🚀 Production DPDK Pipeline Guide

Full Stack: External Traffic → DPDK NIC → Suricata (DPDK) → Kafka → ML Flow Prediction

Status: ✅ FULLY IMPLEMENTED
Updated: October 3, 2025

📋 Table of Contents

🎯 Overview

This guide covers running the **complete production pipeline** with:

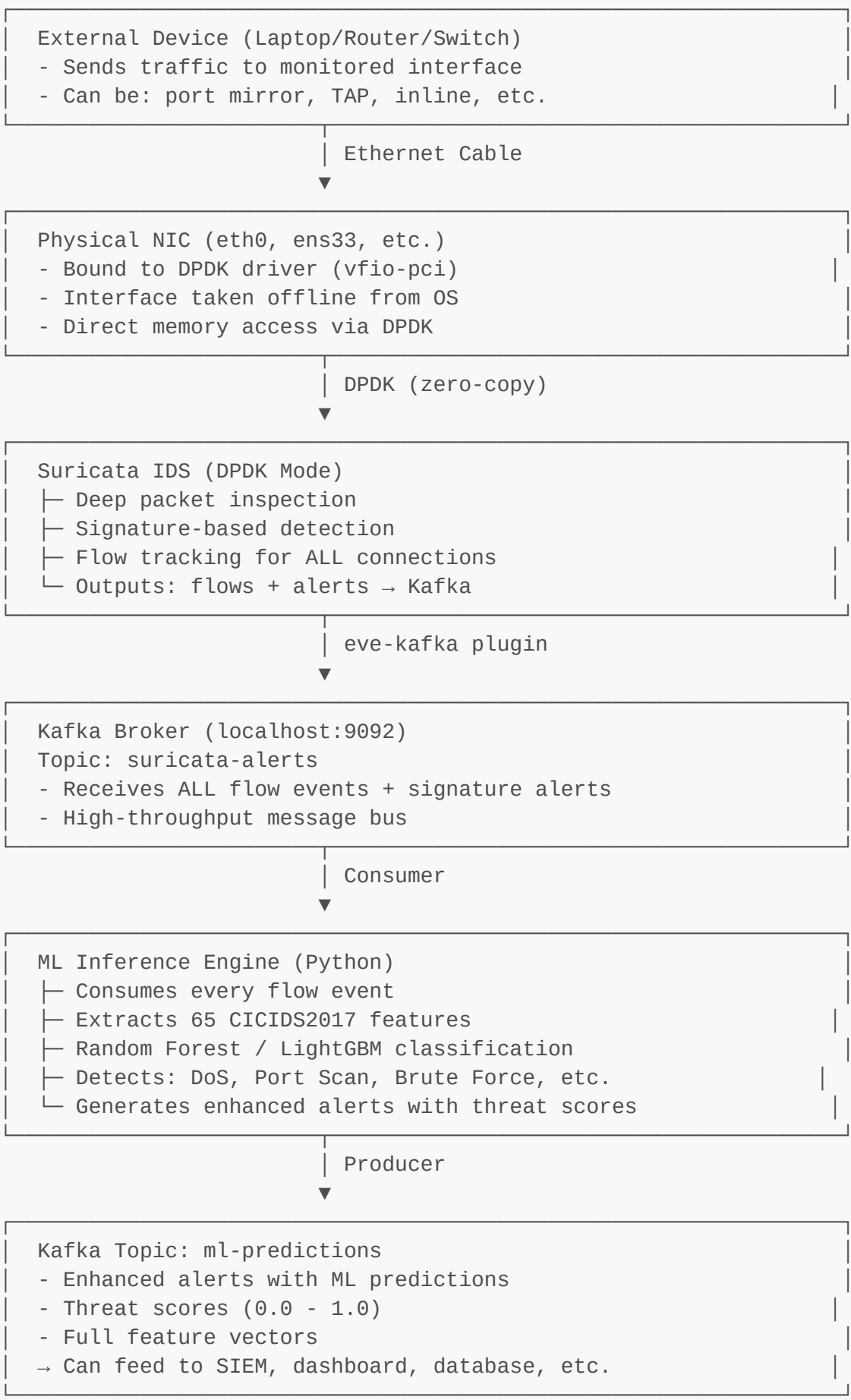
✅ **DPDK-bound network interface** for high-performance packet capture
✅ **External traffic** from a separate device/network
✅ **Suricata in DPDK mode** for packet processing
✅ **Flow-based ML inference** for ALL network flows
✅ **Real-time threat detection** with combined signature + ML detection

PROF

What's Different from Test Mode?

