Network Security Analytics

ML-Powered Intrusion Detection System

Efe Sirin - S4808746 & Amr Abdou - S4678753

University of Groningen

October 2025

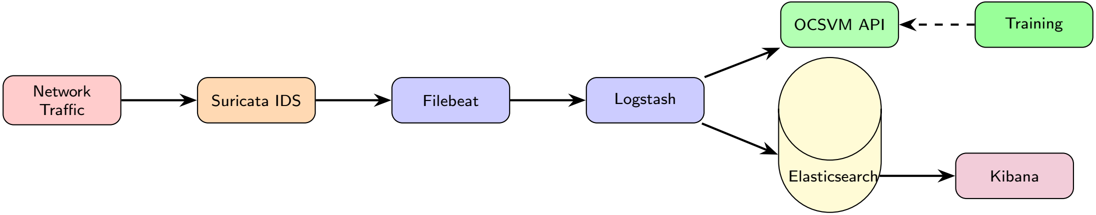
Outline

Introduction

Project Overview

- **Goal:** Complete end-to-end network intrusion detection system
- **Approach:** ML-powered anomaly detection
- **Dataset:** CICIDS2017 - Comprehensive network traffic dataset
- **Architecture:** Full ELK Stack + Suricata + ML API

Complete System Architecture



Pipeline: Network $\rightarrow$ Suricata $\rightarrow$ Filebeat $\rightarrow$ Logstash $\rightarrow$ {Elasticsearch/Kibana, ML API}

Dataset

Dataset: CICIDS2017

Dataset Characteristics

8 CSV files covering different days and attack types

File	Content
Monday-WorkingHours	Benign traffic
Tuesday-WorkingHours	FTP-Patator, SSH-Patator
Wednesday-workingHours	DoS attacks, Heartbleed
Thursday-Morning	Web attacks (Brute Force, XSS, SQL Injection)
Thursday-Afternoon	Infiltration
Friday-Morning	Botnet (Ares)
Friday-Afternoon (PortScan)	Port Scanning
Friday-Afternoon (DDoS)	DDoS attacks

Features: 78 network flow features (duration, packet counts, byte counts, etc.)

Exploratory Analysis of the Dataset

Key Exploration Steps:

1. Load and merge all CSV files
2. Analyze class distribution (benign vs. attacks)
3. Identify missing values and outliers

Insight

The dataset is highly imbalanced – perfect for OCSVM approach!

Model Training & Evaluation

OCSVM with CICIDS2017[1]

Core Principle

Train on benign traffic, classify deviations as anomalies (potential attacks)

Advantages:

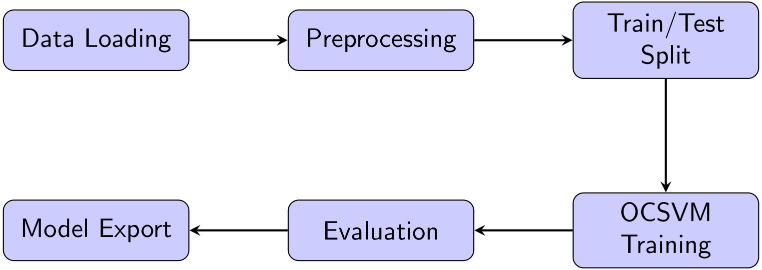
- Inherent data class imbalance handling
- Detects novel/unknown attacks
- No need for labeled attack data

Use Case:

- Zero-day attack detection

[1] Xu, Z. & Liu, Y. (2025). Robust Anomaly Detection in Network Traffic: Evaluating Machine Learning Models on CICIDS2017. arXiv preprint.

Training Pipeline Architecture



Data Preprocessing Steps

1. **Load Data:** Merge 8 CSV files (approx. 2.8M records)
2. **Label Processing:**
   - Binary classification: BENIGN (1) vs. ATTACK (-1)
   - Filter benign traffic for training
3. **Feature Cleaning:**
   - Remove infinite values
   - Handle missing data
   - Remove constant features
4. **Normalization:**
   - StandardScaler for feature scaling
   - Save scaler for deployment
5. **Train/Test Split:**
   - 80% benign traffic for training
   - 20% benign + all attacks for testing

Training Results

Model Summary (Sept 28, 2025)

Training Configuration:

- Mode: Full dataset
- Random Seed: 42 (reproducibility)
- Benign Train Ratio: 80%
- Cache Size: 22 GB (optimized for performance)

Metric	Score
Accuracy	0.5798 (57.98%)
Precision	0.6381 (63.81%)
Recall	0.5484 (54.84%)
F1-Score	0.5898 (58.98%)

Performance Analysis

Strengths:

- Good precision (63.81%)
  - When flagged as attack, likely correct
  - Low false positive rate
- Balanced F1-score (58.98%)

Considerations:

- Moderate recall (54.84%)
  - Some attacks may be missed
  - Trade-off with false positives
- Room for optimization

Note on OCSVM Performance

OCSVM is designed for anomaly detection, not perfect classification. The model prioritizes detecting abnormal patterns while minimizing false alarms.

