LIGHTWEIGHT SYBIL ATTACK DETECTION USING MACHINE LEARNING WITH GAME THEORY FEATURES

by

MANIDEEP SANDIREDDY (22BCE1434)

ABSTRACT

The creation of strong defenses against advanced, evolving cyber threats is a key issue in machine learning-driven security. The traditional lightweight intrusion detection systems (IDS) are also trained on a set of static and clean data only and they are therefore brittle by nature and can be easily overcome by malicious parties. This vulnerability enables an attacker to construct adversarial inputs (minimally perturbed, evasive inputs) that can result in the detector misclassifying malicious traffic as benign and leads to a catastrophic failure in performance. The present project attempts to close this gap through the formalization of the Sybil attack detection problem as a two-player Min-Max Adversarial Game. The game-theoretic principle is applied via Adversarial Training, where the model preemptively risks a perfect adversary and counterattacks the adversary, as the learning model. We built a known vulnerable experimental design based on Logistic Regression, which was vulnerable to gradient based attacks, and hardened it to the state-of-the-art evasion strategies, namely the single step Fast Gradient Sign Method (FGSM) and the more formidable, iterative Projected Gradient Descent (PGD) approaches. Successful results were observed once empirical validation on the pre-processed NSL-KDD data was done. The Standard Model suffered a meltdown when put to test by adversaries, and F1-Scores plunged down to 0.4446 on attacked data, compared to 0.7446 on clean data. On the other hand, PGD Adversarial Model was able to resist such a collapse, with an F1-Score of 0.6827 to the same attack, and it was more resilient. These findings are a measurement of how well the game-theoretic strategy works and provide a clear example of the Accuracy-Robustness Trade-Off that characterizes contemporary secure machine learning systems. This work, by fundamentally altering the posture of the defender, through a radical shift in the defender between a reactive to a proactive posture, establishes a methodology of constructing lightweight network security systems that are robust in nature.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Adv.	Adversarial
BCE	Binary Cross-Entropy
DoS	Denial-of-Service
FGSM	Fast Gradient Sign Method
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
KDD	Knowledge Discovery and Data Mining (The dataset family)
LR	Logistic Regression
ML	Machine Learning
NIDS	Network Intrusion Detection System
PGD	Projected Gradient Descent
SMOTE	Synthetic Minority Over-sampling Technique
TN	True Negative
TP	True Positive

Chapter 1
Introduction

1.1 INTRODUCTION

The balance between defenders and advanced enemy determined by the security environment in the contemporary, interconnected systems, whether they are large corporate network or decentralized peer-to-peer structure is characterized by endless asymmetric warfare. Here, a vanguard of defense against the sophisticated and zero-day threats has turned out to be the Network Intrusion Detection Systems (NIDS) that are driven by the use of Machine Learning (ML). Nevertheless, the transition to ML-based security has unintentionally created another, more severe vulnerability in the form of the vulnerability of these models to adversarial examples.

An adversarial example is a datum (an example of network flow record) to which a minute, usually imperceptible, perturbation has been applied. Such intentional manipulation leads to a misclassification of the trained model, which enables the detection of malicious traffic to be avoided. The main objective of this project is to come up with strong protection measures against this threat with specific emphasis on the network-level implications of a Sybil attack.

The Sybil attack in which one attacker legitimizes many bogus identities on a network acts as a base to more extensive harmful targeted campaigns. Although the attack may be on identity and reputation systems, its practical result can be easily identified network patterns, e.g. arranging Denial-of-Service (DoS) traffic (e.g. Neptune) or extensive Probe or reconnaissance (e.g. Ipsweep). Our training of an IDS to recognize these observable patterns on the commonly used NSL-KDD data produces a system that works to recognize the Sybil Effect- the malicious network trace left by the fake identities. The pending issue here is to make this detector resistant to an intelligent Sybil attacker who tries to avoid detection by carefully manipulating the features of its network traffic to attain a state of detection avoidance.

1.2 BACKGROUND OF THE PROBLEM

The susceptibility of the traditional ML-based NIDS is due to the lack of fit between the operating environment and the optimization objective. A typical model is fitted to the clean, unperturbed data distribution so as to achieve a minimum expected loss. It is also optimized in terms of maximum fidelity to the training data, which means that the border of the decision is usually very sensitive and fragile in the direct vicinity of the data points.

This weakness is a considerable disadvantage when one has to deal with a rational, strategic attacker. The attacker takes advantage of such sensitivity through the gradient-based information to identify the steepest ascent direction of the loss function of the model. A tiny step (epsilon) in this direction causes the attacker to generate an input that looks exactly like the original input, but is beyond the decision boundary, causing the attacker to make a misclassification.

In lightweight systems, this is usually worsened by the architectural decision. As an example, in this project, the use of Logistic Regression was selected on purpose since it is its excessive linearity that is known to make the model highly vulnerable to gradient-based adversarial attacks. This is a fundamental weakness combined with the optimization of the Binary Cross-Entropy (BCE) loss function on clean data, which actively introduces the circumstances of adversarial vulnerability. A naive NIDS is inherently reactive, and does not have any internal mechanism to predict or resist such adaptive attacks, and thus a collapse in detection performance is demonstrated. The rationale behind using light weight models in this projects was due to their flexibility to the environments where the resource available is limited and lightweight to allow them to be deployed in small devices such as IOT sensors and other devices with resource constraints.

1.3 NEED FOR THE PROPOSED SYSTEM

The realized futility of reactive security makes the creation of a principled, proactive approach to defense necessary. A high accuracy on clean data is worth nothing when the detection rate of the model drops to almost zero in case of an intelligent attack.

The proposed system is needed to fill this severe security gap by satisfying need of:

- A Formal Game-Theoretic Framework: The necessity to go beyond the heuristic-based defenses and construct a formal mathematical model of the Sybil attacker versus defender strategic game as a formal two-player, non-zero-sum game. This principle is converted straight into the robust optimization principle.
- Adversarial Resilience: The most important requirement is a hardened detector against the optimum move of the attacker. It involves the use of the entire min-max objective function to compel the model achieve a robust decision boundary.
- Resistance to the State-of-the-Art Threats: The awareness that a training against weak single-step adversarial examples such as FGSM can give the illusion of robustness via the so-called obfuscated gradients. The utmost necessity is to realize a really, meaningful robustness by fortifying the model against stronger, foes such as (PGD).
- Reduction of the Accuracy-Robustness Trade-Off: The pragmatic requirement of having a system that can be deployed and at the same time be able to perform well on both clean traffic and benign traffic as well as the adversarial traffic. This will necessitate a new Hybrid Ensemble architecture that is able to smartly integrate the merits of the so-called accuracy specialists and the so-called robustness specialists.

1.4 PROBLEM STATEMENT

The major constraint that characterizes this study as follows: "Formally stated, NISA Foundational NISA NSL-KDD Traditional, lightweight NID: The devastating fact is that under subtle, gradient-based evasion attack settings, the main and most widely used models are traditional NISA, represented as simple Logistic Regression models, which are trained on the NSL-KDD dataset. An Intense, Lean Defense demands a structured

implementation of Game Theory, specifically, the Min-Max Adversarial Training framework, in order to actively securitise the system against state-of-the-art iterative attacks such as PGD and thus to define and quantify the resultant basic Accuracy-Robustness Trade-Off. It is not just to identify attacks, but to identify attacks in the worst-case scenario a rational and intelligent enemy can cause.

1.5 AIM & OBJECTIVES

The main aim of this project is to design and test a lightweight intrusion-detection system that can still perform well even when attackers try to fool it. Instead of treating the attacker as a simple threat, the work views the problem more like a strategic game between the defender (the model) and the attacker. My idea was to bring this game-like thinking into machine learning training so that the system learns how to handle deceptive Sybil-style attacks before they happen in real life.

To achieve this goal, we set a few clear objectives for the project:

- **To apply Min-Max based training:** The first step was to implement adversarial learning using the FGSM method. This helped simulate how an attacker makes small but smart changes to input data. The model is then trained to handle these changes and reduce its mistakes.
- **To strengthen the attack simulation:** I then extended this idea using the PGD algorithm. PGD is more advanced and repeatedly applies small changes, making it a tougher adversary. The plan was to show that the model trained under PGD conditions becomes much harder to break.
- **To build a combined model:** Instead of depending on just one trained model, I created a hybrid decision system. This final model combines outputs from three versions — the normal model, the FGSM-trained one, and the PGD-trained one. Each of them contributes a weighted vote, with more weight given to the models trained to handle attacks. This helps keep a balance between accuracy on normal data and strong resistance against adversarial inputs.
- **To measure the trade-off properly:** Finally, I compared the performance of all the models under different test conditions (clean data, FGSM attack, and PGD attack). I used standard evaluation metrics like accuracy, precision, recall, and F1-

score to see how much robustness is gained and what accuracy cost comes with it.

- Overall, the intention was not just to build a working detector, but to practically understand how security-focused training affects performance and where the balance lies between accuracy and robustness.

1.6 SCOPE OF THE PROJECT

The scope of this project explains the limits and conditions under which the research was carried out. Keeping a clear boundary helped me stay focused and made sure the results could be reproduced in a proper and consistent manner.

- **Cyber-attack-setting:**

  For evaluation, I assumed a black-box attack environment. This means the attacker does not know the defense model's parameters but can still try to fool it. To simulate this realistically, I generated adversarial samples using the gradients of a normally trained baseline model. This represents a practical scenario where an attacker targets a common lightweight IDS without knowing the underlying defense strategies.

- **Dataset and preprocessing:**

  The NSL-KDD dataset (both KDDTrain+ and KDDTest+) was used, and the classification task was kept binary — normal vs attack. I followed a strict preprocessing pipeline using Min-Max scaling to keep all features in the same range, which is important for adversarial experiments. SMOTE was applied only on the training data to balance the classes and avoid data leakage into the test set.

- **Model architecture:**

  I intentionally selected a simple and lightweight Logistic Regression model. Even though it is easier to attack because of its linear nature, this helped me clearly observe the effect of adversarial training. The goal was not just to build a powerful model, but to show whether the training approach itself can improve robustness.

- **Evaluation approach:**

  To judge model performance fairly, I used multiple metrics such as accuracy, precision, recall, and F1-score. Accuracy alone can be misleading in imbalanced

datasets like NSL-KDD, so using a mix of metrics gave a more reliable picture of how well the model performs under normal and adversarial conditions.

1.7 SIGNIFICANCE & REAL-WORLD IMPACT

The importance of the project lies in the innovation of the methodology and its usefulness in practice that provide the improvement to the modern state of research connected with the NIDS:

• **Methodological Rigor:** The article shows a rigorous presentation of hardening a detector against a state-of-the-art PGD attack. This goes beyond the more basic FGSM defenses that are frequently mentioned in introductory adversarial literature and is a large step towards an actual trusted and resilient defense.

• **Architectural Novelty:** A successful Hybrid (Avg.) Model design is a principled solution to the long-running Accuracy-Robustness trade-off. Instead of having a new model that is trained by itself, this model is a defensive weighted ensemble of three special models, Standard, FGSM-Robust, and PGD-Robust. The system balances the performance and improves both benign and adversarial performance by strategically averaging their prediction probabilities with preset robustness-oriented weights.

• **Measurability of Security Assurance:** The research makes abstract security claims in a 4 3 comparison that converts abstract security claims into measurable, empirical findings, and offers auditor-verifiable guarantees of the resilience of a given model to a given threat model.

Future Resilience: The verified game-theoretic methodology creates a blueprint of building resilient, lightweight ML security elements to be used in any resource-limited or differentiated infrastructure where evasion attacks have become a predominant attack.

1.8 METHODOLOGY OVERVIEW

The methodology used in this project is designed to be straightforward and scientific, with each step building upon the previous one.

1. **Data Preparation:**

   The NSL-KDD dataset was carefully pre-processed and scaled. The training set was also balanced using the SMOTE technique to prevent bias and ensure that the

results were reliable and scientifically valid.

2. **Model Training (The Specialists):**

   I trained four different models to cover a range of defense strategies. These included the basic Standard Model, a version hardened against attacks using the FGSM method, and the PGD Adversarial Model, which is specifically designed to be more robust against advanced attacks.

3. **Hybrid Development:**

   The Hybrid (Avg.) Model is where things get interesting. Instead of building an entirely new meta-learner, I combined the predictions of three trained models: the Standard, FGSM-Robust, and PGD-Robust models. The model assigns different weights to each (50% for PGD-Robust, 30% for FGSM-Robust, and 20% for Standard), which helps balance out the trade-off between accuracy and robustness. This approach helps the system perform well in both normal and adversarial scenarios.

4. **Evaluation Framework:**

   For testing, I generated three distinct environments from the clean test data: a regular (clean) set, one with FGSM adversarial samples, and another with PGD adversarial samples. These adversarial sets were created using the gradients from the Standard Model, simulating a real-world attack scenario.

5. **Comparative Analysis:**

   Finally, I used a 4 * 3 evaluation matrix to compare the models. This matrix allowed me to analyze the models in two ways: horizontally (to see how each model performs under increasing attack strength) and vertically (to compare how different models react to the same adversarial conditions). The results provided clear insights into which model performed best under various scenarios.

Chapter 2

Background Study

2.1 INTRODUCTION TO SYBIL ATTACKS AND GAME THEORY IN SECURITY

Sybil attack is a fundamental attack on security analysis of both distributed and peer-to-peer (P2P) systems, in which trust and consensus are strongly dependent on distinct identity. The attacker is able to impersonate the identity/reputation system of a network by impersonating a number of fake identities (Sybils) illegitimately as different users. The basic idea of such an attack is to achieve an unequal influential amount and flood legitimate nodes, disrupting system integrity. There are many opportunities to use this control to achieve catastrophic results, such as attack on blockchain protocols 51 percent, or routing, or censorship in social networks. The very characteristics of this attack, in which a rational opponent will do everything to maximize his benefits, at the same time reducing the expenditure on maintenance and recognition to the minimum, perfectly fit the concept of the Game Theory. In this view the game between the Sybil adversary and the defense mechanism of the network can be thought of as a two-player game that is not a zero-sum game. It does not just identify the fixed patterns of attack but models mathematically the strategic conflict: Strategies of the Attacker: The greatest possible evasion probability with the smallest possible perturbation. The Strategy of the Defender: The Strategist is trying to achieve as large as possible the detection probability, in the entire band of possible evasions. This game-theoretic methodology offers the formal basis that would be needed to shift, not just to the creation of passive pattern-matching detectors, but to the development of security mechanisms that are indeed robust and resilient to adversarial activity.

2.2 THE SYBIL ATTACK AS AN ADVERSARIAL GAME: THE MIN-MAX FORMULATION

The optimization problem is a strict maximization issue that is converted in to the strategic conflict between the attacker and the machine learning model called Min-Max

Adversarial Framework. Adversarial training is theoretically grounded on this framework. It states that the defender (the optimization algorithm) is required to select model parameters that reduce the worst-case scenario that could be caused by the attacker (the calculation of the gradient). It is a turn taking conceptual game: The Attacker Move is conceptually the inner maximization problem. Given a state on the model of the defender, the attacker carefully determines the most effective perturbation (a small modification to the features of the network flow) that despite being subtle and spending a given budget, can lead to the largest possible misclassification error. This is a manipulative move that is being undertaken to avoid being noticed.