Feature	Test Mode (PCAP)	Production Mode (DPDK)
Traffic Source	PCAP replay	External device/network
Network Interface	Loopback/Any	DPDK-bound physical NIC
Suricata Mode	AF_PACKET	DPDK
Performance	Limited	High-performance (zero-copy)
Use Case	Testing/Development	Production monitoring

🏗️ Architecture


External Device (Laptop/Router/Switch)
- Sends traffic to monitored interface
- Can be: port mirror, TAP, inline, etc.
Ethernet Cable
Physical NIC (eth0, ens33, etc.)
- Bound to DPDK driver (vfio-pci)
- Interface taken offline from OS
- Direct memory access via DPDK
DPDK (zero-copy)
Suricata IDS (DPDK Mode)
├─ Deep packet inspection
├─ Signature-based detection
├─ Flow tracking for ALL connections
└─ Outputs: flows + alerts → Kafka
eve-kafka plugin
Kafka Broker (localhost:9092)
Topic: suricata-alerts
- Receives ALL flow events + signature alerts
- High-throughput message bus
Consumer
ML Inference Engine (Python)
├─ Consumes every flow event
├─ Extracts 65 CICIDS2017 features
├─ Random Forest / LightGBM classification
├─ Detects: DoS, Port Scan, Brute Force, etc.
└─ Generates enhanced alerts with threat scores
Producer
Kafka Topic: ml-predictions
- Enhanced alerts with ML predictions
- Threat scores (0.0 - 1.0)
- Full feature vectors
→ Can feed to SIEM, dashboard, database, etc.

PROF

✅ Prerequisites

1. Hardware Requirements

- **Dedicated Network Interface**: One NIC that can be taken offline
  - This will be bound to DPDK and unavailable to the OS
  - Should NOT be your primary network interface (unless you have another way to access the system)
- **CPU**: Minimum 4 cores recommended
  - 2 cores for Suricata DPDK workers
  - 1 core for ML processing
  - 1 core for system/Kafka
- **RAM**: Minimum 8GB
  - 2GB for DPDK hugepages
  - 4GB for Suricata
  - 2GB for ML model + Python
- **External Traffic Source**: Device to send traffic through the NIC
  - Can be: laptop, router, switch with port mirroring, network TAP, etc.

2. Software Requirements

Run the status check:

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline/scripts
./status_check.sh
```

PROF

Expected:

- ✅ DPDK installed
- ✅ Suricata with DPDK support
- ✅ Hugepages allocated (2GB+)
- ✅ Kafka ready
- ✅ Python virtual environment with all packages
- ✅ ML models available

3. Installation Check

If anything is missing:

```
# Install DPDK + Suricata
cd /home/sujay/Programming/IDS
```

```
sudo ./install_dpdk_suricata.sh

# Install Python packages
cd dpdk_suricata_ml_pipeline
./install_missing_packages.sh

# Verify
./scripts/status_check.sh
```

⚙ Configuration

Step 1: Edit Pipeline Configuration

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline
nano config/pipeline.conf
```

Key Settings to Configure:

```
# ======================================
# CRITICAL: Network Interface
# ======================================
# This interface will be TAKEN OFFLINE and bound to DPDK!
# DO NOT use your primary network interface unless you have
console/physical access!

NETWORK_INTERFACE="eth1"              # Change to your monitoring
interface
                                      # Common names: eth0, eth1, ens33,
enp0s8, etc.

# PCI address (auto-detected, but you can specify)
INTERFACE_PCI_ADDRESS=""              # Leave empty for auto-detection
                                      # Example: "0000:00:08.0"

# DPDK driver (vfio-pci recommended for modern systems)
DPDK_DRIVER="vfio-pci"                # Options: vfio-pci,
uio_pci_generic, igb_uio

# ======================================
# DPDK Resources
# ======================================
DPDK_HUGEPAGES="2048"                 # 2GB hugepages (increase for high
traffic)
DPDK_CORES="0,1"                      # CPU cores for DPDK (use isolated
cores if possible)
DPDK_MEMORY_CHANNELS="4"              # Match your system (usually 2 or
4)
```

```
# ======================================
# Suricata Configuration
# ======================================
SURICATA_CORES="2"                    # Number of worker threads (match
DPDK_CORES)
SURICATA_HOME_NET="192.168.0.0/16"    # YOUR network range (update this!)
                                       # Examples: "10.0.0.0/8",
"172.16.0.0/12"

# ======================================
# ML Model Selection
# ======================================
ML_MODEL_PATH="../ML Models/random_forest_model_2017.joblib"
# Alternative: "../ML Models/lgb_model_2018.joblib"

# ======================================
# Performance Tuning
# ======================================
ML_BATCH_SIZE="100"                   # Process flows in batches
ML_CONFIDENCE_THRESHOLD="0.7"         # Alert threshold
STATS_INTERVAL_SECONDS="10"           # Statistics reporting interval
```