Training Configuration

OCSVM Hyperparameters

```
# Model hyperparameters
kernel='rbf'           # Radial Basis Function
gamma='scale'          # Auto-computed: 1/(n_features * X.var())
nu=0.05               # 5% anomaly tolerance
max_iter=1000         # Maximum iterations
```

Training Configuration

```
# Data split
train_ratio=0.8       # 80% benign for training
random_state=42       # Reproducibility
cache_size=22000      # 22GB cache for performance
```

OCSVM Model Configuration

Hyperparameters

Kernel: RBF (Radial Basis Function)

- Captures non-linear patterns
- Formula: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$

Nu (ν): 0.05

- Upper bound on training errors
- Lower bound on support vectors
- 5% anomaly tolerance

Gamma: 'scale' (auto-computed)

- $\gamma = \frac{1}{n\_features \times X.var()}$
- Controls decision boundary smoothness

Max Iterations: 1000

Network Monitoring Infrastructure

Technology Stack

Data Collection & Processing:

- **Suricata:** IDS/IPS engine
- **Filebeat:** Log shipper
- **Logstash:** Log processor
- **Elasticsearch:** Search & analytics
- **Kibana:** Visualization

Machine Learning:

- **Python:** Core language
- **scikit-learn:** OCSVM model
- **FastAPI:** REST API
- **Docker:** Containerization
- **DVC:** Data versioning

Container Orchestration

All components deployed via Docker Compose for easy setup and scaling

Suricata IDS - Network Traffic Analysis

What is Suricata?

Open-source network IDS/IPS and network security monitoring engine

Key Capabilities:

- **Real-time packet inspection:** Deep packet analysis
- **Protocol detection:** HTTP, TLS, DNS, and more
- **EVE JSON output:** Structured event logging
- **PCAP processing:** Offline traffic analysis

Our Configuration:

- Flow tracking
- Protocol parsing
- Alert generation
- JSON event logging

Output Events:

- Alerts
- Flow records
- HTTP logs
- TLS/DNS metadata

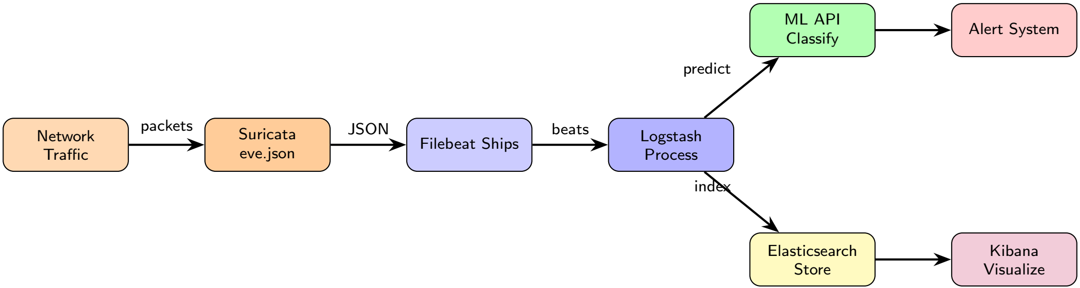
ELK Stack - Log Management Pipeline

Elastic Stack Components

Industry-standard log collection, processing, and visualization

1. **Filebeat (Log Shipper):** Monitors Suricata logs, lightweight forwarding agent with pre-configured module
2. **Logstash (Processing):** Receives, parses, enriches, and transforms events; forwards to Elasticsearch and ML API
3. **Elasticsearch (Storage):** Indexes security events with fast full-text search and time-series optimization
4. **Kibana (Visualization):** Interactive dashboards for real-time monitoring and alert visualization

Data Flow Through the Pipeline



Conclusion

Project Achievements

1. **Complete IDS System:**
   - End-to-end network monitoring pipeline
   - Real-time threat detection
   - Production-ready containerized deployment
2. **ML-Powered Detection:**
   - OCSVM model trained on 2.8M flows
   - 63.81% precision, 58.98% F1-score
   - RESTful API for real-time inference
3. **Operational Stack:**
   - ELK stack for log management
   - Suricata for network analysis
   - Docker Compose orchestration

Thank You!

Questions?

Project: Network Security Analytics
Complete ML-Powered IDS System

Suricata → ELK Stack → OCSVM API → Real-time Detection