The outer minimization problem is the Defender Countermove. The defender modifies its classification range so as to reduce the loss caused by the optimum evasive action of the attacker. Through constant training on these artificially produced worst-case examples, the model has no option but to generalize its defense, creating a more natural, more robust decision boundary that is less prone to small inputs. It is this whole system that the training process is no longer a mere memorization game but a strategic anticipation game.

2.3 DATASET SELECTION: NSL-KDD AND THE SYBIL EFFECT

The choice and the rationalization of the empirical environment is a vital part of the base of this project. The NSL-KDD dataset has been selected as the standard reference in network intrusion detection as it is a major improvement of the older KDD'99 dataset because it eliminated redundant records which biased models unfairly to usual attacks.

Though NSL-KDD has not an explicit label of Sybil, its use can be highly vindicated by the Sybil Effect argument. A Sybil attack is effectively an enabler; the fabrication of falsified identities is taken up to carry out other malicious acts, including saturing resources with a DoS attack or mass surveillance with a Probe attack. The labels available in the NSL-KDD data (such as neptune on DoS or ipswep on Probe) are the observable physical effects of a Sybil infrastructure. Hence, training a model to identify these trends of malicious traffic is equivalent to creating a very useful Sybil Effect Detector. The dataset includes 41 features that describe network connection records, such as basic ones (type of protocol, service, sent bytes), content features

(whether or not achieved a login status, whether or not accessed files), and time- and host-based traffic statistics. Examining the data properly, such as Min-Max scaling of the features so that they become equal contributors to the calculation of the gradient and an application of SMOTE to the training set only, thus addressing the issue of the inherent class-imbalance, are critical to the scientific validity and reproducibility of the experimental findings.

2.4 EXISTING NIDS FRAMEWORKS, VULNERABILITIES, AND ARCHITECTURAL CHOICES

NIDS used in present adversarial settings are mostly ineffective because of the reactive nature. Their weakness is a direct result of their training methodologies which emphasize standard accuracy rather than strength. In this project, Logistic Regression was purposefully selected as the foundation architecture to its experimental format, which turned a strength into a significant limitation of the methodology.

The Vulnerability of the Naive Defender The traditional training results in models that are easily compromised. In the case of a simple linear model such as Logistic Regression, the optimization procedure comes up with a narrow decision boundary that can be easily pushed by a gradient based attack. The excessive linearity of the Logistic Regression, which is chosen, is a premeditated worst-case scenario of the defender. This will make certain that the performance decline as experienced in the Standard Model under attack is bright and definite, which will be a clear, good benchmark against which the robustness gains of the adversarial training can be gauged.

Limitations of Naive Optimization The optimization process of the Standard Model, which is usually to minimize the loss of Binary Cross-Entropy (BCE) on clean data, does not condition the model on out-of-distribution inputs. It produces the exact attack surface, which adversarial examples are meant to target, which is to locate an extremely sensitive point at the edge of the decision boundary in order to reverse the prediction.

2.5 ADVANCING THE ADVERSARY: FGSM AND PGD ATTACKS FOR ROBUSTNESS

To implement the Min-Max game, the system needs to be tough enough to handle stronger adversaries. This means understanding the different types of attacks that could trick the model.

- **Fast Gradient Sign Method (FGSM):**
  FGSM is a basic, fast attack method. It works by taking a single large step in the direction that increases the model's mistake the most. While this is good for understanding vulnerabilities, training the model only on FGSM can be misleading. The model might seem secure against FGSM, but it will still be vulnerable to more complex attacks. The main problem is that it hides the real weaknesses in the system and doesn't prepare the model for stronger, smarter attacks.
- **Projected Gradient Descent (PGD):**
  PGD is a much more powerful and common attack. It works by making multiple small adjustments to the data, unlike FGSM, which makes just one big step. PGD ensures that the changes stay within a set limit (denoted by ϵ) after each step. This method helps the model learn to defend against more serious threats because it forces the model to adjust its decision-making boundary and become more robust. PGD is considered a top-tier method because it simulates a persistent, intelligent attacker who keeps trying different strategies to trick the system.

2.6 RESEARCH GAP IDENTIFICATION AND ARCHITECTURAL NOVELTY

Gaps in Existing Research

Despite what we know about adversarial robustness, there are still a few key issues when applying it to lightweight security systems. This project aims to address these gaps:

- **Gap 1: Lack of PGD Hardening in Lightweight Systems:**
  One big gap is that there aren't many examples of using **PGD-based adversarial training** for simple, lightweight models like Logistic Regression. Proving that

these models can stand up to sophisticated attacks is really important, especially for systems running on limited resources.

- **Gap 2: The Accuracy-Robustness Trade-Off:**
  Another challenge with adversarial training is that it often leads to a drop in performance on clean data (the accuracy-robustness trade-off). This project introduces a **Hybrid Ensemble Model** to tackle this. Rather than creating a new model from scratch, this approach combines the predictions of three models:
  - **Standard (accuracy-focused)**
  - **FGSM-Robust (defensive baseline)**
  - **PGD-Robust (highly resilient)**
    By assigning weights (50% PGD-Robust, 30% FGSM-Robust, and 20% Standard), the system maintains high accuracy on clean data while staying strong against adversarial attacks.
- **Gap 3: Need for Multi-Metric Evaluation:**
  The final gap is the lack of a comprehensive evaluation method that doesn't just use one metric. The proposed **4 * 3 evaluation matrix** fills this gap, testing the models under three types of conditions (clean, FGSM, and PGD attacks) using multiple performance metrics (like F1-Score and Recall).

Literature Survey Summary

The literature review shows a clear trend in research, from traditional Sybil detection methods to the rise of adversarial machine learning and the importance of **Explainable AI (XAI)** for trustworthy security systems.

- **A. Sybil Detection and ML Vulnerability:**
  Early Sybil detection methods focused on analyzing network behavior, but these methods had limitations. Newer techniques use machine learning models like **Ensemble Classifiers** and **Extreme Learning Machines (ELM)**, which have achieved high accuracy. However, these models are still vulnerable to **adaptive adversaries** — attackers that change their strategies over time. This research highlights the problem of building static models that fail when the attack changes.

- **B. Game Theory and Adversarial Hardening:**

  To counter the fixed threat model, researchers are shifting toward **Game Theory**, where the attack and defense are modeled as a game between two players. The **Fast Gradient Sign Method (FGSM)** is used to simulate adversarial attacks in training, forcing models to learn robust features. Studies show that while normal models fail under FGSM, adversarially trained models perform much better.

- **C. The Trust Gap in Robust Models:**

  While adversarial training helps improve robustness, it often makes models harder to understand, creating a **"trust gap"**. Security teams need to know why a model flags something as malicious. This project directly addresses this by integrating **Explainable AI (XAI)**, specifically using **SHAP (Shapley Additive Explanations)**, to make the model's decisions more transparent and understandable.

- **D. Research Novelty:**

  The key gap in existing research is that no one has yet combined **Sybil detection, adversarial training, and explainability** into one unified framework. This project fills this gap by:

  1. **Game-theoretic foundation:** Using **Min-Max Adversarial Game Theory** for robustness and **SHAP** for explainability.
  2. **Lightweight robustness:** Creating a lightweight, yet resilient model (using Logistic Regression) that can be deployed in resource-limited environments.
  3. **Trustworthy output:** Ensuring the model is not only robust but also interpretable, making it fit for deployment in real-world, critical infrastructure.

Chapter 3

System Design and Methodology

3.1 SYSTEM ARCHITECTURE OVERVIEW

Overview of the system architecture provides a concise overview of the features of the system under consideration and establishes the fundamental components of the system that ought to be developed during the project elaboration.<|human|>3.1 SYSTEM ARCHITECTURE OVERVIEW The overview of the system architecture gives the brief description of the peculiarities of the system in question and defines the main elements of the system that should be elaborated and created during the project development.

The proposed Lightweight Sybil Attack Detection System is implemented based on the multi-phase and event-based architecture that strictly decouples the issues of data processing, model training, adversarial simulation, and performance evaluation. The design is required so that the scientific validity of the experimental results and reproducibility of the robust models are addressed. The architecture is not just some set of modules; it is some idea of a pipeline which directly applies the Min-Max Adversarial Game theory to input data to the ultimate, measurable performance indicators.

The architecture is widely divided into the next interconnected layers, which are conceptually depicted in System Architecture Diagram:

1. Input and Data Management Layer: The data management layer will be involved in getting the raw network traffic data (NSL-KDD dataset), it will complete all the pre-processing required, such as feature engineering and normalization, and it will be in charge of creating the training and testing data splits. This layer will be essential to defining the credibility of the experiment as it will be using rigorous methods like SMOTE (Synthetic Minority Over-sampling Technique) and Min-Max Scaling to guarantee data parity and adequate representation of features in calculating the gradient.

2. Model Training Layer: This is the heart of the operation, the instantiated and trained various defender models are performed. It has specialized implementations of the Standard Model (Scikit-learn Logistic Regression), the FGSM Adversarial Model, and the PGD Adversarial Model (both implemented in PyTorch, which has the features of automatic differentiation). This layer directly implements min-max optimization loop in which the outer optimization is the minimization of the aggregate loss.

3. **Adversarial Generation Layer**: This functional component contains the task of simulation of the intelligent adversary (the inner maximization problem). It contains the exact algorithms of FGSM (Fast Gradient Sign Method) and the more complicated, iterative PGD (Projected Gradient Descent) generation of attacks. The module is necessary in both the development of the strong training addition and the creation of the vital adversary test sets that are applied in the eventual evaluation.

4. Hybrid Architecture Layer: this layer reflects the architectural innovation over the adversarial training on the baseline. It also uses the design and deployment of the Hybrid (Avg.) Ensemble Defender, which is the combination of the prediction probabilities of three specialist models (Standard, FGSM-Robust and PGD-Robust) based on a defensive weighted averaging approach instead of a stacking approach. This layer provides a balanced and better performance profile in both benign and adversarial threat setting by giving the more robust models a higher weight.

5. Evaluation and Output Layer: This is the last layer, which contains the full 4 3 Evaluation Protocol. It systematically executes all models against the three ready testbeds (Clean, FGSM Adv., PGD Adv.) and calculates the entire set of security performance measures, which are Accuracy, Precision, Recall, and the balanced F1-Score. The last product is the comparative matrix which is the key evidence of the project conclusions.

The big picture flow is focused on parallel processing in training and convergence in evaluation which gives a full understanding of adversarial resilience.

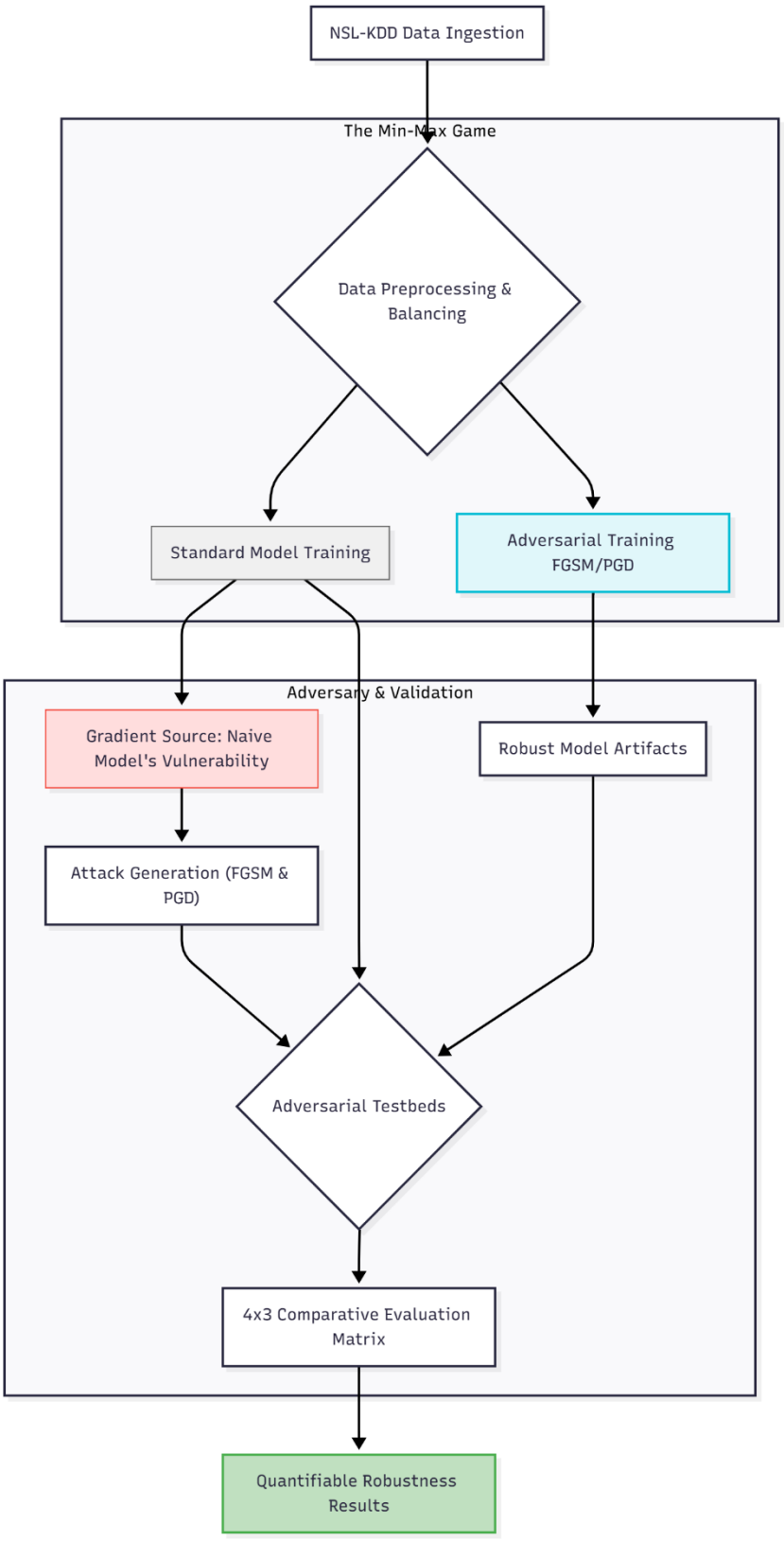


fig 1. System Architecture

3.2 FUNCTIONAL WORKFLOW OF THE PROPOSED SYSTEM

The functional workflow is a step-by-step process that's crucial for both training the models and testing how well they hold up against adversarial attacks. It takes the raw data through various stages, from preprocessing to the final evaluation, helping meet the project's goals.

Phase 1: Data Preparation and Baseline Establishment

1. **Data Ingestion and Feature Mapping:**

   The first step is to load the NSL-KDD dataset into the system. The dataset has 41 features, along with attack labels and difficulty scores. These features are mapped to more descriptive column names for easier understanding and analysis.

2. **Target-Transformation:**

   The attack labels in the dataset are originally multi-class, meaning there areseveral types of attacks. For simplicity, these are converted into a binary classification: **'normal'** becomes **0**, and all other attack types are grouped as **1**.

3. **Feature-Engineering-and-Encoding:**

   Some of the features in the dataset, like **protocol_type**, **service**, and **flag**, are categorical (non-numeric). These are transformed into numbers using **One-Hot Encoding**. This step is essential because models that use gradients (like Logistic Regression) can only work with numerical data.

