Machine Learning-Based Network Attack Detection Using Graph Attention Networks

CYBERUS Master in Cybersecurity - Traffic Analysis Internship

Volonterio Luca

CYBERUS Master

Université Bretagne Sud

Keywords: Graph Neural Networks, Network Anomaly Detection, Graph Attention Networks

The Problem

Modern LAN Security Challenges:

- Increasing network complexity with heterogeneous devices (IoT, traditional computers)
- Sophisticated cyber threats targeting LANs as entry points
- Encrypted communication protocols hiding malicious activities
- Evolution of malware techniques evading traditional detection

Limitations of Current Approaches:

- Signature-based detection fails against zero-day attacks
- Polymorphic malware and encrypted traffic bypass traditional methods
- Need for behavior-based solutions without predefined signatures

Challenge: Detect unknown threats in encrypted traffic

Comprehensive Open-Source Proof-of-Concept

Key Innovation: Graph-based traffic analysis + Machine Learning

Dual-Module Architecture:

- High-performance C++ component for graph construction
- Python module with Graph Attention Networks (GATs) for anomaly detection

Primary Contributions:

- Efficient real-time graph representation framework
- Hybrid ML detection combining graph analysis + attention networks
- Complete deployable open-source solution

Result: Behavioral anomaly detection without signatures

Related Work Foundation

Three Key Influences:

1. **HyperVision**: Real-time encrypted malicious traffic detection
    - Flow interaction graph representation
    - Unsupervised ML approach
    - 0.92 AUC, 80.6 Gb/s throughput
2. **Dynamic Graph Survey**: Comprehensive anomaly detection taxonomy
    - GNN advantages: topological adaptation, multimodal integration
    - Three anomaly types: point, contextual, collective
3. **AddGraph**: Attention-based temporal GCN
    - Semi-supervised learning with limited labeled data
    - Edge-focused detection for suspicious connections

System Architecture Overview

Zeek Logs
Network Traffic

C++ Graph Builder (Log2Graph)

- Real-time network traffic monitoring
- Efficient graph construction from Zeek logs
- High-performance data processing

Graph Data

Python Graph Monitor

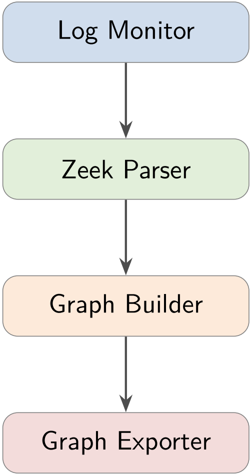
- GAT-based autoencoder for pattern learning
- Online learning and anomaly detection
- Visualization and analysis tools

Anomalies Detected Threats

Graph Builder (Log2Graph) Architecture

Core Classes & Roles:

- **LogMonitor & ZeekLogParser:** Log ingestion and parsing
- **GraphBuilder (Singleton):** Orchestrates graph construction
- **TrafficGraph:** Complete network representation
  - **GraphNode:** Network endpoints (IP addresses) with dynamic attributes
  - **AggregatedGraphEdge:** Connections aggregated by protocol/service/port
- **GraphExporter:** DOT format output for Python processing



Graph Monitor (Python) Architecture

Core Modules:

- **main.py:** Entry point and continuous monitoring
- **workflow.py:** High-level operational logic coordination
- **neural_net.py:** GAT-based autoencoder implementation
- **evaluate.py:** Performance assessment and feature quality rating

Two Operating Modes:

Train Mode
Learn normal network behavior patterns

Detect Mode
Identify deviations from learned patterns in real-time

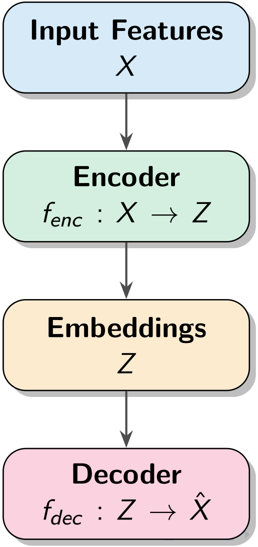
GAT Architecture Fundamentals

Why Graph Neural Networks?

