Research-Grade Implementation Plan: Zero-Trust AI Defense

This document details the final, high-rigor implementation of the AI-driven Zero-Trust policy engine, upgraded for adversarial robustness and security engineering best practices.

□ Final System Goals

- **Quantifiable Robustness**: Measure evasion rates under adaptive black-box and white-box threats.
- **Defense-in-Depth**: Implement layered security (Supervised + Unsupervised + Uncertainty) to "diffuse" decision boundaries.
- **Academic Rigor**: Eliminate data leakage and implement realistic feature constraints for adversarial examples.
- **Security Observability**: Simulate SOC/SIEM logging for adversarial detection events.

□ Proposed Architecture

1. Training Phase (High Rigor)

[MODIFY] train_model.py

- ✅**Isolation Forest (IF)**: Trained strictly on benign data (`y == 0`) to model "normalcy."
- ✅**Anti-Leakage**: Exports separate `train_set.csv` (for surrogate models) and `test_set.csv` (for evaluation).
- ✅**Feature Bounds**: Saves statistical bounds for ART feature clipping to ensure realistic inputs.

2. Adversarial Evaluation Framework

[MODIFY] adversarial_evaluation.py

- ✅**Black-Box (HopSkipJump)**: Increased budget to **50 iterations** for rigorous stress testing.
- ✅**White-Box (FGM Transfer)**: Trains a separate surrogate NN on training data only (Preventing leakage).
- ✅**Metrics Engine**: Computes Evasion Rate, Robust Accuracy, and Perturbation Norms (L2/Linf).
- ✅**Clipping**: Enforces telemetry-valid inputs during sample generation.

3. Security Engineering Layer

[NEW] threat_model.md

- Formally defines attacker capabilities, knowledge, and system assumptions.

[NEW] security_logger.py

- Simulates real-time security alerts when the ensemble layers detect a defensive override.

✅Final Research Summary Table (Example Results)

Metric (Evasion Rate)	Baseline (RF)	Defended (Ensemble)	Gain
Black-Box (HSJ)	~66%	**~4%**	**94% Improvement**
White-Box (FGM)	~25%	**~16%**	**36% Improvement**

□ Verification Plan

Core Integrity

- `python src/detection/train_model.py`: Ensure all models, bounds, and split datasets are saved.
- `python src/adversarial/adversarial_evaluation.py`: Verify robustness report generation.

Security Logic

- `python src/adversarial/security_logger.py`: Verify that "ADV_EVASION_DETECTED" alerts are logged when anomaly layers override the supervised decision.