Matthew Lewis

Network Intrusion Detection System Simulator

Project Introduction:

In an era where cyber threats are evolving rapidly, understanding and testing intrusion detection systems (IDS) is crucial for building resilient networks. This project is my attempt at creating a self-contained simulator that generates synthetic network traffic, detects anomalies through hybrid rule-based and machine learning methods, and provides real-time monitoring via a web dashboard. Rather than relying on live data, which can be unpredictable and risky, this simulator allows for controlled experimentation in a safe environment, making it ideal for testing detection strategies, iterating on rules, and demonstrating cybersecurity concepts.

The core focus is on simulating realistic network intrusions such as oversized payloads or unusual port usage to mimic common attack vectors, while enabling quick analysis and visualization. This tool serves as a practical bridge between theoretical cybersecurity knowledge and hands-on application, particularly in defense or enterprise settings where rapid prototyping of IDS behaviors is valuable.

Solution and Core Concept:

To achieve effective simulation, the project integrates a full pipeline: from packet generation to detection, alerting, and user-friendly visualization. By generating a mix of normal (80%) and malicious (20%) traffic, it creates a battlespace-like environment for testing IDS logic without external dependencies. Malicious packets are crafted with anomalies like large payloads (1000–48000 bytes) or invalid ports to trigger detections, reflecting real-world threats such as data exfiltration or port scanning.

The detection layer combines rule-based thresholds (e.g., payload >1000 bytes, high packet rates) for interpretable, fast responses with an unsupervised machine learning model (Isolation Forest) for spotting complex patterns. Alerts are generated with context, and results are exposed through APIs and a dynamic dashboard, allowing for iteration on parameters like thresholds or features without rebuilding the system.

This centralized yet modular approach reduces complexity while preserving key causal relationships in intrusion detection, making it a reasoning tool for cybersecurity experimentation rather than a production IDS.

Project Goals:

- Create a synthetic network traffic generator

- Accurately simulate intrusion detection via rules and ML
- Build an alerting system with contextual notifications
- Develop a web dashboard for real-time monitoring and visualization
- Enable easy iteration through CLI, API, and configurable parameters
- Ensure deployability with Docker for consistent environments

Scope Clarification and Non-Goals:

This project prioritizes simulation speed and interpretability over full-scale production features. It does not handle real network interfaces or encrypted traffic analysis, focusing instead on abstracted packet data to keep it lightweight and educational. These choices allow it to function as a design tool for testing hypotheses in cybersecurity, not a deployable enterprise IDS.

Tech Stack:

- Python (core language for all components)
- Scikit-learn (for Isolation Forest ML detection)
- Pandas (data parsing and manipulation)
- Matplotlib (anomaly visualization)
- FastAPI and Uvicorn (web server and API)
- Jinja2 (HTML templating)
- Pytest (unit testing)
- Docker (containerization)

Implementation:

The simulator is structured as a modular pipeline, with key files handling specific roles:

1. Packet Generation (ids_simulator.py): Creates CSV logs with timestamped packets, including source/destination IPs, protocols, ports, and malicious flags.
2. Log Parsing and Detection (parse_logs.py): Loads data into DataFrames, applies time-based aggregations, runs rule-based and ML detections, and generates alerts.
3. Plotting (plot_utils.py): Produces time-series anomaly plots using Matplotlib's non-interactive backend for server compatibility.
4. Web Server and Dashboard (app.py, templates/dashboard.html): Serves APIs for logs, alerts, and plot generation; renders an interactive UI with auto-refreshing tables and buttons for on-demand updates.

Usage is flexible: Run via CLI for quick tests, APIs for integration, or the full dashboard for monitoring. Docker ensures easy deployment, with CI/CD via GitHub Actions for reliability.

This project demonstrates my ability to build end-to-end cybersecurity tools, blending data science, web development, and simulation to address practical challenges in network defense.