- Traditional NNs work on regular structures (images, sequences)
- Networks are irregular, non-Euclidean structures
- GNNs handle message-passing on graph topologies

Attention Mechanisms:

- Learn adaptive weights for neighboring nodes
- Focus on most relevant connections for each node



Our GAT Architecture Details

NodeGNNAnomalyDetector Components:

1. **GAT Encoder:**
   - Multi-layer with edge feature integration
   - Multi-head attention (4 heads $\rightarrow$ 1 head)
   - Dimensions: Input $\rightarrow$ 64D $\rightarrow$ 32D embeddings
2. **Dual Pathway Detection:**
   - **Reconstruction Decoder:** Feature space recovery
   - **Anomaly Scoring MLP:** Direct pattern recognition
3. **Online Learning Framework:** Continuous adaptation with replay buffer
4. **Combined Decision Rule:** Union of reconstruction + MLP-based detection

Enhanced Attention Mechanism

Attention coefficient:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^T\left[Wh_i \,||\, Wh_j \,||\, W_e e_{ij}\right]\right)\right)}{\sum_k \exp\left(\text{LeakyReLU}\left(a^T\left[Wh_i \,||\, Wh_k \,||\, W_e e_{ik}\right]\right)\right)}$$

- Node feature projections — Wh_i, Wh_j are the learned embeddings (hidden states) of nodes i and j transformed by weight matrix W.
- Edge feature projection — $W_e e_{ij}$ transforms the edge features e_{ij} between nodes i and j.
- Attention vector — a^T is a learnable vector that scores the combined features to produce a raw attention score.
- LeakyReLU activation — introduces non-linearity

Network Graph Features

Node Features (IP Addresses):

- **Behavioral:** outgoing/incoming ratios, server/client scores
- **Protocol Diversity:** unique protocols, entropy measures
- **Port Patterns:** privilege ratios, diversity metrics
- **Temporal:** cyclical time features for daily rhythms
- **Traffic Volume:** bytes/packets sent/received

Edge Features (Connections):

- **Metadata:** protocol, service, destination port
- **Traffic Statistics:** bytes, packets, connection counts
- **Aggregation:** By source/dest IP, protocol, service, port

Rich Feature Set:
Captures behavioral patterns without payload inspection

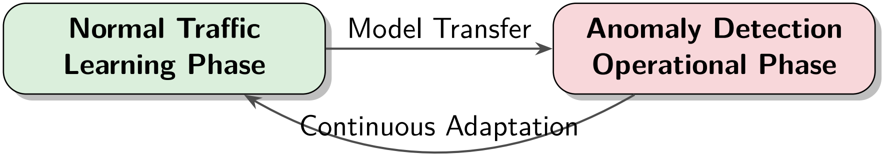
Online Learning & Adaptation

Sophisticated Continuous Learning:

- **Experience Replay:** Reservoir sampling to prevent forgetting
- **Statistical Monitoring:** Welford's algorithm for stability
- **Adaptive Learning Rate:** Cosine annealing with warm restarts

Train vs Detect Mode Workflow:

- **Training:** 50 epochs initial + 25 steps online updates
- **Detection:** Pre-trained model inference without parameter updates
- **Critical Dependency:** Comprehensive training with normal traffic required



Anomaly Detection Methodology

Dual-Threshold Approach:

1. **Statistical Threshold (Reconstruction-based):**
   - Flag if: reconstruction_error $> \mu + 2\sigma$
   - 95% coverage under Gaussian assumptions
2. **Learned Threshold (MLP-based):**
   - Flag if: anomaly_probability > 0.8
   - Direct pattern recognition

Combined Decision: Union of both approaches

Result: Maximized sensitivity
with maintained interpretability

Key Technical Innovations

Performance Optimizations:

- C++ for high-speed graph construction
- Incremental graph updates
- Efficient memory management

ML Innovations:

- Edge-aware attention mechanisms
- Dual-pathway anomaly detection
- Online learning with catastrophic forgetting prevention
- Focal loss for class imbalance handling

Practical Features:

- Real-time processing capability
- Scalable architecture
- Open-source implementation

Core Innovation: Behavioral analysis of encrypted traffic through graph-based machine learning

Summary & Next Steps

What We've Built:

- Complete graph-based network anomaly detection system
- Sophisticated GAT architecture with dual detection pathways
- Real-time processing with online learning capabilities

Key Strengths:

- Handles encrypted traffic through behavioral analysis
- Adapts to evolving network patterns
- Combines multiple detection approaches for robustness

Next Steps:

- Evaluation on CIC-IDS2017 dataset
- Deployment architecture optimization
- Performance benchmarking
- Integration with existing security infrastructure

Ready for Real-world Deployment

Thank You

Questions & Discussion

Volonterio Luca
CYBERUS Master in Cybersecurity
June 25, 2025