadi, H. (2019). Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287.

[180] Zhang, R., Luo, S., Pan, L., Hao, J., and Zhang, J. (2022). Generating adversarial examples via enhancing latent spatial features of benign traffic and preserving malicious functions. *Neurocomputing*, 490:413–430.

[181] Zhang, S., Xie, X., and Xu, Y. (2020b). A Brute-Force Black-Box Method to Attack Machine Learning-Based Systems in Cybersecurity. *IEEE Access*, 8:128250–128263.

[182] Zhong, Y., Zhu, Y., Wang, Z., Yin, X., Shi, X., and Li, K. (2020). An adversarial learning model for intrusion detection in real complex network environments. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 794–806. Springer.

[183] Zolbayar, B.-E., Sheatsley, R., McDaniel, P., Weisman, M. J., Zhu, S., Zhu, S., and Krishnamurthy, S. (2022). Generating practical adversarial network traffic flows using nidsgan. *arXiv preprint arXiv:2203.06694*.