DeepTrace: RAG-Augmented LLM Network Forensics Copilot

Arjun Ghoshal

November 8, 2025

Contents

Abstract

DeepTrace is an intelligent network forensics copilot integrating transformer-based flow embeddings, retrieval-augmented generation (RAG), and large language model (LLM) explainability. The system performs semantic understanding over network traffic, reconstructing events, explaining behaviors, and narrating causal hypotheses of intrusions with human-readable explanations.

1 Introduction

DeepTrace is a network forensics system for deep packet inspection (DPI), behavioral flow analysis, and intelligent anomaly detection. It combines traditional network security with modern machine learning and natural language processing.

Key Capabilities:

- Multi-source data ingestion (live capture and PCAP files)
- Advanced multi-tier feature extraction (temporal, statistical, protocol)
- Transformer-based semantic flow embeddings
- RAG-powered contextual retrieval and analysis
- LLM-driven explainable narratives
- Modular, extensible architecture

2 System Architecture

DeepTrace consists of six layered components processing network data from raw packets to semantic analysis:

Layer Overview:

1. **Data Source**: Live traffic (Scapy/PyShark), PCAP files, DPI enrichment
2. **Feature Extraction**: Flow reconstruction, temporal/statistical/protocol features
3. **Flow Embedding**: Transformer-based semantic representation (64-D vectors)
4. **RAG & Reasoning**: Vector database retrieval + LLM causal reasoning
5. **Visualization**: Grafana dashboards, CLI/web interface, reports

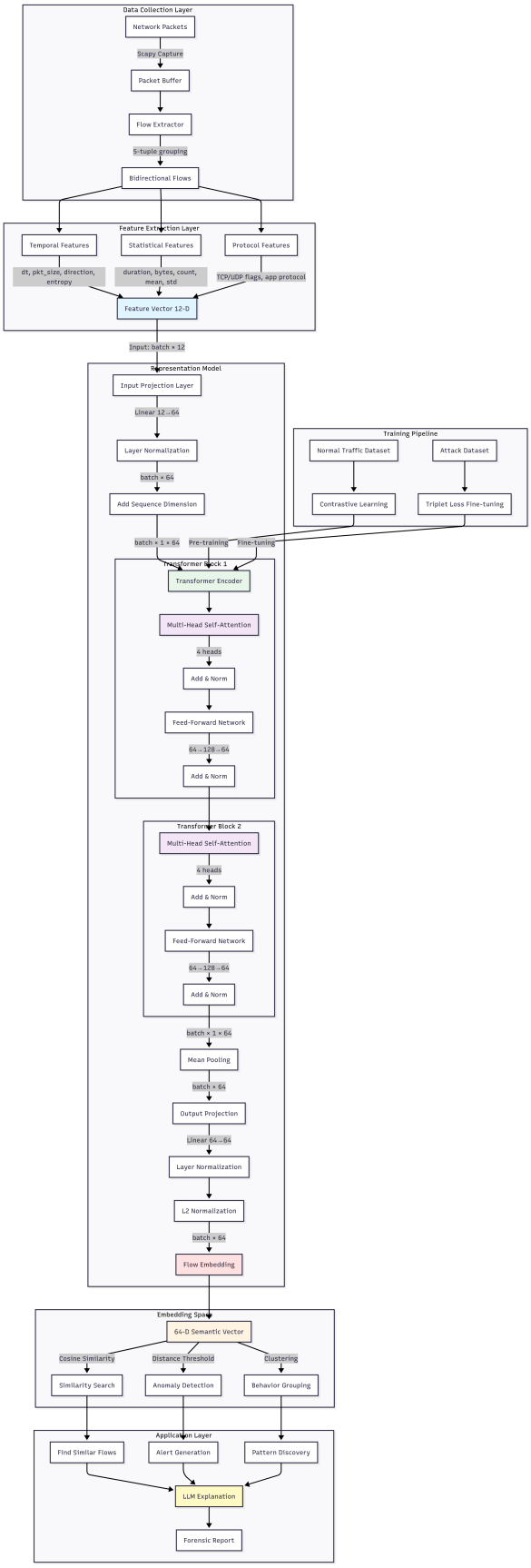

Figure 1: Architecture Design

3 Core Innovations

3.1 Semantic Flow Embeddings

Network flows are transformed into fixed 64-dimensional vectors capturing behavioral intent:

$$v_{\text{flow}} = f_\theta(\Delta t, \text{pkt\_sizes}, \text{direction}, \text{entropy}, \text{proto}) \tag{1}$$

The transformer encoder learns semantic patterns enabling: unsupervised anomaly detection, zero-shot novel attack detection, traffic classification, and behavioral profiling.