Step 2: Identify Your Network Interface

Important: Choose the right interface!

```
# List all network interfaces
ip link show

# Example output:
# 1: lo: <LOOPBACK,UP,LOWER_UP> ...
# 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> ...  ← Primary interface
(DON'T USE)
# 3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> ...  ← Monitoring interface
(USE THIS)
```

PROF

Recommendations:

- ✅ Use a **secondary interface** (eth1, ens34, etc.)
- ✕ **DO NOT** use your primary network interface unless:
  - You have physical/console access to the machine
  - You have another way to access it (IPMI, KVM, etc.)
  - You're comfortable restoring network manually

Step 3: Check Interface Details

```
```

```
# Get interface information
ip addr show eth1           # Replace eth1 with your interface
ethtool -i eth1             # Shows driver and PCI address

# Example output:
driver: e1000               # Current driver
bus-info: 0000:00:08.0    # PCI address (will be needed for DPDK)
```

Step 4: Verify Hugepages

```
# Check current hugepages
grep Huge /proc/meminfo

# Should show:
# HugePages_Total:    1024   (or more)
# HugePages_Free:     1024
# Hugepagesize:       2048 kB
```

If hugepages are not allocated:

```
# Allocate 2GB of hugepages (1024 pages × 2MB)
sudo sysctl -w vm.nr_hugepages=1024

# Make permanent (add to /etc/sysctl.conf)
echo "vm.nr_hugepages=1024" | sudo tee -a /etc/sysctl.conf
```

🚀 Step-by-Step Execution

PROF

Phase 1: Start Kafka Message Broker

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline/scripts
./02_setup_kafka.sh
```

Expected Output:

```
╔═══════════════════════════════════════════════════╗
║  Kafka Setup Script                               ║
╚═══════════════════════════════════════════════════╝

✓ Kafka downloaded
✓ Starting Zookeeper on port 2181...
✓ Starting Kafka on port 9092...
```

```
✓ Creating topic: suricata-alerts
✓ Creating topic: ml-predictions
✅ Kafka is ready!
```

Verification:

```
# Check Kafka is running
netstat -tuln | grep 9092

# List topics
kafka-topics.sh --list --bootstrap-server localhost:9092
```

Wait 30 seconds for Kafka to fully initialize before proceeding.

Phase 2: Bind Network Interface to DPDK

⚠ **CRITICAL WARNING**: This will take your network interface **OFFLINE**!

Before running:

1. ✅ Ensure you're NOT using your primary network interface
2. ✅ Ensure you have another way to access the system (console, IPMI, second NIC)
3. ✅ Save your work (the interface will go down)

```
# Bind the interface
sudo ./01_bind_interface.sh
```

Expected Output:

PROF

```
╔═══════════════════════════════════════════════════╗
║  DPDK Interface Binding Script                    ║
╚═══════════════════════════════════════════════════╝

✓ Configuration loaded
Interface: eth1
PCI Address: 0000:00:08.0
Current Driver: e1000
Target Driver: vfio-pci

⚠  WARNING ⚠
This will bind eth1 to DPDK driver.
The interface will be taken OFFLINE and unavailable for normal use!

Continue? (yes/no): yes
```

```
✓ Backing up interface configuration
✓ Taking interface down
✓ Loading DPDK kernel modules (vfio-pci)
✓ Binding interface to DPDK
✅ Interface eth1 successfully bound to DPDK!

DPDK Status:
Network devices using DPDK-compatible driver
============================================
0000:00:08.0 'Device 100e' drv=vfio-pci unused=e1000
```

Verification:

```
# Check DPDK binding
dpdk-devbind.py --status

# Should see your interface under "Network devices using DPDK-compatible
driver"
```

If you lose network access:

```
# From console/physical access:
sudo dpdk-devbind.py -u 0000:00:08.0          # Unbind from DPDK
sudo dpdk-devbind.py -b e1000 0000:00:08.0   # Bind back to original
driver
sudo ip link set eth1 up                      # Bring interface up
sudo dhclient eth1                            # Get IP address
```

Phase 3: Start Suricata in DPDK Mode

PROF

```
sudo ./03_start_suricata.sh
```

Expected Output:

```
╔═══════════════════════════════════════════╗
║  Suricata DPDK Start Script               ║
╚═══════════════════════════════════════════╝

✓ Suricata with DPDK support detected
✓ DPDK interface bound
✓ Kafka running

Creating Suricata DPDK configuration...
```

```
✓ Suricata config generated at /etc/suricata/suricata-dpdk.yaml

Starting Suricata in DPDK mode...
✅ Suricata started successfully (PID: 12345)

Suricata is now:
  - Capturing packets via DPDK (0000:00:08.0)
  - Processing with 2 worker threads
  - Logging ALL flows (not just alerts)
  - Sending events to Kafka topic: suricata-alerts

Monitor logs:
  sudo tail -f /var/log/suricata/suricata.log
  sudo tail -f /var/log/suricata/stats.log
```

Verification:

```
# Check Suricata is running
pgrep -a suricata

# Check Suricata logs
sudo tail -f /var/log/suricata/suricata.log

# Should see:
# [DPDK] DPDK interface 0000:00:08.0 running in DPDK mode
# [kafka] Kafka producer initialized
```

Monitor Suricata stats:

```
# Live statistics (refreshes every 10 seconds)
sudo suricatasc -c "dump-counters" | jq .

# Or watch the stats log
sudo tail -f /var/log/suricata/stats.log
```

PROF

Phase 4: Start ML Inference Consumer

Open a new terminal (or use tmux/screen):

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline

# Activate Python virtual environment
source ../venv/bin/activate
```

```
# Start ML consumer
python src/ml_kafka_consumer.py --config config/pipeline.conf --verbose
```

Expected Output:

```
🚀 Starting ML Kafka Consumer...
═══════════════════════════════════════════════════════

📋 Configuration:
   Kafka Broker: localhost:9092
   Input Topic: suricata-alerts
   Output Topic: ml-predictions
   Model Path: ../ML Models/random_forest_model_2017.joblib

🔄 Loading ML model...
✅ Successfully loaded ML model: random_forest_model_2017.joblib
   Model type: RandomForestClassifier
   Features: 65
   Classes: ['BENIGN', 'DoS Hulk', 'PortScan', 'DDoS', 'DoS GoldenEye',
...]

🔌 Connecting to Kafka...
✅ Connected to Kafka broker: localhost:9092
✅ Subscribed to topic: suricata-alerts

📊 ML Inference Engine Ready!
⏳ Waiting for flow events...
═══════════════════════════════════════════════════════
```

The consumer is now waiting for traffic!

PROF

Phase 5: Send Traffic from External Device

Now connect your external device and send traffic through the monitored interface.

Option A: Direct Connection (Laptop → Monitored NIC)

```
[External Laptop] ──ethernet cable──> [eth1 on IDS system]
```

From the external laptop:

```
# Configure IP on the same subnet
sudo ip addr add 192.168.100.10/24 dev eth0

# Send test traffic
```

```
ping 192.168.100.1
curl http://example.com
wget http://testmyids.com/test.txt
```

Option B: Port Mirroring (Switch/Router)

```
[Switch/Router] ──port mirror──> [eth1 on IDS system]
       │
       └── All network traffic
```

Configure your switch to mirror traffic from active ports to the monitoring port.

Option C: Network TAP

```
[Network] ──TAP──> [eth1 on IDS system]
              │
              └──> [Original destination]
```

Physical TAP device copies all traffic to monitoring interface.

Option D: Inline Mode

```
[Network] ──> [eth1] → [IDS System] → [eth2] ──> [Network]
```

IDS system acts as transparent bridge (requires 2 DPDK interfaces).