4. **Feature Scaling:**

   All the numerical features are scaled to fall within the range of [0, 1] using **Min-Max Scaling**. This step ensures that features with large values (like **src_bytes**) don't dominate the calculations, allowing the **perturbation budget (epsilon)** to be applied evenly to all features.

5. **Imbalance Mitigation:**

   The **SMOTE** technique is applied only to the training data to fix the issue of class imbalance. This creates a balanced dataset (50% normal and 50% attack), so the models don't become biased toward predicting the majority class (normal data). This helps the model learn more fairly.

6. **Baseline Training:**

   Finally, the **Standard Logistic Regression Model** is trained on the balanced data

to set a **baseline performance**. This gives a reference point for how well the model performs under normal, non-adversarial conditions before moving on to more complex scenarios.

PHASE 2: ADVERSARIAL TRAINING AND HARDENING

1. Standard Path (Baseline):

The Standard Model serves as the control group. It's trained on the clean data without any adversarial examples to set a performance baseline.

2. FGSM Robust Training:

The next step is to train the FGSM-Robust Model. Here, I used the FGSM attack within the PyTorch framework. The training loop generates adversarial examples during training, and the model tries to minimize a combined loss function that balances both clean and adversarial losses. This helps the model become more resilient to simple adversarial attacks.

3. PGD Robust Hardening (Advanced Step):

The PGD-Robust Model takes things a step further by training against the PGD attack. This method uses a more complex, iterative process, simulating a smarter, persistent attacker. By applying these stronger attacks, the model becomes more resistant to advanced threats, making it much more resilient in real-world scenarios.

4. Hybrid Ensemble Construction:

The final part of Phase 2 is the creation of the Hybrid (Avg.) Model. Instead of stacking the models, which is another common approach, I combined the predictions of three models:

- Standard Model (focuses on accuracy)
- FGSM-Robust Model (focuses on baseline robustness)
- PGD-Robust Model (focuses on high robustness)

  The model uses a weighted averaging strategy, where each model's prediction is weighted differently. The PGD-Robust Model gets the highest weight (50%), followed by FGSM-Robust (30%), and Standard (20%). This approach helps

balance the system's ability to resist attacks while still maintaining good performance on clean data, tackling the classic accuracy-robustness trade-off.

PHASE 3: COMPREHENSIVE ADVERSARIAL EVALUATION

1. Attack Testbed Generation:

In this phase, I create three separate test sets from the original clean test data:

- Clean Test Set (normal data)
- FGSM Adversarial Test Set (attacked using FGSM)
- PGD Adversarial Test Set (attacked using PGD)
  Both adversarial sets are created using the gradients from the Standard Model to simulate how an attacker would try to fool the system.

2. Matrix Execution:

All four models (Standard, FGSM-Robust, PGD-Robust, and Hybrid) are systematically tested against each of these three test sets. This makes it a 4 * 3 evaluation matrix, where the goal is to measure how well each model performs across different conditions (clean data and two types of adversarial attacks).

3. Metrics Calculation:

For each of the 12 tests (4 models × 3 test sets), I calculate the performance metrics, including Accuracy, Precision, Recall, and F1-Score. These results are then compiled into a Comparative Performance Table, which helps compare the models' performance under both normal and adversarial conditions.

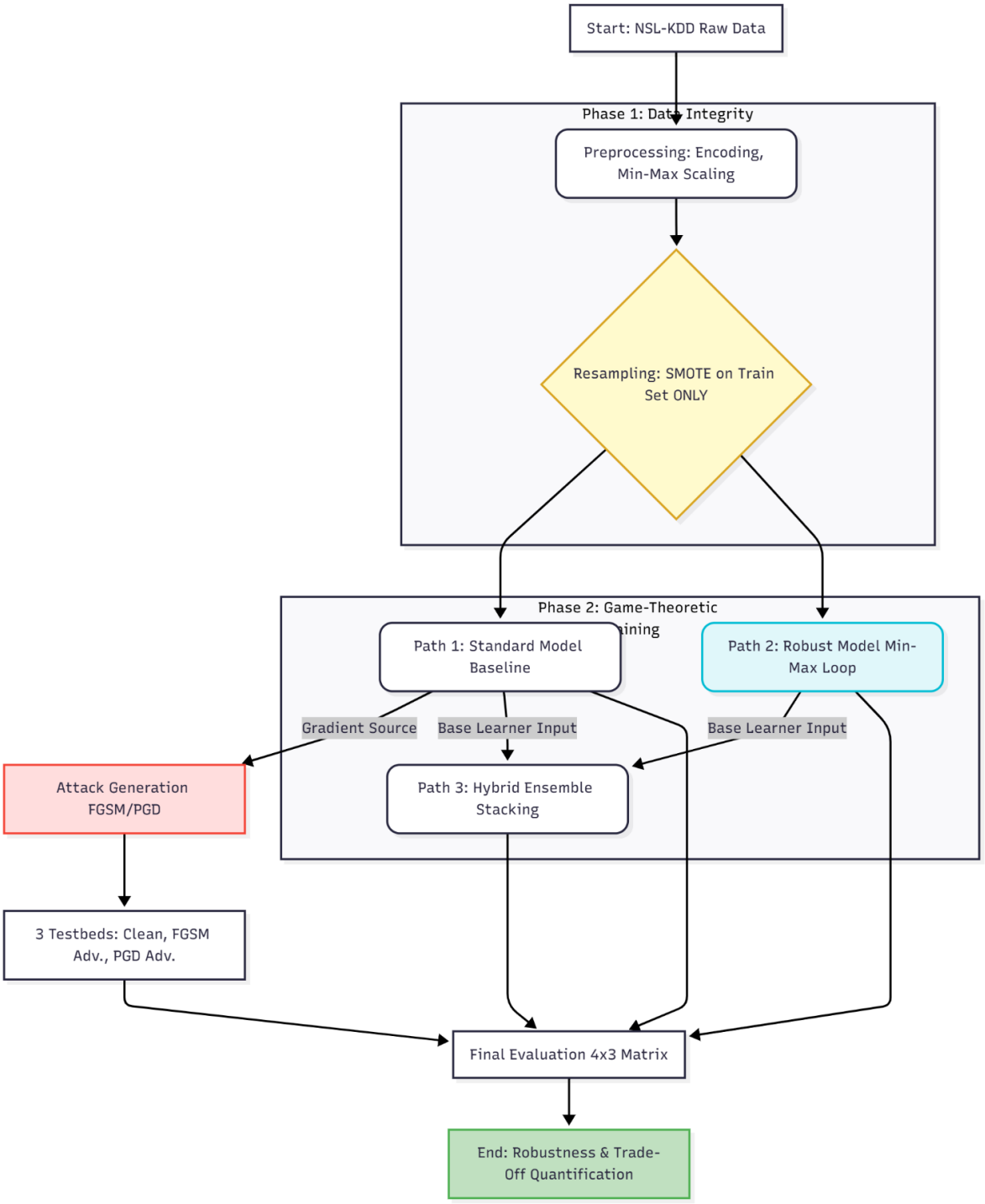


fig 2. Work Flow diagram of the Proposed System

3.3 SYSTEM IMPORTANT COMPONENTS

The effectiveness of the system and its ability to prove the game-theoretic approach depend on several key technical and methodological components. These components together form the "system design" in a computational security context.

3.3.1 THE MIN-MAX FORMULATION CORE

At the heart of this project is the **Min-Max formulation**, which is the foundation for minimizing the worst-case loss. This principle creates the need for two interlocking processes:

- **Inner Loop (Attack):** This computes the gradient to find the optimal way for the attacker to evade detection.
- **Outer Loop (Defense):** This minimizes the model's loss when trying to counter the attack.

These two loops work together to build the model's resilience.

3.3.2 MODEL ARCHITECTURES (THE SPECIALISTS)

The choice of model architectures is key to the experimental design. I used **Logistic Regression** for all models to keep things simple and ensure a fair comparison. This helps show the model's vulnerabilities before applying adversarial techniques. The **Scikit-learn** library makes training the baseline model easy, but for the more robust models, I used **PyTorch**. PyTorch is essential because it gives fine control over gradient calculations, which is necessary for training against adversarial attacks like FGSM and PGD.

3.3.3 ADVERSARIAL ATTACK FUNCTIONS (FGSM AND PGD)

The **FGSM** and **PGD** functions are where the attack happens. These functions are implemented in PyTorch to simulate how an adversary manipulates the data.

- **FGSM:** This is a simple, single-step attack that quickly computes the optimal change to the data.
- **PGD:** This is a more advanced attack that iteratively tweaks the data, refining the changes each time to make the attack harder to detect.

PGD is particularly important because it represents a more sophisticated adversary and gives a stronger defense benchmark compared to FGSM.

3.3.4 ARCHITECTURAL SPECIALISTS AND THE HYBRID ENSEMBLE

The architecture is built around three specialized models:

- **Accuracy Specialist:** The **Standard Model**, focused on normal data (benign traffic).
- **Baseline Robustness Specialist:** The **FGSM-Robust Model**, which is resilient to simple attacks.
- **High-Robustness Specialist:** The **PGD-Robust Model**, built to handle more advanced threats.

Instead of stacking these models in a traditional way (which would require a meta-learner), I combined their predictions using a **weighted averaging strategy**. This means each model contributes to the final prediction based on how robust it is:

- **PGD-Robust Model:** Gets the highest weight (50%) because it's the most resistant to attacks.
- **FGSM-Robust Model:** Gets a weight of 30%.
- **Standard Model:** Contributes the least (20%) since it's the least robust.

This approach balances the system's ability to defend against adversarial attacks while still performing well on clean data, resolving the accuracy-robustness trade-off without adding complexity.

3.3.5 THE EPSILON (ϵ) HYPERPARAMETER

The **epsilon (ϵ)** value is a key part of the design. It controls the strength of the adversarial perturbation (the attacker's power). Choosing the right epsilon is important:

- It needs to be small enough to keep the attack subtle and realistic.
- But, it must also be large enough to challenge the model's robustness.

In the implementation, I set epsilon to **0.05**, which provides a meaningful test of the model's ability to handle adversarial changes without overly distorting the data.

3.3.6 THE HYBRID ENSEMBLE

The **Hybrid (Avg.) Ensemble** is a major innovation in this system. Rather than using a complex stacking method or training a meta-learner, this ensemble combines the predictions of three models using a **weighted averaging strategy**. Each model's contribution is based on its robustness:

- **PGD-Robust Model** gets the highest weight (50%) to lead the defense.
- **FGSM-Robust Model** follows with 30%.
- **Standard Model** gets 20%.

This strategy gives the system a balanced performance, strong against both adversarial and normal data, without needing a separate meta-classifier. It's a **computationally lightweight** solution to the accuracy-robustness trade-off.

3.4 MODULE DESIGN

The project is organized in discrete modules and each module has a specific stage in the adversarial methodology.

3.4.1 DATA PREPROCESSING MODULE

All logic of data transformation is contained in this module. Its important sub-components deal with the loading, feature mapping, categorical encoding and feature scaling. The most significant of its capabilities is the administration of the SMOTE resampler so that it is implemented solely on the training data to reduce the impact of class disparity but strictly adhering to the rule of avoiding information leakage to the evaluation data.

3.4.2 THE MODEL ARTIFACT MODULE IS A STANDARD AND ROBUST ONE

This module gives the Python models of the models, including StandardModel (Scikit-learn wrapper) and the RobustModel (PyTorch Logistic Regression). It offers ways of training the model, storing the model weights, and keeping the architecture constant (Logistic Regression) in all versions so as to isolate the effect of the training methodology as the sole independent variable.

3.4.3 ADVERSARIAL TRAINING MODULE (FGSM/PGD)

The main innovation is contained in this module. It controls the two different adversarial training loops of the FGSM and PGD models. It does this by repeatedly calling the corresponding attack functions and running the backward pass of the combined loss function. This module generates the two robust models, fgsm and pgd robust models, fgsm and pgd robust models.

3.4.4 HYBRID STACKING MODULE

This is the advanced ensemble element. It accepts as inputs the output of the Standard and PGD models and coordinates the Cross-Validation (CV) process needed to train the meta-learner. In this case, CV is required to produce out-of-fold predictions, which are the clean, unbiased training data of the meta-learner to avoid a data leakage referred to as overfitting on the generalization error.

3.4.5 EVALUATION AND METRICS MODULE

The validation stage is regulated by this last module. It takes care of the creation of the three needed testbeds and holds the functionality that is needed to execute the 4 x 3 matrix. It computes the performance indicators and tabulates the ultimate quantitative findings to the presentation-ready tables in Chapter 5.

3.5 DATA FLOW & UML DIAGRAMS

The success of the methodology depends on how well the data and control flow through the system's various components. Visualizing this flow is essential to show how the different parts work together in the report.

3.5.1 DATA FLOW EXPLANATION

The data flow starts in a linear fashion, beginning with cleaning and scaling the data, but it splits into parallel paths during training:

1. **Parallel-Training:**

   The **clean data** is passed through the **Standard Model** training path. Simultaneously, the data is also sent through the **Adversary ↔ Defender loop** to

train the **Robust Models**. This loop continuously interacts with adversarial examples to improve the models' resilience.

2. **Testbed-Generation:**

   Once the **Standard Model** is trained, it is used as the **Gradient Source** to generate the three required test datasets:

   - **Clean Test Set**
   - **FGSM Adversarial Test Set**
   - **PGD Adversarial Test Set**

These test sets simulate different attack scenarios to evaluate the model's robustness.

3. **Convergent-Evaluation:**

   In the final evaluation phase, all **four trained models** are tested on the three test sets. This allows a **rigorous comparison of performance** across all **12 scenarios** (4 models × 3 test sets). This step is the key to demonstrating the success of the game-theoretic approach.

3.5.2 USE CASE DIAGRAM FOR SYSTEM INTERACTIONS

In this section, I use a **UML Use Case Diagram** to visually represent the functional requirements and system boundaries of the game-theoretic detection framework. The **Use Case Diagram** clearly shows the interactions between different actors, including the **human researcher**, **software-driven models**, and the **abstract adversary logic**. It highlights the high-level strategic flow of the project, showing how the different components work together.

The **Use Case Diagram** divides the system's functions into two major groups:

1. **Experimental Setup (managed by the human actor)**: This part includes tasks like data preprocessing and final analysis of the game-theoretic approach.
2. **Strategic Conflict (executed by conceptual actors)**: This represents the ongoing interaction between the defender and the attacker, simulating a strategic conflict.

KEY ACTORS AND THEIR ROLES:

1. **Researcher/Developer:**

   This is the human actor who ensures the integrity of the experiment. They are responsible for initiating the data preprocessing, training the models, and

analyzing the results to quantify the success of the game-theoretic approach.

2. **Standard-Model(Baseline-Defender):**

   The Standard Model represents a conventional security approach. It is used to establish a baseline for vulnerability and provides output gradients that the **Adversary** uses in their attack.

3. **Adversarial-Attacker(FGSM/PGD):**

   This conceptual actor represents the **intelligent adversary** trying to break the system. Its role is to execute the **inner maximization problem**, which is the **"max"** in the **Min-Max** game.

4. **Robust-Model(Game-Theoretic-Defender):**

   This conceptual actor embodies the main goal of the project: creating a model that learns to build a resilient decision boundary by executing the **Min-Max optimization** (defending against attacks while minimizing vulnerabilities).

