Infosys SpringBoard Model Research

Model Architectures for Threat Prediction Platform

Sanjay Dhakar (Group - 1)

1 . HYBRID MODEL — *Full System*

🧩 Composition:

LightGBM (Tabular) + Transformer (Sequence) + GNN (Entity Graph)
→ Combined via **meta-learner** or weighted ensemble.

⚙️ Data Flow

1. **Input Sources:**
   - Static CSVs (CICIDS, NSL-KDD, TON_IoT)
   - Live feeds (VirusTotal, OTX, AbuseIPDB)
   - Organization logs (CSV/PCAP/JSON)
2. **Feature Engineering:**
   - Aggregated tabular features (packet size, flags, frequency)
   - Sequential logs (events per host/time window)
   - Entity graph (IP ↔ domain ↔ host relationships)
3. **Model Pipeline:**
   - **LightGBM:** handles tabular statistics, reputation, numeric fields
   - **Transformer:** processes time-ordered event sequences per host
   - **GNN:** correlates multi-entity relationships
4. **Fusion Layer:**
   - Concatenate final embeddings → MLP or Logistic Regression → risk score
   - Output: Threat category + risk probability

💡 Why it's great

- Combines **pattern recognition (Transformer)** + **statistical reasoning (LightGBM)** + **contextual awareness (GNN)**
- Best overall detection accuracy (>94% F1 achievable)
- Modular & extendable (can run each model independently)

🧱 Implementation Stack

- Data: Pandas, DGL/PyG for graphs, HuggingFace Transformers for sequence modeling
- Model: LightGBM + Transformer encoder + GraphSAGE
- Serving: FastAPI or Flask API; ONNX export for unified inference
- Infra: Kafka (streaming), Redis (feature store), PostgreSQL (historical data)

2 . LIGHTGBM / CATBOOST ENSEMBLE

⚙️ How it works

1. Use preprocessed **tabular network data** (each row = flow, alert, or connection).
2. Add enriched metadata from WHOIS, GeoIP, OTX, etc.
3. Train LightGBM (or CatBoost if many categoricals).
4. Predict threat_probability or price_score (depending on task).

💡 Why it's great

- Fast training and inference
- Handles missing values and mixed data types
- Great baseline for comparing deeper models
- Explainable via SHAP

🧱 Use Case

- Real-time classification of alerts
- Works well as **Stage 1 filter model** in hybrid stack
- Use SHAP to explain why an event was flagged

3 . SEQUENCE TRANSFORMER — *Behavioral Context Model*

⚙️ How it works

1. Each host's recent activity (events, flows, commands) = a sequence.
2. Represent events as embeddings (categorical + numeric tokens).
3. Train **Transformer Encoder** to learn patterns over time.
4. Output = threat class / anomaly score.

🧩 Architecture

Input Sequence → Embedding → Positional Encoding →

Transformer Encoder Layers → Global Attention Pooling → Dense Output Head

💡 Why it's great

- Captures **contextual behavior**, not just static values
- Excellent for insider threat or lateral movement detection
- Transfer learning possible (like "BERT for logs")

🧱 Implementation

- Framework: PyTorch / TensorFlow
- Model: TransformerEncoder / BERT
- Optional Pretraining: Masked event prediction on large benign logs

4 . GRAPH NEURAL NETWORK (GNN) — *Entity & Relationship Intelligence*

⚙️ How it works

1. Construct a **graph**:
   - Nodes: IPs, domains, hashes, users
   - Edges: communications, resolutions, co-occurrence
2. Features: node degree, reputation, time-based activity
3. Train **GNN (GraphSAGE / GAT)** to classify nodes or predict malicious links.

🧩 Architecture

Node Features → Graph Convolutions (GCN/GAT) → Node Embeddings →

Classifier → Malicious/Benign

💡 Why it's great

- Detects coordinated attacks & related threat clusters
- Works even when direct labels are limited
- Scalable with sampled mini-batches (GraphSAGE)

🧱 Implementation

- DGL or PyTorch Geometric
- Graph store: Neo4j / NetworkX for static graph representation
- Output: node-level risk score or community risk level

5 . ANOMALY DETECTION STACK

⚙️ How it works

1. Model learns **normal baseline** using Autoencoder / Isolation Forest.
2. During inference, compute **reconstruction error or isolation score**.
3. High deviation → potential anomaly / zero-day.

🧩 Architecture

Input Features → Autoencoder → Reconstruction → Error Score

💡 Why it's great

- No labels needed — detects unseen threats
- Adds safety net around main classifiers
- Can be deployed parallel to hybrid classifier for continuous monitoring

🧱 Implementation

- Use sklearn.ensemble.IsolationForest or AutoEncoder (Keras/PyTorch)
- Threshold from ROC/PR curves for sensitivity tuning

⚙️ Combined Hybrid System Design (Recommended)

🔹 Stage 1: Data & Feature Pipeline

- Ingest → Normalize → Enrich (GeoIP, WHOIS, OTX) → Store
- Create per-host sequences and entity graphs

🔹 Stage 2: Model Inference Flow

Stage	Model	Output	Role
1	**LightGBM**	Threat probability	Fast filter baseline
2	**Transformer**	Sequence anomaly score	Contextual pattern detector
3	**GNN**	Entity correlation score	Multi-hop context
4	**Autoencoder**	Reconstruction error	Zero-day catch
5	**Meta-Learner**	Fused threat score	Final prediction

🔹 Stage 3: Output Layer

- Unified **Risk Score [0–1]**
- Predicted Threat Category (Malware / Phishing / DDoS / Others)
- Top 3 explanation features (via SHAP or Attention)