3.2 Retrieval-Augmented Generation

RAG grounds LLM explanations in actual network data:

1. Convert target flow to embedding v_{query}
2. Retrieve top-k similar flows from vector database (FAISS)
3. Assemble context with metadata, timestamps, labels
4. LLM generates explanation from retrieval context

Benefits: Factual grounding, context-aware analysis, reduced hallucination, continuous learning.

3.3 LLM-Powered Explainability

Transforms technical data into natural language narratives:

- Behavioral descriptions of network activity
- Anomaly explanations with reasoning
- Attack attribution and hypothesis formation
- MITRE ATT&CK technique mapping
- Investigation guidance for analysts

4 Flow Representation Model

4.1 Architecture

Input Features (12-D): flow_duration, total_bytes, packet_count, mean_pkt_size, std_pkt_size, mean_dt, std_dt, entropy_mean, entropy_std, byte_ratio, is_tcp, is_udp

Model Pipeline:

1. **Input Projection**: 12-D $\rightarrow$ 64-D with LayerNorm
2. **Transformer Encoder**: 2 layers, 4 attention heads, FFN dim=128
3. **Mean Pooling**: Sequence $\rightarrow$ single vector
4. **Output Projection**: 64-D with L2 normalization

4.2 Training Strategy

Phase 1 - Self-Supervised Pretraining (6 hours):

- Contrastive learning on 10,000 normal flows
- NT-Xent loss with temporal augmentation
- Batch size: 32, Learning rate: 1e-4

Phase 2 - Supervised Fine-tuning (3 hours):

- Triplet loss on 9,000 labeled flows (normal + attacks)
- Batch size: 16, Learning rate: 5e-5

5 Feature Engineering

5.1 Three-Tier Feature Hierarchy

1. Temporal Features (Per-Packet Time Series):

- **Inter-arrival time** (Δt): Detect beaconing, rate-limiting
- **Packet size sequences**: Identify exfiltration, tunneling
- **Direction indicators**: Analyze request-response, detect scans
- **Payload entropy**: Distinguish encrypted vs plaintext traffic

2. Statistical Features (Aggregated):

- Flow duration, total bytes, packet count
- Mean/std of packet sizes and inter-arrival times
- Entropy mean/std for consistency analysis
- Bidirectional byte ratio: $R = \frac{\text{bytes}_{\text{fwd}}}{\text{bytes}_{\text{total}}}$

3. Protocol Features (Categorical):

- Transport protocol (TCP/UDP/ICMP)
- Application protocol (HTTP/HTTPS/SSH/DNS)
- TCP flag sequences (detect scans, connection states)

5.2 Flow Identification

Flows identified by 5-tuple: (src_ip, src_port, dst_ip, dst_port, protocol). Bidirectional canonicalization ensures complete conversation tracking.

6 System Workflow

End-to-End Pipeline:

1. **Capture**: Scapy captures packets from interface/PCAP
2. **Flow Reconstruction**: Group packets by 5-tuple
3. **Feature Extraction**: Compute 12-D feature vector
4. **Embedding**: Transformer generates 64-D vector
5. **Vector Store**: Index in FAISS for similarity search
6. **RAG Query**: Retrieve similar flows + assemble context
7. **LLM Reasoning**: Generate explanation and report

7 Case Study: Traffic Analysis

7.1 Flow 1: HTTPS to Cloudflare CDN

Connection: 172.64.148.235:443 ↔ 192.168.1.40:56682

Features: Duration 0.31s, 3 packets, entropy 5.17, byte_ratio 0.624, flags A_P

Analysis: High entropy confirms HTTPS encryption. Server-dominant byte ratio typical of content download. RAG retrieves similar CDN flows (similarity ¿0.90).