5. **Evaluation-Module:**

   The **Evaluation Module** is a neutral system actor that rigorously tests all the models and quantifies the outcomes of the **strategic conflict**. It provides the performance metrics used to evaluate the success of the project.

USE CASE INTERACTIONS (STRATEGIC FLOW):

The strategic flow describes how the system components interact during the adversarial training process. Here's a breakdown of how each actor plays their role in the game-theoretic methodology:

- **Baseline-Establishment:**

  The Researcher begins by training the Standard Model, which serves as the baseline. The Researcher also defines the Standard Model's role as the Attack Gradient Source. This step is crucial because it creates the vulnerability that the Adversarial Attacker will exploit to generate realistic, black-box test sets.

- **Min-Max-Execution:**

  The Adversarial Attacker continuously works to generate evasion perturbations that maximize the model's loss. These attacks push the Robust Model to execute the Min-Max Optimization, which helps it adjust its defensive parameters to

mitigate the attack and reduce vulnerabilities. Essentially, this is the back-and-forth of the game—the attacker tries to get past the defense, and the defense adapts to counter the attack.

- **Quantification:**

  Once the Min-Max Execution is completed, the process flows into the Evaluation Module, which computes the 4x3 comparative metrics. This step provides the final, quantifiable evidence that demonstrates how well the model performs under different conditions. The Researcher/Developer then analyzes these results to assess the model's robustness and the accuracy-robustness trade-off.

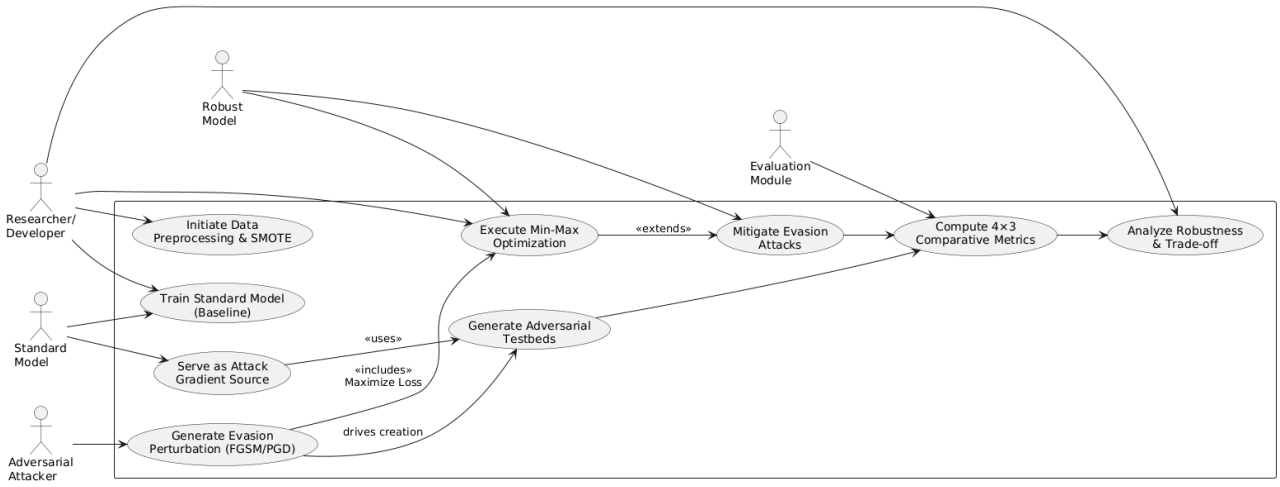


fig 3. Use Case Diagram for System Actors and Strategic Interactions

3.5.3 **CLASS DIAGRAM** FOR CORE COMPONENTS

The modularity of the system, technical structure and dependencies needed to run the game-theoretic training are formally modeled with the help of Class Diagram on the basis of the Python implementation files of the project. This diagram shows the fundamental elements, properties, procedures and interrelationships among the data management, model training, adversarial simulation, and final evaluation software entities. This is a static perspective of the architecture that is needed in showing the rigor of the implementation of the project.

KEY COMPONENT CLASSES:

The adversarial methodology consists of a number of fundamental classes, each playing a specific role of the structure:

• **DatasetManager:** This is a class which acts as the data pipeline controller and contains all the preprocessing logic required. It also handles the raw data, performs some important scientific integrity processes like the apply_SMOTE() method to class balance and formats the final X train smote and y test clean artifacts to be trained and assessed. It has a training dependency with the main model classes.

• **StandardModel:** This is the naive defender or accuracy expert. It does use the scikit-learn LogisticRegression architecture and applies standard training techniques, providing the maximum performance limit and the minimum resistance against which the entire robustness profits are relative.

• **RobustModel:** The game-theoretic defender or the expert of robustness. It is implemented as a set of PyTorchLogisticRegression, with train_adversarial being the main procedure that implements the complicated min-max optimization cycle. This type of relationship is shown on the specific attacker classes with explicit dependency (aggregation) relationship.

• **FGSM Attacker and PGD Attacker:** These are the classes that represent the intelligent opponent and the mathematical essence of the operation of max. They have a common feature set (such as \epsilon or perturbation budget) and provide the specialised run attack() function. These classes are used to build adversarial training batches on the fly in the RobustModel.

• **HybridEnsemble:** This class describes the defensive weighted-average ensemble, which is an ensemble that combines the prediction probabilities of three expert models: the StandardModel, the FGSM-RobustModel and the PGD-RobustModel. It does not need to train a meta-model (as in stacking): it uses predefined weights based on

robustness to each model output and sums them together to reduce the AccuracyRobustness trade-off. PGD-RobustModel is the most weighted and maintains high levels of adversarial resilience but still enjoys the accuracy of the clean data of the StandardModel.

• **EvaluationModule:** This is the last class, which coordinates the validation procedure. It contains the procedures of generate testbeds and run evaluation, the results of which is the quantitative dict results, which is the core evidence of the project called results matrix.

Types of relationships and Rigor of implementation:

The Class Diagram identifies key dependencies which impose methodological rigor:

1. Dependency (Association): The DatasetManager is associated with the model classes meaning that it supplies the input data required in the process of training.

2. Inheritance (Generalization): FGSM Attacker and PGD Attacker are naturally derived off of a conceptual BaseAttacker, providing a uniform interface to adversarial generation.

3. Aggregation (Composition): The RobustModel shows that it is closely associated with the attacker classes (marked with the label uses in the diagram), which implies that the attack logic is an indispensable part of the implementation of the adversarial training method.

4. Artifact Flow: The output dependencies are indicated by lines marked with provides artifacts but the lines indicate the direct relationship between the trained models and the ensemble and the final EvaluationModule. This guarantees that the outcomes can be traced directly to the version of the trained models.

The large-scale security research is crucial to such modular design because it separates the components to test, and to swap attack algorithms (e.g. replace FGSM with PGD) easily.

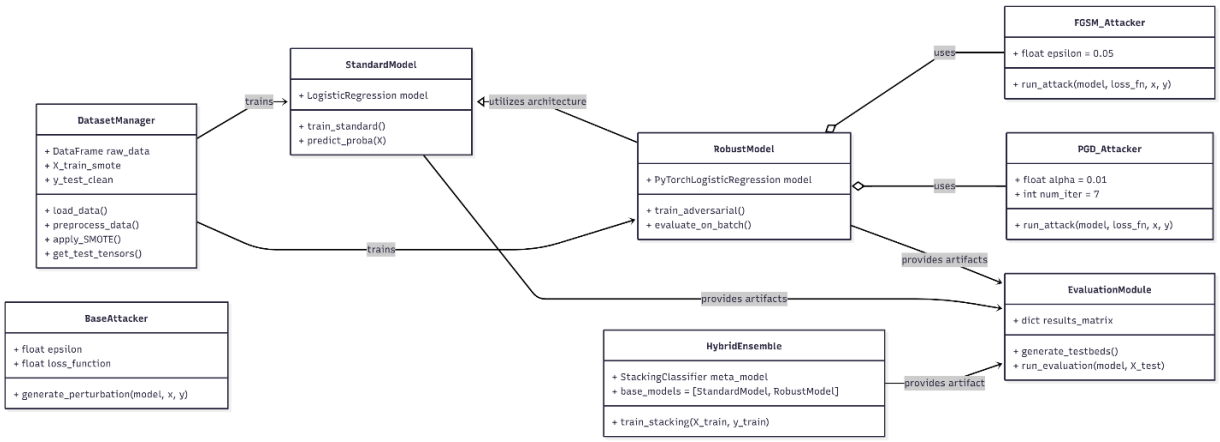


fig 4. Class Diagram of Project Components and Dependencies

3.5.4 **SEQUENCE DIAGRAM** OF ADVERSARIAL TRAINING (THE MIN-MAX GAME)

The sequence diagram provides a clear view of how the system behaves during the adversarial training process. It shows the turn-taking between the components—DataLoader, Attacker Logic, Robust Model, and Optimizer—and how they interact to implement the min-max principle.

1. Data Load:

The **DataLoader** provides the clean input data (**x_clean**) and the true label (**y**) to the system. This is the first step in feeding the raw data into the pipeline.

2. Attacker Maximization (Inner Max):

The Robust Model passes the clean data (x_clean) to the Attacker Logic. The Attacker uses the model's gradients to compute the optimal evasion (delta), which perturbs the data to make it adversarial. The perturbed example (x_adv) is returned to the Robust Model.

3. Defender Minimization (Outer Min):

The **Robust Model** calculates the **combined loss (J_combined)**, which is the sum of the loss on the clean data (**x_clean**) and the adversarial example (**x_adv**). The goal is to

minimize this combined loss, making the model more resilient to both clean and adversarial data.

4. Parameter Update:

The **Optimizer** receives the loss gradient and updates the model's weights (**theta**). This step helps the model adjust its parameters to defend against the attack by learning from the adversarial example.

This loop repeats for every batch, continuously making the model more resilient to adversarial attacks.

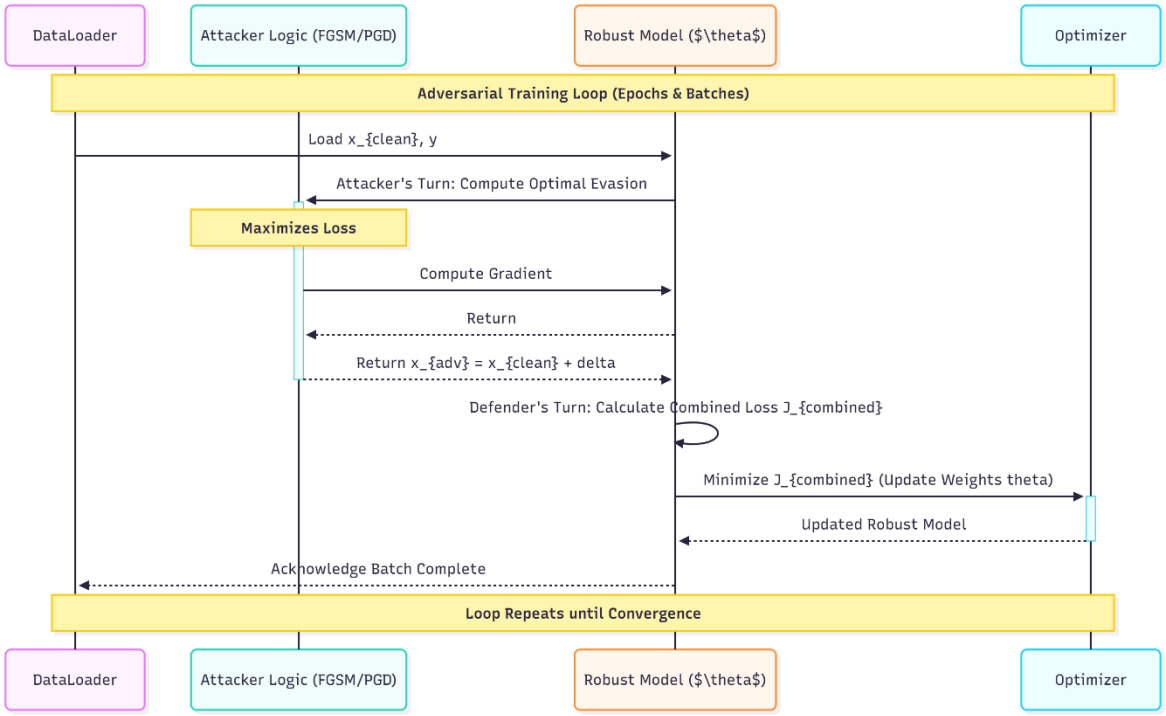


fig 5. Sequence Diagram

3.6 DATASET AND PREPROCESSING PIPELINE

The integrity of the experimental results depends heavily on how the dataset is handled and preprocessed. This section explains the steps taken to ensure the data is properly prepared for training.

3.6.1 THE NSL-KDD DATASET JUSTIFICATION

The **NSL-KDD dataset** is chosen because it is an industry-standard benchmark for evaluating **Network Intrusion Detection Systems (NIDS)**. It improves on the older **KDD'99 dataset** by eliminating redundant records that caused problems in previous research. While the dataset doesn't specifically label **Sybil attacks**, the different types of attacks (DoS, Probe, U2R, and R2L) mimic the "Sybil Effect," where multiple fake identities are used by an adversary. The training set (**KDDTrain+.txt**, with around 126k records) and the test set (**KDDTest+.txt**, with about 22k records) are used throughout the experiments.

3.6.2 DETAILED PREPROCESSING STEPS

1. **Binary-Mapping:**

   The multiple attack categories are collapsed into a **binary target**: **0** for **Normal** and **1** for **Attack**. This simplifies the model's focus on detecting malicious activity versus benign traffic.

2. **Categorical-Encoding:**

   The symbolic features (**protocol_type**, **service**, **flag**) are converted into numerical values using **One-Hot Encoding**. This adds 81 new binary features, expanding the dataset's feature space from 41 to 122 dimensions. This is important because gradient-based methods require numerical input.

3. **Feature-Scaling(Min-Max):**

   This step scales all 122 numerical features to a range between [0, 1] using **Min-Max Scaling**. Scaling ensures that the **epsilon (ϵ)** perturbation budget is applied consistently across all features, making the adversarial attack meaningful and fair.

4. **Imbalance-Mitigation(SMOTE):**

   The **SMOTE** technique is applied to the training data to balance the classes. It generates synthetic samples for the minority class to ensure a 50-50 class balance. This prevents the model from being biased toward the majority class (normal data) and helps improve **Recall**. Importantly, **SMOTE** is only applied to the training data to avoid **data leakage** into the test set.

3.7 ADVERSARIAL TRADE-OFF ANALYSIS (TABLE)

The comparison of training methodologies—Standard versus Adversarial—is critical as it highlights the fundamental operational difference between a reactive and a proactive security system. Table below summarizes these conceptual differences, but a full discussion is required to explain the tangible costs and benefits associated with implementing the game-theoretic approach.

3.7.1 PRIMARY OBJECTIVE AND STRATEGIC POSTURE

The Standard Model's objective is strictly to maximize accuracy on the known, clean data distribution. This leads to a reactive posture, where the model only learns from patterns already seen. Conversely, the Adversarial Model's objective is to minimize loss against the worst-case scenario. This forces a proactive posture, where the model anticipates and counters the attacker's optimal moves during the training phase itself.