PROF

Phase 6: Observe ML Predictions

As traffic flows, you should see the ML consumer processing flows:

```
[2025-10-03 09:45:12] 📊 Flow Event Received
═══════════════════════════════════════════
Flow ID: abc123def456...
Source: 192.168.100.10:45678
Dest: 93.184.216.34:80
Protocol: TCP
Duration: 1.234s
Packets: 42 (fwd: 22, bwd: 20)
Bytes: 4096 (fwd: 2048, bwd: 2048)

🔍 Feature Extraction...
✅ Extracted 65/65 CICIDS2017 features
```

```
   Duration: 1.234
   Fwd Packets: 22
   Bwd Packets: 20
   Flow Bytes/s: 3320.42
   ... [62 more features]

🤖 ML Inference...
✅ Prediction: BENIGN
   Confidence: 0.987
   Threat Score: 0.013

📤 Publishing to ml-predictions topic...
✅ Published
═══════════════════════════════════════════════════

[2025-10-03 09:45:13] 📊 Flow Event Received
═══════════════════════════════════════════════════
Flow ID: def789ghi012...
Source: 10.0.0.5:54321
Dest: 192.168.100.10:22
Protocol: TCP
Duration: 0.052s
Packets: 150 (fwd: 120, bwd: 30)
Bytes: 18000 (fwd: 15000, bwd: 3000)

🔍 Feature Extraction...
✅ Extracted 65/65 CICIDS2017 features
   Duration: 0.052
   Fwd Packets: 120
   Bwd Packets: 30
   Flow Pkts/s: 2884.62
   PSH Flag Count: 120  ⚠
   ... [62 more features]

🤖 ML Inference...
🚨 Prediction: SSH-Patator
   Confidence: 0.947
   Threat Score: 0.947 ⚠  ATTACK DETECTED!

📤 Publishing to ml-predictions topic...
✅ Published

⚠  ALERT: High threat score detected!
═══════════════════════════════════════════════════
```

PROF

📊 Monitoring

Real-Time Monitoring Dashboard

Terminal 1: Suricata stats

```
sudo tail -f /var/log/suricata/stats.log | grep -E "capture\.|flow\."
```

Terminal 2: ML Consumer output

```
# Already running from Phase 4
```

Terminal 3: Kafka topics

```
# Monitor incoming flows from Suricata
kafka-console-consumer.sh \
    --bootstrap-server localhost:9092 \
    --topic suricata-alerts \
    --from-beginning

# Monitor ML predictions (in another terminal)
kafka-console-consumer.sh \
    --bootstrap-server localhost:9092 \
    --topic ml-predictions \
    --from-beginning
```

Terminal 4: System resources

```
# Watch system resources
htop

# Watch network statistics
watch -n 1 'dpdk-devbind.py --status'
```

PROF

Check Pipeline Status

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline/scripts
./status_check.sh
```

Expected Output:

```
╔═══════════════════════════════════════════════╗
║  IDS Pipeline Status Check                    ║
╚═══════════════════════════════════════════════╝

▶ DPDK Status
```

```
──────────────────────────────────────────────────────
✓ DPDK installed and 1 device(s) bound

Network devices using DPDK-compatible driver
============================================
0000:00:08.0 'Device 100e' drv=vfio-pci unused=e1000

Hugepages:
HugePages_Total:    1024
HugePages_Free:      512

▶ Kafka Status
──────────────────────────────────────────────────────
✓ Kafka running on port 9092

Topics:
  suricata-alerts
  ml-predictions

▶ Suricata Status
──────────────────────────────────────────────────────
✓ Suricata running (PID: 12345)
✓ DPDK support enabled

Recent stats:
  capture.kernel_packets: 15234
  capture.kernel_drops: 0
  flow.tcp: 523
  flow.udp: 128

▶ ML Consumer Status
──────────────────────────────────────────────────────
✓ ML Consumer running (PID: 12456)

Recent activity:
  [2025-10-03 09:45:12] Processed 150 flows
  [2025-10-03 09:45:22] Detected 3 attacks
  [2025-10-03 09:45:32] Avg processing time: 12ms
```

PROF

Performance Metrics

```bash
# Suricata throughput
sudo suricatasc -c "dump-counters" | jq '.message |
  {
    packets: .capture.kernel_packets,
    drops: .capture.kernel_drops,
    flows: .flow.tcp + .flow.udp,
    alerts: .detect.alert
  }'
```

```
# Kafka message counts
kafka-run-class.sh kafka.tools.GetOffsetShell \
    --broker-list localhost:9092 \
    --topic suricata-alerts \
    --time -1

# ML consumer metrics (shown in console output)
# - Flows/second processed
# - Average feature extraction time
# - Average ML inference time
# - Attack detection rate
```

🧪 Testing with External Device

Test 1: Basic Connectivity

From external laptop:

```
# Ping test (generates ICMP flows)
ping -c 100 192.168.100.1

# Check in ML consumer - should see flows processed
```

Test 2: HTTP Traffic

From external laptop:

```
# Generate HTTP traffic
curl http://example.com
wget http://testmyids.com
ab -n 1000 -c 10 http://192.168.100.1/  # Apache bench
```

PROF

Test 3: SSH Traffic

From external laptop:

```
# Normal SSH (should be classified as BENIGN)
ssh user@192.168.100.1

# Simulated brute force (should trigger SSH-Patator detection)
for i in {1..100}; do
    sshpass -p "wrongpassword" ssh -o StrictHostKeyChecking=no
user@192.168.100.1
    sleep 0.1
done
```

Test 4: Port Scan

From external laptop:

```
# Run nmap scan (should trigger PortScan detection)
nmap -sS 192.168.100.1
nmap -sV -p- 192.168.100.1

# Check ML consumer for PortScan classification
```

Test 5: DoS Attack Simulation

From external laptop:

```
# SYN flood (should trigger DoS detection)
hping3 -S -p 80 --flood 192.168.100.1

# HTTP flood
ab -n 100000 -c 100 http://192.168.100.1/
```

Test 6: Mixed Traffic

From external laptop:

```
# Run comprehensive test script
cat > traffic_generator.sh << 'EOF'
#!/bin/bash

TARGET="192.168.100.1"

echo "Generating benign traffic..."
ping -c 50 $TARGET &
curl -s http://example.com > /dev/null &

echo "Generating suspicious traffic..."
nmap -sS $TARGET &

echo "Generating malicious traffic..."
for i in {1..50}; do
    nc -w 1 $TARGET 22 < /dev/null
    sleep 0.1
done

wait
echo "Traffic generation complete"
```

PROF

```
EOF

chmod +x traffic_generator.sh
./traffic_generator.sh
```

🐛 Troubleshooting

Issue 1: Interface Binding Fails

Symptoms: `01_bind_interface.sh` fails, interface won't bind

Solutions:

```
# 1. Check DPDK is installed
dpdk-devbind.py --version

# 2. Load kernel modules manually
sudo modprobe vfio-pci
sudo modprobe uio_pci_generic

# 3. Check hugepages
grep Huge /proc/meminfo
# If insufficient, allocate more:
sudo sysctl -w vm.nr_hugepages=1024

# 4. Check interface is not in use
sudo lsof -i  # Check no processes using the interface
sudo ip link set eth1 down

# 5. Try alternative driver
# Edit config/pipeline.conf:
DPDK_DRIVER="uio_pci_generic"  # Instead of vfio-pci
```

PROF

Issue 2: Suricata Not Capturing Packets

Symptoms: Suricata running but no packets captured

Solutions:

```
# 1. Verify DPDK interface is bound
dpdk-devbind.py --status | grep -A 5 "DPDK-compatible"

# 2. Check Suricata is using DPDK
sudo tail -f /var/log/suricata/suricata.log | grep DPDK

# 3. Verify external traffic is reaching the interface
# (This is tricky since interface is bound to DPDK)
```

```
# Use tcpdump on another interface to verify traffic is being sent

# 4. Check Suricata configuration
sudo grep -A 20 "dpdk:" /etc/suricata/suricata-dpdk.yaml

# 5. Restart Suricata with verbose logging
sudo pkill suricata
sudo suricata -c /etc/suricata/suricata-dpdk.yaml --dpdk -vvv
```

Issue 3: No Flows in Kafka

Symptoms: Suricata running, but Kafka topic empty

Solutions:

```
# 1. Check Kafka is running
netstat -tuln | grep 9092

# 2. Verify Suricata eve-kafka output is enabled
sudo grep -A 10 "eve-kafka" /etc/suricata/suricata-dpdk.yaml

# 3. Check Suricata can connect to Kafka
sudo tail -f /var/log/suricata/suricata.log | grep -i kafka

# 4. Manually check Kafka topic
kafka-topics.sh --describe --topic suricata-alerts --bootstrap-server
localhost:9092

# 5. Enable flow logging
# Edit /etc/suricata/suricata-dpdk.yaml:
outputs:
  - eve-log:
      enabled: yes
      filetype: kafka
      kafka:
        topic: suricata-alerts
      types:
        - flow:
            enabled: yes  # ← Ensure this is yes
```

PROF

Issue 4: ML Consumer Not Processing

Symptoms: ML consumer connected but not processing flows

Solutions:

```
# 1. Verify flows are in Kafka
kafka-console-consumer.sh \
```

```
    --bootstrap-server localhost:9092 \
    --topic suricata-alerts \
    --max-messages 5

# 2. Check ML consumer is subscribed
# Should see in console:
# "✅ Subscribed to topic: suricata-alerts"

# 3. Check for errors in ML consumer
# Look for:
# - Feature extraction errors
# - Model loading errors
# - Kafka connection errors

# 4. Test ML model manually
source ../venv/bin/activate
python << EOF
import joblib
model = joblib.load('../ML Models/random_forest_model_2017.joblib')
print(f"Model loaded: {type(model)}")
print(f"Features: {model.n_features_in_}")
EOF

