DPDK-Suricata-Kafka-ML Complete Pipeline

Architecture Overview

DPDK-Suricata-Kafka-ML Complete Pipeline

Architecture Overview

```
External Traffic (tcpreplay/PCAP)
    ↓ (via Ethernet)
DPDK Packet Capture (bound interface)
    ↓ (zero-copy)
Suricata IDS (DPDK mode)
    ↓ (eve-kafka output: flows + alerts)
Kafka Broker
    ↓ (consume ALL events)
ML Inference Engine (Flow-Based)
    ↓ (CICIDS2017 feature extraction + predictions for EVERY flow)
Enhanced Alert Stream
    ↓
Alert Dashboard / Database
```

Key Features

✨ **Flow-Based ML Inference**: Process **ALL network flows**, not just signature alerts
🧠 **CICIDS2017 Feature Extraction**: 65-feature extraction from Suricata flow events
🔍 **Dual Detection**: Combines Suricata signature detection + ML anomaly detection
⚡ **Real-Time Processing**: Streaming architecture with Kafka message bus
📊 **Threat Scoring**: Combined threat scores from multiple detection sources

PROF

Components

1. DPDK Packet Ingestion

- Binds network interface to DPDK driver (vfio-pci/uio_pci_generic)
- Zero-copy packet capture
- High-performance packet processing

2. Suricata IDS (Enhanced)

- Runs in DPDK mode
- **NEW**: Logs ALL network flows (not just alerts)
- Processes packets with signature-based detection
- Outputs flows + alerts to Kafka via eve-kafka plugin

3. Kafka Streaming

- Acts as message broker
- Buffers flows and alerts between Suricata and ML
- Enables scalable processing
- Topics: `suricata-alerts` (input), `ml-predictions` (output)

4. ML Inference Engine (Enhanced)

- **NEW**: Consumes ALL flow events from Kafka (not just alerts)
- Extracts 65 CICIDS2017 features from every flow
- Performs real-time ML inference on all traffic
- Correlates ML predictions with Suricata alerts
- Generates enhanced alerts with combined threat scores
- Supports Random Forest and LightGBM models

Directory Structure

```
dpdk_suricata_ml_pipeline/
├── README.md                    # This file
├── SETUP_GUIDE.md               # Detailed setup instructions
├── config/                      # Configuration files
│   ├── suricata-dpdk.yaml       # Suricata config with DPDK & Kafka
│   ├── kafka-config.properties  # Kafka settings
│   └── pipeline.conf            # Pipeline configuration
├── scripts/                     # Management scripts
│   ├── 01_bind_interface.sh     # Bind NIC to DPDK
│   ├── 02_setup_kafka.sh        # Install/configure Kafka
│   ├── 03_start_suricata.sh     # Start Suricata in DPDK mode (with flow
logging)
│   ├── 04_start_ml_consumer.sh # Start ML inference
│   ├── 05_replay_traffic.sh     # Replay PCAP files
│   ├── stop_all.sh              # Stop all services
│   ├── status_check.sh          # Check pipeline status
│   └── unbind_interface.sh      # Restore network interface
├── src/                         # Python source code
│   ├── ml_kafka_consumer.py     # ML inference consumer (flow-based)
│   ├── feature_extractor.py     # CICIDS2017 65-feature extraction
│   ├── model_loader.py          # ML model loading (RF, LightGBM)
│   └── alert_processor.py       # Alert correlation & threat scoring
├── pcap_samples/                # Sample PCAP files for testing
│   └── README.md
└── logs/                        # Pipeline logs
    ├── dpdk/
    ├── suricata/
    ├── kafka/
    └── ml/
```

ML Inference Features

Feature Extraction (CICIDS2017 Compatible)

The pipeline extracts **65 network flow features** from Suricata events:

Flow Statistics: Duration, packet counts, byte counts
Packet Length Stats: Min, max, mean, std (forward & backward)
Inter-Arrival Time (IAT): Mean, std, min, max (flow, fwd, bwd)
TCP Flags: FIN, SYN, RST, PSH, ACK, URG, ECE counts
Header Lengths: Forward & backward header sizes
Packet Rates: Packets/sec, bytes/sec (overall, fwd, bwd)
Protocol Features: Protocol type, port numbers
Derived Features: Down/up ratio, avg segment sizes, active/idle times

Supported ML Models

- **Random Forest** (scikit-learn): `random_forest_model_2017.joblib`
- **LightGBM**: `lgb_model_2018.joblib`
- Models located in: `../ML Models/`

Alert Processing

- Correlates Suricata signature alerts with ML predictions
- Calculates combined threat scores (0-1 scale)
- Assigns threat levels: BENIGN, LOW, MEDIUM, HIGH, CRITICAL
- Outputs enhanced alerts to Kafka topic

Quick Start

Prerequisites

- DPDK installed (run `install_dpdk_suricata.sh`)
- Suricata installed with DPDK support
- Kafka installed and running
- Python virtual environment activated
- ML models in `../ML Models/` directory

PROF

Step 1: Configure Interface

Edit `config/pipeline.conf` and set your network interface:

```
NETWORK_INTERFACE="eth0"  # Change to your interface
```