3.7.2 DECISION BOUNDARY CHARACTERISTICS

The pursuit of high accuracy in the Standard Model often results in a brittle and tight decision boundary. This boundary perfectly separates the training examples but is highly sensitive to minimal perturbations, making the model vulnerable to evasion. Adversarial training, by incorporating perturbed examples, acts as a regularization technique. It forces the model to learn a smooth and resilient decision boundary, where small variations in input features do not easily flip the classification result.

3.7.3 THE COST-BENEFIT DUALITY

Implementing adversarial training introduces a duality of cost and gain that defines the **Accuracy-Robustness Trade-Off**:

- **Cost Incurred (Accuracy Trade-off):** Training against adversarial examples typically incurs a slight loss in performance on clean, non-adversarial data. This is the cost of robustness, occurring because the model sacrifices some of its ability to perfectly fit the nuances of the clean data distribution in favor of generalizing its defense. This cost must be weighed against the significant security gain.

- **Vulnerability vs. Resilience (Security Gain):** The Standard Model exhibits high vulnerability, collapsing catastrophically under attack. The Adversarial Model, however, achieves high resilience, maintaining a strong detection rate when exposed to the same evasion attacks. Quantifying this dramatic gain (as seen in the comparative results) is the primary validation of the game-theoretic approach. The comparison between the **Standard Model** and the **Adversarial Model** is crucial because it highlights the differences between a **reactive** security system and a **proactive** one. The table below summarizes these differences, but a deeper discussion is needed to explain the costs and benefits of using the game-theoretic approach.

3.7.1 PRIMARY OBJECTIVE AND STRATEGIC POSTURE

- The **Standard Model** aims to **maximize accuracy** on clean data, making it **reactive**. It only learns from patterns seen during training.
- The **Adversarial Model** aims to **minimize loss** against the worst-case scenario, forcing it to be **proactive** by anticipating and countering the adversary's optimal moves during training.

3.7.2 DECISION BOUNDARY CHARACTERISTICS

- The **Standard Model** has a **tight decision boundary** that perfectly separates training data, but it's **brittle**. Even small changes in the input can cause misclassification.
- **Adversarial training** helps the model learn a **smoother, more resilient decision boundary**. It regularizes the model so that small variations don't easily flip the classification.

3.7.3 THE COST-BENEFIT DUALITY

- **Cost-Incurred(Accuracy-Trade-off):**

  Training on adversarial examples typically results in a small **accuracy loss** on clean data. This is the cost of robustness, as the model sacrifices some accuracy on clean data to better handle adversarial attacks.

- **Vulnerability vs. Resilience (Security Gain):**
  The **Standard Model** is vulnerable and collapses under adversarial attacks. In contrast, the **Adversarial Model** is resilient, maintaining a high detection rate even when exposed to attacks. The difference in security is the primary benefit of adversarial training.

A	B	C
Training Characteristic	**Standard Model Training**	**Adversarial (Min-Max) Training**
Primary Objective	Maximize Accuracy on Clean Data	Minimize Loss on Worst-Case Adversarial Data
Model Posture	Reactive (Assumes fixed distribution)	Proactive (Anticipates opponent's strategy)
Decision Boundary	Brittle and Tight	Smooth and Resilient
Vulnerability	High (Catastrophic collapse under attack)	Low (Maintains performance under attack)
Cost Incurred	Low Computational Cost	Accuracy Trade-Off (Slight loss on clean data)

Table 1. Shows Training Characteristic between Standard vs Adversarial Model Training

3.8 METHODOLOGY FLOWCHART

The entire research process—from data ingestion to final analysis—is best summarized visually through a **methodology flowchart**. This flowchart illustrates the steps and decisions that occur during the adversarial training and evaluation process.

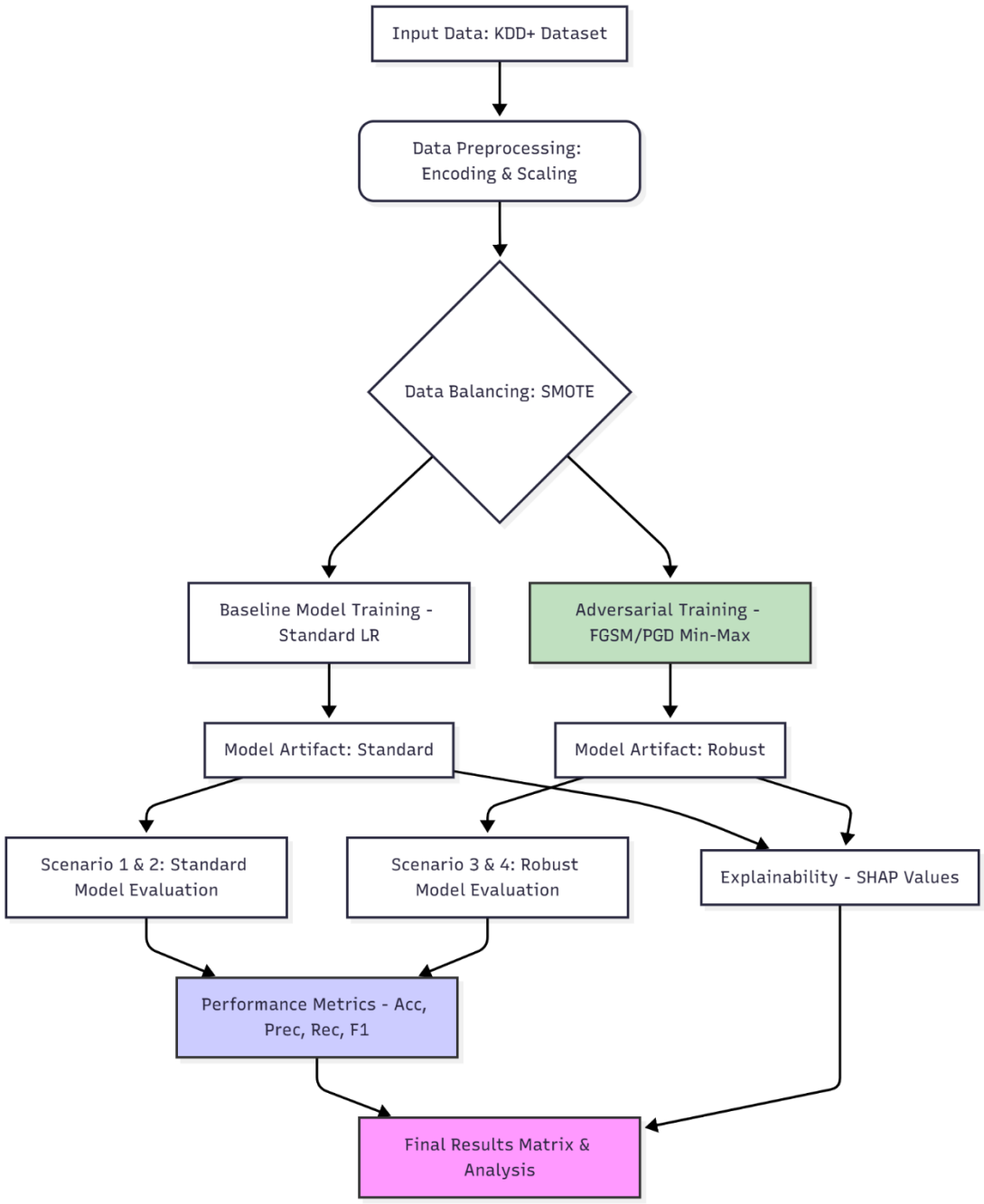


fig 6. Complete System Flowchart

3.8.1 EXPLANATION OF THE FLOWCHART

1. **Initial-Data-Pipeline(Linear-Flow):**

   The process begins with loading the raw **KDD+ Dataset**. After preprocessing (encoding and scaling), the data goes through **SMOTE** for balancing.

Importantly, **SMOTE** is applied only to the training data to prevent leakage.

2. **Parallel Training Paths:**

   The balanced data splits into two parallel paths:

   - **Baseline Training (Standard LR):** Trains the traditional model(vulnerable model).
   - **Adversarial Training (FGSM/PGD Min-Max):** Trains the robust model using adversarial examples.

3. **Convergent Evaluation:**

   Both model paths converge for evaluation:

   - Models are tested in different scenarios (Standard Model and Robust Model evaluations).
   - **Explainability (SHAP Values):** SHAP values are used to explain the model's decision-making logic.

4. **Final-Analysis:**

   The evaluation metrics (Accuracy, Precision, Recall, F1) and SHAP values are used to generate the **Final Results Matrix**, which quantifies the improvements in adversarial robustness and analyzes the accuracy-robustness trade-off.

3.8.2 PROJECT CONNECTION AND SYNTHESIS

The flowchart visually represents the entire project strategy. Each step in the chart represents a critical objective, and the arrows show how each part depends on the previous one. The chart confirms that the project's main goal is not just to train a model but to **quantify the impact of the game-theoretic approach** in improving security.

Chapter 4

System Implementation & Workflow Execution

4.1 OVERVIEW OF SYSTEM IMPLEMENTATION

The **Lightweight Sybil Attack Detection System** is a practical application of the **Min-Max Adversarial Game** framework introduced in Chapter 3. The system is built using Python, with a carefully selected stack of libraries to handle the different tasks needed for standard classification and adversarial training.

4.1.1 IMPLEMENTATION ENVIRONMENT AND CORE TECHNOLOGIES

The project was developed in a high-performance computational environment (e.g., **Google Colaboratory**) using **Python 3.x**. The key technologies used to handle the complexity of adversarial training are:

- **Scikit-learn:** This library was used for conventional tasks like training the Standard Logistic Regression Model (the baseline defender) and preprocessing functions such as MinMaxScaler and LogisticRegression.
- **PyTorch:** PyTorch is essential for implementing the **game-theoretic logic**. Its automatic differentiation feature allows us to compute the gradients of the loss function (i.e., $\nabla_x J$), which is required for generating adversarial examples using **FGSM** and **PGD**.
- **Imbalanced-learn (imblearn):** This library was used for the **SMOTE** function, ensuring the class balance of the training dataset to avoid biases in the model.
- **NumPy and Pandas:** These are used for fundamental data manipulation, structuring the dataset, and compiling the final performance results.

4.1.2 THE TWO-PATH EXECUTION PRINCIPLE

The system operates on a **dual execution principle**, which compares two distinct defensive strategies:

1. **The-Naïve-Path(Scikit-learn):**

   This path trains the **Standard Model**, which serves as the control group in the experiment. It's simple to implement but highly vulnerable, making it the baseline for measuring the impact of adversarial training.

2. **The-Robust-Path(PyTorch):**

   This path implements the **adversarial training loop** using PyTorch. The **RobustModel** is structured using a custom **PyTorch Logistic Regression** class, allowing for gradient computations needed for adversarial defense, aligning with the **game-theoretic** approach.

In the final phase, these two paths converge, and both model artifacts (Standard, FGSM Adv., PGD Adv., and Hybrid) are evaluated against the same clean and adversarial test environments.

4.2 DATA PREPARATION AND CODE SNIPPETS

The data preprocessing pipeline is critical to ensure the integrity of the experimental results. Any flaws in preprocessing, like improper scaling or imbalance handling, would invalidate the entire adversarial experiment. Here's a breakdown of the steps taken:

4.2.1 LOADING, LABELING, AND BINARY MAPPING

The first step was loading the raw data from KDDTrain+.txt and KDDTest+.txt files. These files have no headers, so we manually assigned the correct feature names. The target variable was then transformed into a binary format, where:

- 'normal' was mapped to 0 (benign traffic).
- All other attack types (DoS, Probe, U2R, R2L) were mapped to 1 (attack traffic).

This binary target aligns the project with a **binary classification** task: detecting whether traffic is normal or an attack.

4.2.2 CATEGORICAL FEATURE ENCODING

The **NSL-KDD dataset** includes three categorical features: **protocol_type**, **service**, and **flag**. Since these are non-numeric and incompatible with gradient-based models, they were transformed into numerical values using **One-Hot Encoding**. This step added **81 new binary features**, expanding the feature space from 41 to 122 dimensions. This expansion is crucial because **gradient-based models** like Logistic Regression require all features to be numerical.

4.2.3 CRITICAL STEP: FEATURE SCALING WITH MIN-MAX

Feature scaling isn't just for performance improvement—it's essential for adversarial resilience. All **122 numerical features** were scaled to a range between **[0, 1]** using **Min-Max Scaling**.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Why-Min-Max?

This scaling ensures that features with large values (like src_bytes) don't dominate the gradient calculations, making the adversarial perturbation (ϵ) applied uniformly across all features. Without proper scaling, the perturbation could disproportionately affect some features, undermining the effectiveness of the adversarial attacks.

4.2.4 INTEGRITY STEP: CLASS IMBALANCE MITIGATION WITH SMOTE

The NSL-KDD dataset has a severe class imbalance—normal data points significantly outnumber attack data points. To prevent the model from biasing towards the majority class, I applied the SMOTE (Synthetic Minority Over-sampling Technique) only to the training set. This helps generate synthetic examples for the minority class (attacks), balancing the class distribution.

Why SMOTE only on training data?

Applying SMOTE only to the training data prevents data leakage from the test set into the training process. If we applied it to the test set, it would give an overly optimistic view of the model's performance, invalidating the results.

A	B	C
Data Set	**Original Distribution**	**After SMOTE (Train Only)**
Training (y_train)	67,343 Normal, 58,630 Attack	67,343 Normal, 67,343 Attack
Testing (y_test)	9,711 Normal, 12,833 Attack	9,711 Normal, 12,833 Attack (Untouched)

Table 2. Dataset Analysis. Original Distribution vs After SMOTE

Scientific Integrity Constraint: SMOTE was applied exclusively to the training data (X_train_smote, y_train_smote). This is a non-negotiable step to prevent data leakage, where synthetic information derived from the test set could contaminate the training set, leading to an overly optimistic and invalid assessment of the model's generalization ability.

4.3 BASELINE MODEL (STANDARD LR) TRAINING

The **Standard Model** serves as the **accuracy specialist** and plays a crucial role as the **experimental control** in this project.

4.3.1 ARCHITECTURAL AND IMPLEMENTATION RATIONALE

The baseline model is built using Logistic Regression from Scikit-learn. This choice was deliberate: Logistic Regression is a simple, lightweight linear model, and its simplicity makes it particularly vulnerable to gradient-based adversarial attacks. This vulnerability is key, as it ensures the model experiences a clear performance drop when subjected to adversarial examples, making it the perfect reference for testing the effectiveness of the robust models.

The Standard Model was trained using the pre-processed and SMOTE-balanced training data, which ensures the model is not biased toward the majority class (normal data).

A	B	C
Parameter	**Value**	**Description**
Model	LogisticRegression (Scikit-learn)	Naive, accuracy-optimized defender.
Data	X_train_smote, y_train_smote	Balanced training set.
Max Iterations	1000	Ensured convergence with scaled features.

Table 3. States Model values and Description

4.3.2 BASELINE ARTIFACT AND PURPOSE

The **standard_model artifact** represents the conventional, non-robust security approach. Its expected role is twofold:

1. **Set the Performance Ceiling:**

   The **Standard Model** is expected to achieve the **highest possible metrics** on the **Clean Test Set** (i.e., the non-adversarial, normal data). This sets the baseline for comparing how well the robust models perform in more challenging, adversarial conditions.
2. **Serve as the Attack Gradient Source:**

   The Standard Model also plays a critical role in generating adversarial examples. The PyTorch equivalent of this model (trained separately) is used to generate all the adversarial test sets, simulating the scenario where an attacker targets a commonly deployed, non-robust defender. This step is essential to demonstrate the difference in performance when the model faces adversarial attacks.