# 5. Restart ML consumer with debug logging
python src/ml_kafka_consumer.py --config config/pipeline.conf --verbose
--debug
```

Issue 5: Low Performance / Packet Drops

Symptoms: Suricata stats show high packet drops

Solutions:

PROF

```
# 1. Check system resources
htop  # Look for CPU/memory bottlenecks

# 2. Increase DPDK resources
# Edit config/pipeline.conf:
DPDK_HUGEPAGES="4096"          # Increase to 4GB
SURICATA_CORES="4"             # More worker threads

# 3. Tune Suricata buffer sizes
# Edit /etc/suricata/suricata-dpdk.yaml:
dpdk:
  interfaces:
    - interface: 0000:00:08.0
      threads: 4
      mempool-size: 65535        # Increase
      mempool-cache-size: 512    # Increase
```

```
# 4. Disable ML processing temporarily to isolate bottleneck
# Stop ML consumer, check if Suricata drops decrease

# 5. Use batching in ML consumer
# Edit src/ml_kafka_consumer.py or config:
ML_BATCH_SIZE="200"  # Increase batch size

# 6. Pin cores (advanced)
taskset -c 0,1 suricata ...
taskset -c 2,3 python src/ml_kafka_consumer.py ...
```

Issue 6: External Device Can't Reach IDS

Symptoms: External laptop can't send traffic to monitored interface

Solutions:

```
# 1. This is EXPECTED if interface is bound to DPDK!
#    DPDK-bound interfaces are not visible to the OS

# 2. For testing, use port mirroring instead:
#    Send traffic to a different IP on the network,
#    and mirror that port to the DPDK interface

# 3. Or use a network TAP device

# 4. Or run in AF_PACKET mode for testing:
# Stop DPDK mode:
sudo ./scripts/stop_all.sh
sudo ./scripts/unbind_interface.sh

# Start Suricata in AF_PACKET mode:
sudo suricata -c /etc/suricata/suricata.yaml -i eth1 --set
outputs.1.eve-log.enabled=yes
```

PROF

⚡ Performance Tuning

For High-Throughput Networks (1 Gbps+)

1. DPDK Configuration

```
# config/pipeline.conf
DPDK_HUGEPAGES="8192"          # 8GB for high traffic
DPDK_CORES="0,1,2,3"           # 4 cores
DPDK_MEMORY_CHANNELS="4"       # Match your system
```

2. Suricata Tuning

```
# Increase workers
SURICATA_CORES="4"

# Edit /etc/suricata/suricata-dpdk.yaml:
dpdk:
  eal-params:
    proc-type: primary

  interfaces:
    - interface: 0000:00:08.0
      threads: 4
      mempool-size: 131071        # Larger buffer
      mempool-cache-size: 1024
      rx-queues: 4                # Multiple RX queues

# Increase stream memory
stream:
  memcap: 4gb

# Tune defrag
defrag:
  memcap: 1gb
```

3. ML Consumer Optimization

```
# Batch processing
ML_BATCH_SIZE="500"

# Parallel processing (edit ml_kafka_consumer.py)
# Use multiprocessing for feature extraction
from multiprocessing import Pool

# Increase Kafka consumer buffer
consumer = KafkaConsumer(
    batch_size=500,
    fetch_min_bytes=10240,
    fetch_max_wait_ms=500,
    max_partition_fetch_bytes=1048576
)
```

PROF

4. System Tuning

```
# Increase network buffers
sudo sysctl -w net.core.rmem_max=134217728
```

```
sudo sysctl -w net.core.wmem_max=134217728
sudo sysctl -w net.core.rmem_default=134217728
sudo sysctl -w net.core.wmem_default=134217728

# Disable CPU frequency scaling
sudo cpupower frequency-set -g performance

# Isolate CPU cores for DPDK
# Add to /etc/default/grub:
GRUB_CMDLINE_LINUX="isolcpus=0,1,2,3 nohz_full=0,1,2,3
rcu_nocbs=0,1,2,3"
sudo update-grub
# Reboot required
```

🛑 Stopping the Pipeline

Clean Shutdown

```
cd /home/sujay/Programming/IDS/dpdk_suricata_ml_pipeline/scripts
sudo ./stop_all.sh
```

This will:

1. Stop ML consumer
2. Stop Suricata
3. Unbind DPDK interface (restore to normal driver)
4. Bring interface back up
5. Stop Kafka and Zookeeper

Manual Shutdown

PROF