Step 2: Bind Interface to DPDK

```
cd dpdk_suricata_ml_pipeline
sudo ./scripts/01_bind_interface.sh
```

Step 3: Start Kafka

```
./scripts/02_setup_kafka.sh
```

Step 4: Start Suricata (with Flow Logging)

````
## Components

### 1. DPDK Packet Ingestion
- Binds network interface to DPDK driver (vfio-pci/uio_pci_generic)
- Zero-copy packet capture
- High-performance packet processing

### 2. Suricata IDS
- Runs in DPDK mode
- Processes packets with signature-based detection
- Outputs alerts to Kafka via eve-kafka plugin

### 3. Kafka Streaming
- Acts as message broker
- Buffers alerts between Suricata and ML
- Enables scalable processing

### 4. ML Inference Engine
- Consumes alerts from Kafka
- Extracts features from network events
- Performs real-time threat classification
- Outputs enhanced predictions

## Directory Structure

```
dpdk_suricata_ml_pipeline/
├── README.md                    # This file
├── SETUP_GUIDE.md               # Detailed setup instructions
├── config/                      # Configuration files
│   ├── suricata-dpdk.yaml       # Suricata config with DPDK & Kafka
│   ├── kafka-config.properties # Kafka settings
│   └── pipeline.conf            # Pipeline configuration
├── scripts/                     # Management scripts
│   ├── 01_bind_interface.sh     # Bind NIC to DPDK
│   ├── 02_setup_kafka.sh        # Install/configure Kafka
│   ├── 03_start_suricata.sh     # Start Suricata in DPDK mode
│   ├── 04_start_ml_consumer.sh # Start ML inference
│   ├── 05_replay_traffic.sh     # Replay PCAP files
│   ├── stop_all.sh              # Stop all services
│   ├── status_check.sh          # Check pipeline status
````

PROF
````

```
│   └── unbind_interface.sh     # Restore network interface
├── src/                         # Python source code
│   ├── ml_kafka_consumer.py    # ML inference consumer
│   ├── feature_extractor.py    # Extract features from Suricata alerts
│   ├── model_loader.py         # Load ML models
│   └── alert_processor.py      # Process and store predictions
├── pcap_samples/                # Sample PCAP files for testing
│   └── README.md
└── logs/                        # Pipeline logs
    ├── dpdk/
    ├── suricata/
    ├── kafka/
    └── ml/
```

Quick Start

Prerequisites
- DPDK installed (run `install_dpdk_suricata.sh`)
- Suricata installed with DPDK support
- Kafka installed and running
- Python virtual environment activated

Step 1: Configure Interface
Edit `config/pipeline.conf` and set your network interface:
```bash
NETWORK_INTERFACE="eth0"  # Change to your interface
```

Step 2: Bind Interface to DPDK
```bash
cd dpdk_suricata_ml_pipeline
sudo ./scripts/01_bind_interface.sh
```

PROF

Step 3: Start Kafka
```bash
./scripts/02_setup_kafka.sh
```

Step 4: Start Suricata
```bash
sudo ./scripts/03_start_suricata.sh
```

Step 5: Start ML Consumer
```bash
./scripts/04_start_ml_consumer.sh
```

Step 6: Replay Traffic
```bash
sudo ./scripts/05_replay_traffic.sh pcap_samples/sample.pcap
```

```

### Monitor Pipeline
```bash
./scripts/status_check.sh
```

Stop Everything
```bash
sudo ./scripts/stop_all.sh
```

Traffic Sources

Option 1: tcpreplay (PCAP files)
Replay captured traffic from PCAP files:
```bash
sudo tcpreplay -i eth0 capture.pcap
```

Option 2: External System
Send live traffic from another machine:
- Configure second machine to send to monitored interface
- Use hping3, scapy, or actual application traffic

Option 3: Traffic Generator
Use DPDK pktgen or similar:
```bash
dpdk-pktgen -l 0-3 -n 4 -- -P -m "[1:2].0" -f traffic.lua
```

Monitoring

Check DPDK Status
```bash
dpdk-devbind.py --status
grep Huge /proc/meminfo
```

Check Suricata
```bash
tail -f logs/suricata/suricata.log
tail -f /var/log/suricata/eve.json
```

Check Kafka
```bash
kafka-console-consumer --bootstrap-server localhost:9092 \
    --topic suricata-alerts --from-beginning
```

Check ML Consumer
```bash
```
```

```
tail -f logs/ml/ml_consumer.log
```

Performance Tuning

DPDK
- Allocate more hugepages: 4-8GB recommended
- Use CPU isolation: `isolcpus=` in GRUB
- Bind to isolated CPU cores

Suricata
- Increase worker threads
- Tune ring buffers
- Disable unnecessary features
- Use optimized rules

Kafka
- Increase buffer sizes
- Adjust retention policies
- Configure compression
- Tune batch sizes

ML Inference
- Batch predictions
- Use GPU if available
- Cache model in memory
- Parallel processing

Troubleshooting

See `SETUP_GUIDE.md` for detailed troubleshooting steps.

License

Part of IDS Project - October 2025