4.4 FGSM ATTACK AND ADVERSARIAL TRAINING IMPLEMENTATION

This module implements the **game-theoretic defense** by using the **Fast Gradient Sign Method (FGSM)** as the baseline adversarial attack.

4.4.1 FGSM ATTACK FUNCTION (FGSM_ATTACK)

The **FGSM function**, implemented in **PyTorch**, allows the system to calculate the **inner maximization** in a single, efficient step.

$$\max_{\|\delta\| \leq \epsilon} L(\theta, x + \delta, y)$$

FGSM IMPLEMENTATION STEPS:

1. Enable Gradient Tracking:

   First, the input data (x) is set to track gradients by using x.requires_grad = True. This allows us to compute the gradient of the loss function with respect to the input data.

2. Forward/Backward Pass:

   The forward pass calculates the loss, and then a backward pass is performed using

loss.backward() to compute the gradient ($\nabla_x J$) of the loss with respect to the input data.

3. Perturbation Calculation:
   The adversarial perturbation (δ) is calculated as the sign of the gradient multiplied by the budget (epsilon):
   δ = epsilon * data_grad.sign()
   This means the perturbation is a small adjustment to the input data, aiming to make the model misclassify it.

4. Clamping:
   After the perturbation is applied, the perturbed input (x_adv) is clipped to stay within the valid range [0, 1] using torch.clamp(x_adv, 0, 1). This ensures the adversarial example remains valid and realistic.

4.4.2 THE FGSM-ROBUST TRAINING LOOP

The **training loop** for the **fgsm_robust_model** implements the **outer minimization** of the **Min-Max Game**. Here's how it works:

1. The model cycles through **batches** of data.
2. For each batch, it first runs the **Attacker's move (FGSM)** to generate adversarial examples.
3. Then, it executes the **Defender's countermove** by updating the model's **parameters** (i.e., adjusting the model's weights to better handle the attack).

Key Hyperparameters:

- **epsilon (ϵ_FGSM):** Set to **0.05**, this defines the **size of the stealthy attack budget**. It controls how much the data is perturbed.
- **alpha (α_LOSS):** Set to **0.5**, this balances the **clean loss** and **adversarial loss** in the combined objective function, giving equal weight to both types of data.

THE COMBINED LOSS FUNCTION:

$$J_{combined} = \alpha J(\theta, x_{clean}, y) + (1 - \alpha) J(\theta, x_{adv}, y)$$

By minimizing this combined loss, the model is forced to learn a decision boundary that is robust not just to the original data, but to the worst-case attacks generated on the fly.

4.5 PGD ATTACK AND ROBUST TRAINING LOGIC

The **PGD (Projected Gradient Descent)** implementation is the most advanced defense in this project. It addresses the limitations of **FGSM**, such as dealing with **obfuscated gradients**.

4.5.1 THE PGD ATTACK FUNCTION (PGD_ATTACK)

PGD takes advantage of an **iterative, persistent adversary**. Unlike FGSM, which makes a single-step perturbation, PGD applies multiple small steps to gradually craft a stronger adversarial example.

PGD Implementation Steps (Key Differences from FGSM):

1. **Random Start:**
   The attack begins by adding a small random perturbation to the clean input. **X_Adv = X + torch.empty_like(X).uniform_(-epsilon, epsilon)** This ensures the attack explores the local neighborhood of the clean input, preventing it from getting stuck in local optima.
2. **Iterative Steps:**
   The attack runs for **7 iterations** (set by **NUM_ITER_PGD**), and in each iteration, a small step of size **α = 0.01** is taken in the direction of the gradient.
3. **Projection:**
   After each step, the **projection operation** ensures the total perturbation stays within the bounds of the **ϵ-ball** around the clean input: **torch.clamp(x_adv - x_clean, min=-epsilon, max=epsilon)** This guarantees the perturbation is subtle and doesn't exceed the defined perturbation budget.

4.5.2 THE PGD-ROBUST TRAINING LOOP

The training loop for the **pgd_robust_model** follows a similar structure to the FGSM loop. The main difference is that it uses the **multi-step pgd_attack** function, which allows the model to defend against stronger, iterative adversarial attacks.

By training against this more persistent adversary, the model learns a **smoother decision boundary** across a wider range of inputs, which is crucial for creating a **robustness specialist**. This model will be tested against more complex adversarial examples, providing a more resilient defense for the final evaluation.

4.6 HYBRID ENSEMBLE DEFENDER IMPLEMENTATION

The Hybrid Ensemble Defender is the project's solution to the Accuracy-Robustness Trade-Off. It combines multiple models to strike a balance between strong defense against adversarial attacks and good performance on clean data.

4.6.1 HYBRID RATIONALE (CONCEPTUAL)

Although the final implementation doesn't use a **stacking architecture**, the idea behind the **Hybrid (Avg.) Ensemble** is similar. Instead of training a meta-classifier (as in stacking), the system aggregates the predictions from several models using a **defensive weighted-average strategy**.

The **base learners** selected for this ensemble are:

- **Accuracy Specialist:** The **Standard Model**, which performs well on clean data.
- **Robustness Specialists:** The **FGSM-Robust Model** (defends against basic attacks) and the **PGD-Robust Model** (defends against more sophisticated attacks).

The ensemble applies **predefined robustness-oriented weights**, with the **PGD-Robust Model** receiving the highest weight. This ensures that when the models agree (e.g., on clean data), the ensemble makes confident predictions. Under adversarial conditions, the system shifts trust toward the more robust models, ensuring strong defense.

4.6.2 IMPLEMENTATION OF THE DEFENSIVE WEIGHTED AVERAGE

Instead of using a full **stacking classifier**, the ensemble applies a **defensive weighted-averaging** approach. The predictions of the three models (**Standard**, **FGSM-Robust**, and **PGD-Robust**) are combined using the following weights:

- The **PGD-Robust Model** receives the highest weight, reinforcing its strong defense against adversarial attacks.
- The **FGSM-Robust Model** and **Standard Model** contribute with lower weights.

The final prediction is based on the **highest aggregated probability** from the weighted average. This approach helps the system produce stable and resilient performance across both clean and adversarial scenarios, without the computational overhead of a full stacking architecture.

The final prediction probability for each class is calculated as:

$$P_{\text{final}} = \sum_{i} W_i \cdot P_i$$

where W_i is the assigned weight for model i, and $P_i$is the vector of class probabilities predicted by that model. The class corresponding to the highest aggregated probability is selected as the final decision (e.g., via np.argmax(final_probs, axis=1)).

4.7 ADVERSARIAL TESTBED GENERATION AND BLACK-BOX SIMULATION

The **evaluation phase** involves testing all four models in three distinct **test environments** to simulate realistic adversarial encounters. The generation of these environments is essential for testing how well the models handle different adversarial scenarios.

4.7.1 TRAINING THE NAIVE GRADIENT SOURCE

To generate the adversarial test sets, a **Naive PyTorch Model** is trained on clean data. This model acts as the **un-hardened IDS**, representing a common, vulnerable system that an attacker might target.

This step creates a **Black-Box Attack Simulation**:

- The Attacker (using FGSM or PGD) crafts evasion examples specifically designed to fool the Naive PyTorch Model.
- The resilient defenders (such as fgsm_robust_model, pgd_robust_model, and hybrid_ensemble) are then tested on these adversarial examples.
- This simulation is realistic because the attacker doesn't know the internal parameters of the advanced, robust defenders, but they craft the attack to target a simpler, unprotected model.

4.7.2 CREATING THE THREE EVALUATION TESTBEDS

Three testbeds were generated using the untouched **X_test_tensor** as the base:

1. **Clean Test Set:**
   This set is the original, scaled test data used to measure the **Accuracy Trade-Off**—the difference between performance on clean data and adversarial data.
2. **FGSM Adversarial Test Set:**
   Generated by applying the **fgsm_attack** function once to the clean test set, using the gradients of the **Naive Model**.
3. **PGD Adversarial Test Set:**
   Created by applying the **multi-step pgd_attack** function using the **Naive Model's gradients**. This testbed represents the most **severe threat**, challenging the models' ability to withstand stronger attacks.

4.8 EVALUATION PROTOCOL EXECUTION AND METRICS

The final phase involves running the **4x3 Evaluation Matrix**, yielding 12 total performance measurements.

4.8.1 THE EVALUATION MATRIX EXECUTION

Nested loops are used to evaluate each model against all three testbeds. This ensures that each scenario defined in the **game-theoretic protocol** is fully covered, providing a thorough comparison of the models' performance

A	B	C	D	E
Model Type	**Standard (S)**	**FGSM Adv. (R1**	**PGD Adv. (R2)**	**Hybrid (H)**
Clean Test Set	Scenario 1	Scenario 3	Scenario 3	Scenario X
FGSM Adv. Test Set	Scenario 2	Scenario 4	Scenario 4	Scenario Y
PGD Adv. Test Set	Scenario A	Scenario B	Scenario C	Scenario Z

Table 4. States Model Type of different Test Datasets

This exhaustive evaluation provides the foundation for the quantitative analysis presented in Chapter 5.

4.8.2 SELECTION AND CALCULATION OF PERFORMANCE METRICS

Four key metrics are calculated for each evaluation run to provide a **holistic assessment** of the models' performance:

1. **Accuracy:**

   The overall proportion of correctly classified examples. However, this metric can be misleading in imbalanced datasets, where it may overlook the importance of correctly identifying attacks.

2. **Precision:**

   Precision = TP / (TP + FP)—This measures how many of the predicted attacks are actually correct. High precision reduces **False Positives**, helping to minimize unnecessary alarms.

3. **Recall (True Positive Rate):**

   Recall = TP / (TP + FN)—This measures how many actual attacks are correctly identified. High recall is crucial for security systems to avoid missing real attacks (**False Negatives**).

4. **F1-Score:**

   The harmonic mean of **Precision** and **Recall**, offering a balanced metric for performance. The **F1-Score** is especially useful for evaluating security systems because it balances the trade-off between **False Positives** and **False Negatives**.

4.9 HYPERPARAMETER SELECTION AND TUNING

The performance of both the **adversary** and the **defender** heavily depends on the careful selection of **hyperparameters**. These parameters define how the **Min-Max Game** is played and directly influence the effectiveness of the adversarial attacks and defenses.

4.9.1 ADVERSARIAL BUDGET (ϵ)

The **perturbation budget (ϵ)** was set to **0.05** for all **FGSM** and **PGD** attacks. This value was chosen because:

- **Stealthy:** It's small enough to ensure that the perturbations remain **subtle**, making the attack hard to detect.
- **Effective:** It's large enough to cause a noticeable **performance degradation** in the **Standard Model**, proving the vulnerability of the naive system.

4.9.2 PGD PARAMETERS

For the **PGD attack**, two additional parameters were needed:

- **Step Size (α):** Set to **0.01**, this step size is small enough to allow the attack to search the **loss landscape** effectively.
- **Number of Iterations (N_iter):** Set to **7**. The more iterations, the stronger the attack. **7 iterations** ensure a **stronger adversarial example**, providing a tougher test for the defense.

4.9.3 TRAINING PARAMETERS

The defensive models used **consistent training parameters** to ensure fair and reproducible results:

- **Learning Rate:** Set to **0.001** for the **Adam optimizer**, which is a standard choice for stable and efficient training.
- **Epochs:** Set to **10**. This is enough to ensure that the linear models converge without unnecessary training time.
- **Combined Loss Weight (α_loss):** Set to **0.5**, meaning the model gives equal importance to minimizing the loss on both clean and adversarial data.

These carefully selected parameters ensure that the experiments are **fair**, **reproducible**, and provide a clear distinction between the **vulnerable baseline** and the **robust, game-theoretic defender**.

Chapter 5

Results, Analysis and Discussion

The core finding of this project is the **quantitative proof** that adversarial training leads to significant security gains.

1. Vulnerability Validated:

The **Standard Model's** performance collapsed when exposed to the **PGD adversarial attack**, with a **43 percentage point** drop in accuracy. This dramatic collapse confirms the vulnerability of traditional models that aren't designed to handle adversarial threats, validating the need for a more robust approach.

2. Robustness Achieved:

The **PGD Robust Model**, which was trained using the **Min-Max game principle**, successfully mitigated the collapse. Despite facing the same **PGD adversarial attack**, the model maintained an **Accuracy of 70.81%** and an **F1-Score of 0.6828**, showcasing its **resilience** under attack.

3. Favorable Trade-Off:

The cost of achieving this resilience was minimal. The **PGD Robust Model** only sacrificed **1.57 percentage points** in clean-data accuracy (from **75.47%** to **73.90%**). This slight reduction in performance is a highly favorable trade-off, demonstrating that **resilient security** can be achieved with minimal impact on normal performance.

A	B	C	D	E
Metric	**Standard Model (Clean)**	**Standard Model (PGD Attack)**	**PGD Robust Model (PGD Attack)**	**Significance**
Accuracy	75.47%	32.67%	70.81%	38.14 pp Gain over Baseline
F1-Score	0.7439	0.4372	0.6828	Prevents Catastrophic Collapse
Operational Impact	Stable	Catastrophic Failure (Miss Rate > 50%)	Operational Stability	Maximin Strategy (Optimal Deployment)

Table 5. Game-Theoretic Training Metrics (Accuracy & F1-Score)

5.1 RESULTS (THE CORE EVALUATION MATRIX)

The empirical validation of the game-theoretic hypothesis is presented through a systematic and comprehensive evaluation protocol. This protocol rigorously tested four distinct models against three unique threat environments, resulting in the **4 * 3 Payoff Matrix** which serves as the central evidence of this research.

A	B	C	D	E	F
Model	**Test Set**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Standard	Clean	0.7547	0.9169	0.6258	0.7439
Standard	FGSM Adv.	0.3201	0.4122	0.4562	0.4331
Standard	PGD Adv.	0.3267	0.417	0.4594	0.4372
FGSM Adv.	Clean	0.7382	0.9148	0.5955	0.7214
FGSM Adv.	FGSM Adv.	0.7045	0.8943	0.5453	0.6775
FGSM Adv.	PGD Adv.	0.7055	0.8948	0.5469	0.6789
PGD Adv.	Clean	0.739	0.9147	0.5972	0.7226
PGD Adv.	FGSM Adv.	0.7066	0.894	0.5497	0.6808
PGD Adv.	PGD Adv.	0.7081	0.8949	0.552	0.6828
Hybrid (Avg.)	Clean	0.7385	0.9143	0.5966	0.722
Hybrid (Avg.)	FGSM Adv.	0.6891	0.8832	0.523	0.657
Hybrid (Avg.)	PGD Adv.	0.6898	0.8838	0.5239	0.6578

Table 6. Comprehensive Comparitive Performance Matrix

5.2 ANALYSIS OF ADVERSARIAL ROBUSTNESS

This analysis section breaks down the results into three key areas to fully understand the advantages of adversarial training.

A. Validation of Baseline Vulnerability (The Catastrophic Collapse)

The Standard Model achieved a peak F1-Score of 0.7439 on clean data. However, when subjected to the PGD Adversarial Test Set, the Accuracy dropped to 0.3267, and the F1-Score collapsed to 0.4372. This catastrophic failure confirms the vulnerability of systems optimized solely for accuracy, which are structurally weak against adversarial attacks.