```
# 1. Stop ML consumer
pkill -f ml_kafka_consumer.py

# 2. Stop Suricata
sudo pkill suricata

# 3. Unbind DPDK interface
sudo ./scripts/unbind_interface.sh

# 4. Stop Kafka
kafka-server-stop.sh
zookeeper-server-stop.sh
```

Emergency Network Recovery

If you lose network access and need to restore:

```
# From physical console access:

# 1. Find the PCI address
lspci | grep -i ethernet

# 2. Unbind from DPDK
sudo dpdk-devbind.py -u 0000:00:08.0

# 3. Bind back to original driver
sudo dpdk-devbind.py -b e1000 0000:00:08.0  # or your original driver

# 4. Bring interface up
sudo ip link set eth1 up
sudo dhclient eth1

# Or use systemd:
sudo systemctl restart NetworkManager
```

📈 Expected Performance

Typical Throughput

Traffic Load	Suricata DPDK	ML Consumer	Latency
Low (< 100 Mbps)	0% drops	< 20ms/flow	< 50ms
Medium (100-500 Mbps)	< 1% drops	< 50ms/flow	< 100ms
High (500 Mbps - 1 Gbps)	< 5% drops	< 100ms/flow	< 200ms
Very High (> 1 Gbps)	Tuning needed	Batching needed	< 500ms

PROF

Resource Usage

- **CPU**: 30-60% per core (Suricata), 20-40% per core (ML)
- **Memory**: 2-4 GB (Suricata), 1-2 GB (ML + Python)
- **Network**: Line rate capture with DPDK (up to 10 Gbps on supported NICs)

✅ Success Checklist

Before running in production:

- ☐ DPDK and Suricata installed with DPDK support
- ☐ Hugepages allocated (2GB minimum)
- ☐ Dedicated network interface identified (NOT primary interface!)
- ☐ Configuration file edited (`config/pipeline.conf`)

- [ ] Network ranges configured (`SURICATA_HOME_NET`)
- [ ] External traffic source ready (laptop, switch, TAP)
- [ ] Kafka running and topics created
- [ ] Python environment activated and packages installed
- [ ] ML model available and loading successfully
- [ ] Backup access method available (console, IPMI, second NIC)
- [ ] Tested in safe environment first

Run the status check:

```
./scripts/status_check.sh
```

🎓 Understanding Flow-Based Detection

What Makes This Different?

Traditional IDS (Signature-Only):

```
Packet → Signature Match? → Alert (only if match)
                          → Discard (if no match)
```

Blind spot: Zero-day attacks, subtle attacks, slow scans

This Pipeline (Flow-Based ML):

```
Packet → Suricata → Flow Event → ML Feature Extraction → Classification
                                        ↓
                                  Signature Check
                                        ↓
                                Combined Threat Score
```

Advantage: Detects ALL suspicious behavior, even without signatures

Attack Types Detected

The ML models can detect these CICIDS2017 attack categories:

1. **BENIGN** - Normal traffic
2. **DoS Hulk** - HTTP flood attack
3. **DoS GoldenEye** - HTTP flood variant
4. **DoS Slowloris** - Slow HTTP attack
5. **DoS Slowhttptest** - Slow HTTP test
6. **DDoS** - Distributed denial of service

7. **PortScan** - Network reconnaissance
8. **FTP-Patator** - FTP brute force
9. **SSH-Patator** - SSH brute force
10. **Bot** - Botnet traffic
11. **Web Attack - Brute Force**
12. **Web Attack - XSS**
13. **Web Attack - SQL Injection**
14. **Infiltration** - Network infiltration
15. **Heartbleed** - SSL vulnerability exploit

📚 Additional Resources

- **Suricata DPDK Guide**: https://suricata.readthedocs.io/en/latest/capture-hardware/dpdk.html
- **DPDK Documentation**: https://doc.dpdk.org/
- **Kafka Documentation**: https://kafka.apache.org/documentation/
- **CICIDS2017 Dataset**: https://www.unb.ca/cic/datasets/ids-2017.html
- **Flow-Based ML Architecture**: `FLOW_BASED_ML_ARCHITECTURE.md`

🎉 You're Ready for Production!

Your IDS pipeline is now configured for:

- ✅ High-performance DPDK packet capture
- ✅ Real-time Suricata IDS with signature detection
- ✅ ML-based anomaly detection on ALL flows
- ✅ Combined threat scoring
- ✅ Scalable Kafka-based architecture

Start monitoring real traffic now!

PROF

Questions or Issues?

- Check `RUNTIME_GUIDE.md` for general troubleshooting
- Check `QUICKSTART.md` for quick commands
- Check `START_HERE.md` for overview

Happy Intrusion Detecting! 🛡️🚀