LLM Output: *"Normal HTTPS traffic to Cloudflare CDN. High entropy confirms proper encryption. Server-dominant byte ratio indicates content delivery. Threat Level: LOW."*

7.2 Flow 2: Suspicious Beaconing Pattern

Connection: 192.168.1.40:49221 → 45.33.32.156:443

Features: Duration 5.4s, 18 packets, entropy 7.8, regular 180ms intervals, byte_ratio 0.48

Analysis: Periodic timing (180ms ± 5ms) deviates from normal HTTPS. Minimal data transfer (342 bytes avg) suggests automated beaconing. RAG retrieves Cobalt Strike patterns (similarity 0.89).

LLM Output: *"Flow exhibits periodic beaconing at 180ms intervals with minimal data transfer. Pattern matches Cobalt Strike C2 traffic (MITRE T1071). Recommend: isolate host, extract TLS cert, analyze process tree. Confidence: 0.91"*

8 Deployment Modes

Mode	Purpose	Throughput
Offline	PCAP analysis, training	1000-2000 flows/min
Live	Real-time monitoring	100-200 flows/sec
Interactive	Threat hunting, queries	On-demand
IDS Assistant	Alert enrichment	Alert-triggered

Usage Examples:

```
# Offline analysis
deeptrace analyze --pcap traffic.pcap --explain

# Live monitoring
sudo deeptrace monitor --interface eth0 --alert-threshold 0.8

# Interactive chat
deeptrace chat --llm openai

# Train custom model
deeptrace train --dataset flows.jsonl --epochs 50
```

9 Performance & Scalability

9.1 Optimization Techniques

- **Quantization**: INT8 reduces model size 75% with ¡2% accuracy loss
- **ONNX Runtime**: 2-3x inference speedup
- **Vector Search**: IVF-PQ indexing for large datasets
- **Batch Processing**: Optimal batch size 8-16 flows

10 Future Research Directions

10.1 Multi-Modal RAG

Integrate network flows with system logs, threat intelligence, and incident reports for holistic analysis:

$$v_{\text{unified}} = f_{\text{fusion}}(v_{\text{flow}}, v_{\text{log}}, v_{\text{intel}}, v_{\text{doc}}) \tag{2}$$

10.2 Continual Learning

Adapt to evolving threats through continuous self-supervised pretraining with elastic weight consolidation to prevent catastrophic forgetting.

10.3 Graph-Aware RAG

Incorporate network topology using Graph Neural Networks:

$$v_{\text{flow}}^{\text{context}} = \text{GNN}(v_{\text{flow}}, \mathcal{G}_{\text{neighborhood}}) \tag{3}$$

Enable lateral movement detection and attack campaign reconstruction through graph path analysis.

10.4 Hybrid Symbolic-Neural Explainability

Combine rule-based signatures with neural reasoning: rule engine identifies patterns, neural model validates, LLM synthesizes hybrid explanation.

11 Implementation

11.1 System Requirements

Minimum: 4-core CPU, 8GB RAM, 50GB SSD
Recommended: 8-core CPU, 16-32GB RAM, 500GB NVMe SSD

11.2 Installation

```
git clone https://github.com/deeptrace/deeptrace.git
cd deeptrace
python3 -m venv venv && source venv/bin/activate
pip install -r requirements.txt
sudo apt-get install libpcap-dev tcpdump

# Download pretrained model
wget https://models.deeptrace.ai/flow_encoder_v1.pt -O models/flow_encoder.pt

# Build vector index
python scripts/build_vector_index.py --flows data/historical_flows.jsonl
```

11.3 Configuration

```
capture:
  interface: eth0
  filter: "tcp or udp"

model:
  path: "models/flow_encoder.pt"
  device: "cpu"

vector_store:
  type: "faiss"
  index_path: "indexes/flow_index.bin"

llm:
  provider: "openai"
  model: "gpt-4"
  temperature: 0.3

alerts:
  threshold: 0.7
```

12 Evaluation Results

13 Conclusion

DeepTrace advances network forensics by integrating transformer-based embeddings, RAG, and LLM explainability. The system enables security analysts to understand complex behaviors through semantic

analysis, investigate with natural language, and explain findings with evidence-grounded narratives. The modular architecture supports deployment across diverse environments from small enterprises to large-scale distributed systems. Future enhancements in multi-modal integration, continual learning, and graph-aware reasoning will further strengthen capabilities.