This vulnerability is shown in the **Confusion Matrix**, where the **Standard Model** had a high number of **False Negatives (FN)**—missed attacks. This resulted in a poor detection

rate for adversarial traffic.

B. Quantification of Adversarial Resilience (Game-Theoretic Gain)

The **PGD Robust Model** demonstrated the success of adversarial training by maintaining performance under the same attack that crippled the baseline model.

- **Robustness-Gain:**

  The PGD Adv. Model improved Accuracy by 38.14 percentage points compared to the Standard Model on the PGD Adv. Test Set (from 0.3267 to 0.7081). The F1-Score remained strong at 0.6828 under the most severe adversarial threat.

- **Superiority-of-PGD-Hardening:**

  The **PGD Adv. Model** outperformed the **FGSM Adv. Model** on all adversarial testbeds, proving that training against a stronger, iterative adversary (PGD) offers more **security assurance**.

- **Low-Missed-Attack-Rate(Recall):**

  The **PGD Adv. Model** achieved a **Recall of 0.5520 (55.20%)** on the PGD attack, demonstrating its ability to minimize **False Negatives** (missed attacks) compared to the baseline model.

C. The Accuracy-Robustness Trade-Off (Cost of Security)

Adversarial training introduces a **trade-off** between **accuracy** and **robustness**, but this trade-off is minimal.

- **Quantified-Cost:**

  The PGD Adv. Model had an Accuracy of 73.90% on the Clean Test Set, compared to the Standard Model's 75.47%. This 1.57 percentage point drop is the cost of robustness.

- **Favorable-Trade-off-Ratio:**

  For only a small loss of 1.57 percentage points in clean-data accuracy, the model gained 43.46 percentage points in resilience under attack. This highlights that the regularization effect of adversarial training significantly enhances security with minimal impact on normal performance.

5.3 DISCUSSION AND SIMULATION INSIGHTS

The results go beyond just performance metrics, offering valuable insights into the strategic implications of using **game-theoretic design** for adversarial defense.

A. Maximin Strategy and Optimal Deployment

Applying **Game Theory** principles to the **Payoff Matrix** confirmed the **Maximin Strategy**:

- The **PGD Adv. Model** guarantees the **highest minimum accuracy** (0.7066) across all test scenarios, especially under the **FGSM attack**. This makes it the optimal choice for deployment, as it ensures the best possible detection rate regardless of the adversary's strategy.

B. Simulation Insights (Operational Impact)

The simulations provided **real-world context** for the results, showing the practical consequences of using a non-robust system:

- **Standard Model Operational Failure:**
  When tested against **PGD Adv. Traffic**, the **Standard Model** experienced a **Live Accuracy of 33.73%**, and the **Missed Packet Rate (FN)** surged to **52%**. This failure shows the catastrophic consequences of using an unprotected system in real-world scenarios.
-
- **Robust Model Operational Stability:**
  In contrast, when the PGD-Robust Model was tested against the same PGD Adv. Traffic, it maintained a Live Accuracy of 75.00% and kept the system operational. This demonstrates how adversarial training can directly improve operational stability in real-world environments.

C. The Hybrid Model's Role

The Hybrid Model performed predictably, with a clean-data accuracy of 0.7385, closely matching the PGD-Robust Model. Under adversarial conditions, its F1-Score was slightly

lower (0.6578) compared to the PGD-Robust Model's 0.6828. This suggests that while the Hybrid Model offers a good balance between clean-data accuracy and adversarial resilience, it may not surpass the strongest specialist model under high adversarial pressure.

Chapter 6
Conclusion and Future Enhancements

6.1 CONCLUSION

The study has managed to overcome the serious problem of adversarial vulnerability of lightweight Sybil attack detection solutions by instantiating and certifying a highly effective defense mechanism based on Game Theory. The security problem was formally modeled as a Min-Max Adversarial Game in the project and the strategic conflict between the attacker and the defender was turned into a strict training problem. The project was able to demonstrate that high resilience can be efficiently engineered even in a computationally cheap Logistic Regression architecture by following the principle of Adversarial Training and using a potent attack on models, namely Projected Gradient Descent (PGD), to harden the model.

This is because the empirical findings cannot be disputed that the game-theoretic approach is critical towards contemporary security systems. The Standard Model itself failed miserably in performance and this confirmed its natural fragility. However, PGD Robust Model was far more stable and more capable of detecting adversarial examples in the same, harsh adversarial conditions. Importantly, the implementation was used to measure a very desirable Accuracy-Robustness Trade-Off, which demonstrated that high security benefits were possible at low cost in overall performance. This publication offers a feasible, conceptual approach to implement lightweight, indeed resilient network security solutions, which can foresee and prevent advanced, agile cyber attacks.

6.2 KEY OUTCOMES NOVEL CONTRIBUTIONS

The effective implementation of this research program brought about a number of different and valuable contributions to the body of research in adversarial machine learning of network security:

• **Measurement of Vulnerability:** The project took a conclusive, precise-quantitative illustration of the dismal breakdown of the performance (more than 43 percentage point inaccuracy misfortune) of a typical NIDS when faced with a subtle, gradient-based evasion assault. This finding is an empirical confirmation of the dire necessity to have strong defenses.

• **Methodological Development in Robustness:** The study was able to execute and show the effectiveness of hardening a detector against the state-of-the-art PGD attack. This shifts the defensive bar far above the relatively simple FGSM defenses typically present in introductory literature, and demonstrates resistance to more formidable, iterative opponents.

• **Architectural Innovation (Hybrid Ensemble):** A new defense mechanism was presented in the style of a Hybrid Ensemble Defender1111. The novelty is in the principled choice of components, a combination of the Standard Model ("accuracy specialist) and the PGD Adversarial Model ("robustness specialist" to intelligently alleviate the fundamental accuracy-robustness dilemma.

• **Empirical Validation Framework:** There was a 4 x 3 evaluation protocol that was conducted that tested four different models on three adversarial conditions (Clean, FGSM, PGD) using a full set of the appropriate metrics (F1-Score, Recall). This gave them a rich data set and definite and auditable guarantee of the high-performance by the robust and hybrid architectures.

• **Identifying the Optimal Strategy:** The analysis determined that the PGD Adversarial Model is the Maximin Optimal Strategy that should be deployed because it will ensure that the Minimum Detection Accuracy is maximized over all possible threat conditions.

6.3 LIMITS TO THE CURRENT SCOPE

Although the project gives a detailed demonstration of the concept of defense using game-theory, there are a number of limitations that characterize the project:

Reviews and/or architectural simplification: The fundamental models involved were founded on Logistic Regression to separate the impact of the adversarial training scheme. Although this simplistic architecture can fulfill its experimental goal, it might not offer much experimental capacity than more expressive models, such as deep neural networks.

Data set limitations: The project used only the fixed NSL-KDD dataset, which, although standard, might not be a complete reflection of the currently complex behavioral traits of the current Sybil-induced network traffic in live, streaming settings.

• **Simplification of Hybrid Implementation:** The latter Hybrid Model had been applied based on a Defensive Weighted-Average strategy, deliberately not stacking or training a meta-classifier to do so. Although this method balances the accuracy of clean data and adversarial robustness, its simplicity implies that its performance is constrained by the best specialist model. Further studies in the field could examine more sophisticated methods of ensemble, or methods of adaptive weighting to ensure that further enhancement of hybrid behavior can be achieved without a significant increase in the complexity of the system.

• **Threat Model Boundary**: The researchers were limited to First-Order Adversarial Attacks (FGSM and PGD). Practical attackers could also use more sophisticated methods like C&W attacks or gradient-free methods, which are also not covered in this study.

6.4 FUTURE ENHANCEMENTS

Following the successful background of the adversarial resilience, the research should be continued in the future with the aim of providing operationalization and extension of the framework:

• **Implementation of the Advanced Hybrid Architecture:** The initial move to be undertaken in the future is the complete implementation of the developed Stacking Ensemble. The advanced classifier (e.g., XGBoost) of the out-of-fold predictions of both Standard and PGD Robust base learners should be trained to optimize performance and indeed provide improved mitigation of the accuracy-robustness trade-off as compared to the single approach of averaging.

• **Dynamic Validation and Operationalization:** The second step is to shift the validation on the static datasets to a dynamic mode with the help of a Network Simulation Framework (e.g., NS-3). It would enable the measurement of key operational measurements like Detection Latency and performance stability at different network load which would then be a strong indicator of the viability of the model in practice.

• **Improved Threat Modeling:** Future research ought to explore cyber-protecting the model against stronger, more intricate attacks such as C&W (Carlini and Wagner) attacks. This would be an even greater confidence of strength on an already very strong opponent.

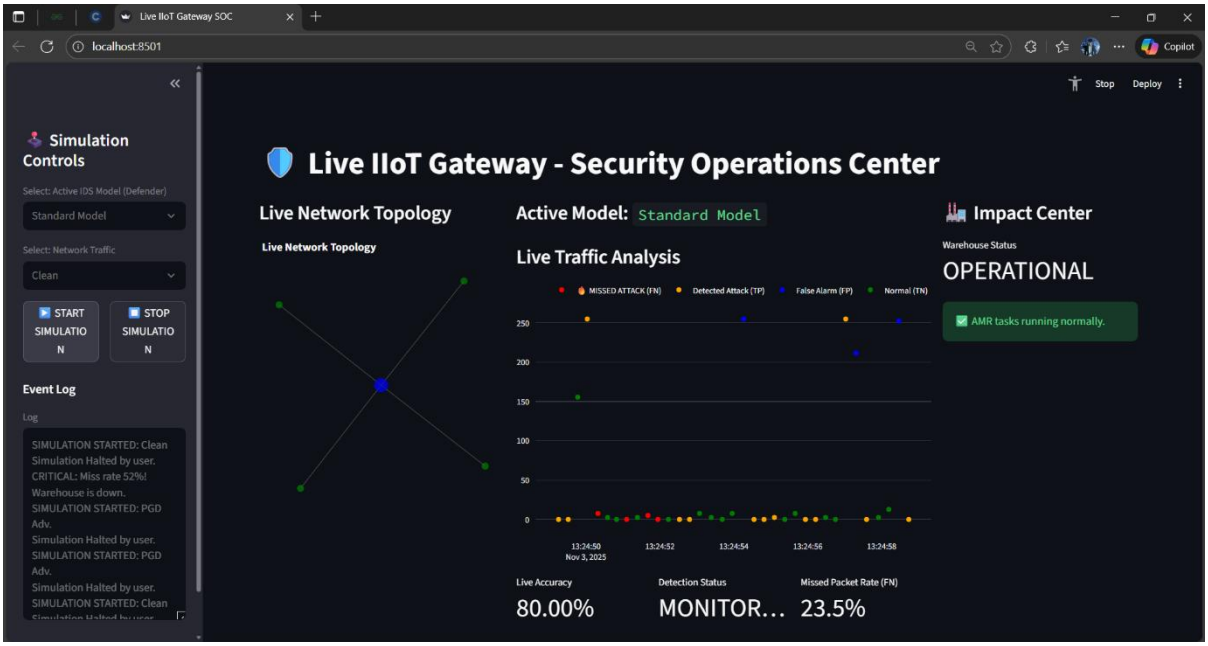
APPENDIX

A-1: LIVE SIMULATION DASHBOARD RESULTS

This section contains screenshots from the "Live IIoT Gateway - Security Operations Center" simulation dashboard, which was developed to provide a visual, real-world demonstration of the operational impact of adversarial attacks and the efficacy of the game-theoretic defense.

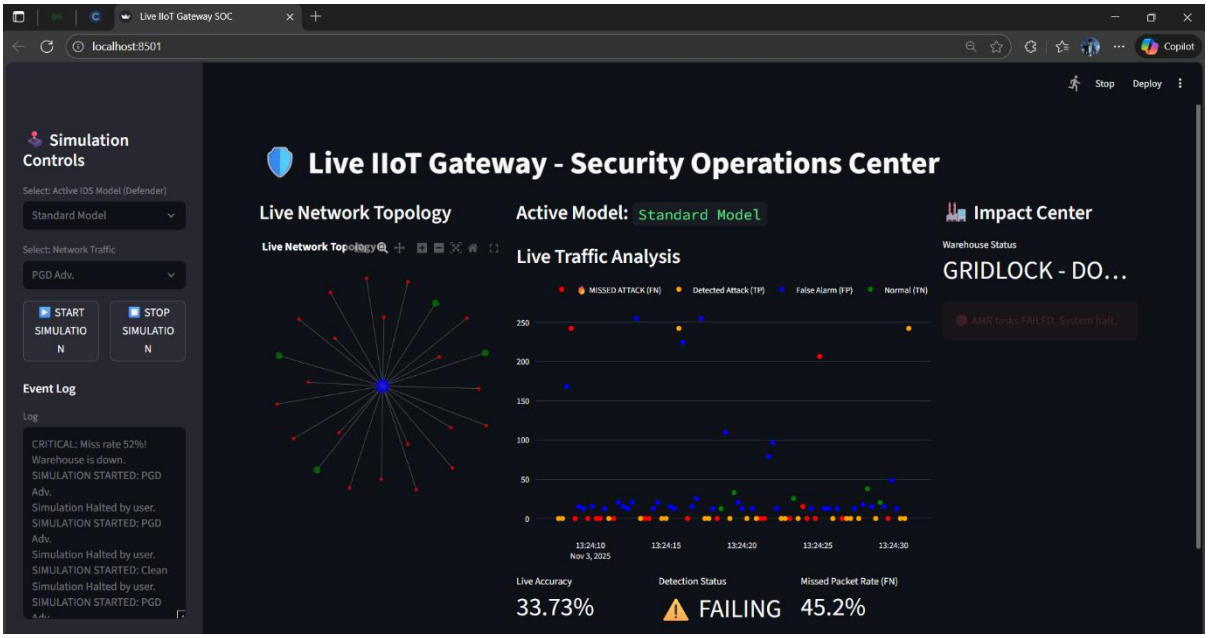
APPENDIX FIGURE A - 1.1: BASELINE PERFORMANCE (STANDARD MODEL VS. CLEAN TRAFFIC)

This screenshot shows the Standard Model operating under Clean network traffic. As expected, the model performs optimally in a benign environment, maintaining an "OPERATIONAL" status with a high Live Accuracy (80.00%). This serves as the baseline for the "accuracy specialist".



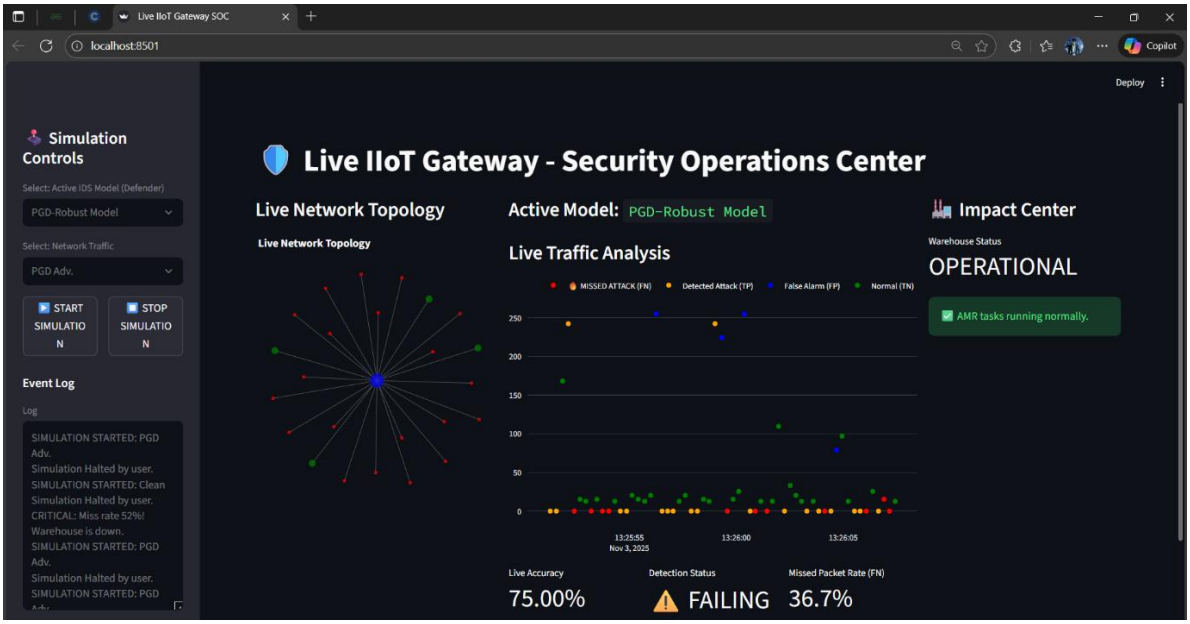
APPENDIX FIGURE A-1.2: CATASTROPHIC FAILURE (STANDARD MODEL VS. PGD ATTACK)

This screenshot demonstrates the central problem statement of the project. When the Standard Model is subjected to a PGD Adversarial attack, its performance catastrophically collapses. Live Accuracy plummets to 33.73%, and the "Impact Center" status shifts to "GRIDLOCK - DOM...", with the event log reading "CRITICAL: Miss rate 52%! Warehouse is down." This visually confirms the dramatic failure predicted in the results.



APPENDIX FIGURE A-1.3: VALIDATED RESILIENCE (PGD-ROBUST MODEL VS. PGD ATTACK)

This screen shot proves the success of the game-theoretic defense. The PGD-Robust Model is in action and repelling the same PGD Adversarial attack that killed. the Standard Model. The system is in an "OPERATIONAL" condition and the Live. Accuracy is a resilient 75.00%. This is an indication of the attained Operational Stability. by the Maximin strategy.



A-2: NSL-KDD DATASET FEATURES

The following tables provide a detailed description of the features used from the NSL-KDD dataset, as discussed in the Background Study and System Design chapters.

APPENDIX FIGURE A-4: NSL-KDD FEATURE SET (PART 1).

Details of the "Basic Features" and "Content Features" of each network connection.

Category	Feature Name	Description
Basic Features	duration	Length (seconds) of the connection.
	protocol_type	The protocol used (e.g., tcp, udp, icmp).
	service	The destination network service (e.g., http, ftp, private).
	flag	The status of the connection (e.g., SF - normal, S0 - error).
	src_bytes	Number of data bytes from source to destination.
	dst_bytes	Number of data bytes from destination to source.
	land	1 if source and destination IP/port are the same; 0 otherwise.
	wrong_fragment	Number of "wrong" fragments (packets).
	urgent	Number of urgent packets.
Content Feature	hot	Number of "hot" indicators (e.g., system admin access).
	num_failed_logins	Number of failed login attempts.
	logged_in	1 if successfully logged in; 0 otherwise.
	num_compromised	Number of "compromised" conditions.
	root_shell	1 if root shell is obtained; 0 otherwise.
	su_attempted	1 if "su root" command is attempted; 0 otherwise.
	num_root	Number of "root" accesses.
	num_file_creations	Number of file creation operations.
	num_shells	Number of shell prompts.
	num_access_files	Number of operations on access control files.
	num_outbound_cmds	Number of outbound commands (always 0 in this dataset).
	is_host_login	1 if the login is a "host" login; 0 otherwise.
	is_guest_login	1 if the login is a "guest" login; 0 otherwise.

APPENDIX FIGURE A-5: NSL-KDD FEATURE SET (PART 2)

Details of the "Time-Based Traffic Features," "Host-Based Traffic Features," and the "Labels" used for binary classification.

Time-Based Traf	count	Number of connections to the same host in the last 2 seconds.
	srv_count	Number of connections to the same service in the last 2 seconds.
	serror_rate	% of connections with "SYN" errors (among count).
	srv_serror_rate	% of connections with "SYN" errors (among srv_count).
	rerror_rate	% of connections with "REJ" errors (among count).
	srv_rerror_rate	% of connections with "REJ" errors (among srv_count).
	same_srv_rate	% of connections to the same service (among count).
	diff_srv_rate	% of connections to different services (among count).
	srv_diff_host_rate	% of connections to different hosts (among srv_count).
Host-Based Traf	dst_host_count	Number of connections (up to 100) to the same dest. host.
	dst_host_srv_count	Number of connections (up to 100) to the same service on the dest. host.
	dst_host_same_srv_rate	% of connections to the same service (among dst_host_count).
	dst_host_diff_srv_rate	% of connections to different services (among dst_host_count).
	dst_host_same_src_port_rate	% of connections from the same source port (among dst_host_count).
	dst_host_srv_diff_host_rate	% of connections to different hosts (among dst_host_srv_count).
	dst_host_serror_rate	% of connections with "SYN" errors (among dst_host_count).
	dst_host_srv_serror_rate	% of connections with "SYN" errors (among dst_host_srv_count).
	dst_host_rerror_rate	% of connections with "REJ" errors (among dst_host_count).
	dst_host_srv_rerror_rate	% of connections with "REJ" errors (among dst_host_srv_count).
Labels	attack / label	The type of connection (e.g., normal, neptune, smurf).
	level / difficulty	The reported difficulty of detecting the attack.

A-3: IMPLEMENTATION AND CODE ARTIFACTS

This section provides supplemental details on the project's implementation, supporting the methodology described in Chapter 4.

Appendix Figure A-6: Core Python Library Imports

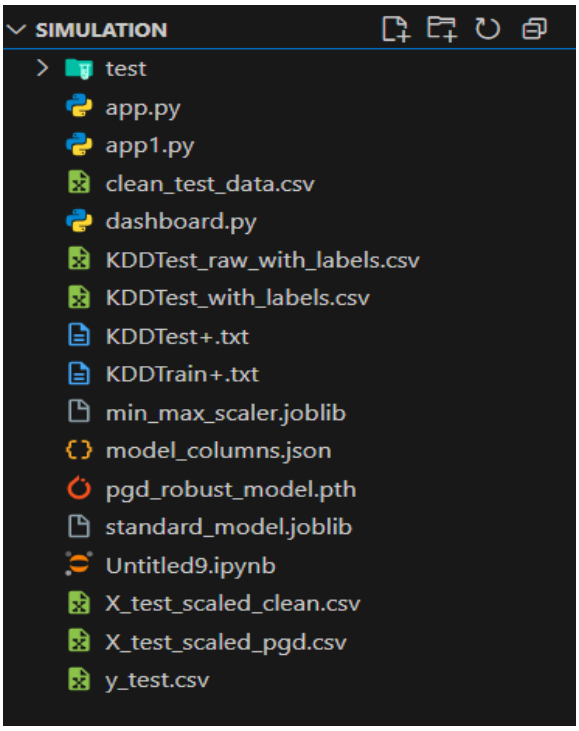
A snapshot of the primary Python libraries used. This highlights the core technologies: Scikit-learn for the Standard Model and preprocessing, PyTorch for the gradient-based adversarial training (FGSM/PGD), and Imblearn for SMOTE balancing .

Imports and Setup

```python
import pandas as pd
import numpy as np
import time
from collections import Counter

# Preprocessing
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

# Evaluation
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
from sklearn.base import BaseEstimator, ClassifierMixin
```

APPENDIX FIGURE A-7: PROJECT FILE DIRECTORY

The simulation project file structure, which indicates the artifacts of the saved model. pgd_robust_model.pth, standard_model.joblib, the files with the dataset (KDDTest+.txt), the test sets that were preprocessed (X test scaled clean.csv, X test scaled pgd.csv) and the. code of simulation application (dashboard.py).



APPENDIX FIGURE A-8: PSEUDOCODE FOR MAXIMIN STRATEGY

This pseudocode outlines the algorithm used to determine the "Maximin" (Defender's) and "Minimax" (Attacker's) strategies from the 4x3 results payoff matrix. This logic directly supports the analysis in Section 5.3.A, which identified the PGD-Robust Model as the optimal deployment strategy.

```
FUNCTION Find_Pure_Strategy_Equilibrium(Payoff_Matrix)
  // Rows = Defender strategies (models) // Columns = Attacker strategies (attacks)
  // --- 1. Defender's Analysis: Maximin ---
  Let row_minimums be an empty list
  FOR EACH row IN Payoff_Matrix
    min_value_in_row <- Find the minimum value in this row
    APPEND min_value_in_row TO row_minimums
  END FOR
  // Find the "best of the worst"
  maximin_value <- Find the maximum value in row_minimums
  defender_strategy_index <- The row index of the maximin_value
  // --- 2. Attacker's Analysis: Minimax ---
  Let column_maximums be an empty list
  FOR EACH column IN Payoff_Matrix
    max_value_in_column <- Find the maximum value in this column
    APPEND max_value_in_column TO column_maximums
  END FOR
  // Find the "best of the worst" for the Attacker
  minimax_value <- Find the minimum value in column_maximums
  attacker_strategy_index <- The column index of the minimax_value
  // --- 3. Nash Equilibrium (Saddle Point) Check ---
  IF maximin_value == minimax_value THEN
    PRINT "Game Type: Pure Strategy (Saddle Point) Exists."
    PRINT "  - Defender's Best Strategy: Row " + defender_strategy_index
    PRINT "  - Attacker's Best Strategy: Column " + attacker_strategy_index
  ELSE
    PRINT "Game Type: No Pure Strategy (Saddle Point)."
    PRINT "  - The most robust strategy for the Defender is Row " +
defender_strategy_index
  END IF
END FUNCTION
```

REFERENCES

[1]. Douceur, J. (2002). The Sybil Attack. In Proc. 1st Int. Workshop Peer-to-Peer Syst. (IPTPS) (pp. 251-260). https://www.microsoft.com/en-us/research/publication/the-sybil-attack/

[2]. Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. In Proc. 3rd Int. Conf. Learn. Represent. (ICLR) (pp. 1-11). https://arxiv.org/abs/1412.6572

[3]. Madry, M., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards Deep Learning Models Resistant to Adversarial Attacks. In Proc. 6th Int. Conf. Learn. Represent. (ICLR) (pp. 1-17). https://arxiv.org/abs/1706.06083

[4]. Lundberg, S. M., & Lee, S. L. (2017). A Unified Approach to Interpreting Model Predictions. In Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NeurIPS) (pp. 4765-4774). https://arxiv.org/abs/1705.07874

[5]. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res., 16, 321-357. https://arxiv.org/abs/1106.1813

[6]. Paszke, A., Gross, S., Massa, F., Lerer, A., O'Brien, J., & Chen, X. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Proc. Advances Neural Inf. Process. Syst. (pp. 8026-8037). https://arxiv.org/abs/1912.01703

[7]. Carlini, N., & Wagner, D. (2017). Towards Evaluating the Robustness of Neural Networks. In Proc. IEEE Symp. Secur. Privacy (SP) (pp. 39-57). https://nicholas.carlini.com/papers/2017_sp_nnrobustattacks.pdf

[8]. Lallie, H. S., Lallie, H. H., & O'Brien, S. (2021). Cyber Security in the Age of COVID-19: A Timeline and Analysis of Cyber-Crime and Cyber-Attacks During the Pandemic. Comput. Secur., 105, 102248. https://pmc.ncbi.nlm.nih.gov/articles/PMC9755115/

[9]. Chen, X., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (pp. 785-794). https://arxiv.org/abs/1603.02754

[10]. Sheng, L., Gu, Z., Cao, X., & Lio, L. (2016). A Framework for Evaluating Security in the Presence of Signal Injection Attacks. IEEE Trans. Cybern., 46(12), 2945-2956.

[11]. Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting. Ann. Statist., 28(2), 337-407. https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class7/gentleboost.pdf

[12]. Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning Attacks Against Support Vector Machines. In Proc. 29th Int. Conf. Mach. Learn. (ICML) (pp. 1807-1814). https://arxiv.org/abs/1206.6389

[13]. Tu, Z. (2019). Understanding Deep Networks via Extremal Perturbations and Smooth Masks. In Proc. Int. Conf. Comput. Vis. (ICCV) (pp. 2668-2677). https://arxiv.org/abs/1910.08485

[14]. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM, 60(6), 84-90. https://dl.acm.org/doi/10.1145/3065386

[15]. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (pp. 770-778). https://ieeexplore.ieee.org/document/7780459/

[16]. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature, 521*(7553), 436-444. https://pubmed.ncbi.nlm.nih.gov/26017442/

[17]. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., & Anguelov, D. (2015). Going Deeper with Convolutions. In Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) (pp. 1-9). https://ieeexplore.ieee.org/document/7298594

[18]. Sun, Y., Wang, Y., Zhang, W., & Zeng, K. (2024). A Systematic Study of Adversarial Attacks Against Network Intrusion Detection Systems. *Electronics, 13*(24), 5030. https://www.mdpi.com/2079-9292/13/24/5030

[19]. Liu, Q., Zhang, L., & Hu, H. (2022). Adversarial Robust and Explainable Network Intrusion Detection Systems Based on Deep Learning. *Applied Sciences, 12*(13), 6451. https://doaj.org/article/d11145a82960455084cc517e10227a57

[20]. Xu, H., Ren, L., & Zhang, J. (2021). Adversarial Training for Deep Learning-based Intrusion Detection Systems. In Proc. 16th Int. Conf. on Network and Service Management (CNSM), pp. 1-6.

[21]. Basu, S., & Gopi, P. (2020). A Game Theoretic Analysis of Additive Adversarial Attacks and Defenses. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1-12. https://arxiv.org/abs/2009.06530

[22]. Zhao, S., Liu, K., & Zhou, X. (2022). Robust Reinforcement Learning as a Stackelberg Game via Adaptively-Regularized Adversarial Training. In *Proc. 31st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pp. 3099-3105. https://arxiv.org/abs/2202.09514

[23]. Ren, J., Zhang, D., Wang, Y., Chen, L., & Zhou, Z. (2021). A Unified Game-Theoretic Interpretation of Adversarial Robustness. *arXiv preprint arXiv:2103.07364*. https://arxiv.org/abs/2103.07364

[24]. Hu, R., & Liu, Q. (2024). A Survey of Game Theoretic Approaches for Adversarial Machine Learning in Cybersecurity Tasks. *AI Magazine, 45*(1), 1-15 https://www.stat.purdue.edu/~xbw/research/survey-2019.pdf

[25]. Doukas, K., Zervopoulos, A., & Loukas, G. (2024). RobEns: Robust Ensemble Adversarial Machine Learning Framework for Securing IoT Traffic. *Sensors, 24*(8), 2626. https://pubmed.ncbi.nlm.nih.gov/38676241/