Dynamic Graph Embedding for Real-Time Anomaly Detection in Network Traffic

# Data Preprocessing

* We started with the KDD Cup 1999 dataset, which contains network traffic data.
* The dataset includes numerous columns, but for our specific task of real-time anomaly detection using dynamic graph embedding, we selected a subset of relevant columns.
* The selected columns include 'duration', 'protocol\_type', 'service', 'flag', 'src\_bytes', 'dst\_bytes', and others.
* We loaded the dataset into a Pandas DataFrame for further processing.

# Data Cleaning and Transformation

* In this step, we handled missing values, duplicate records, and any other data quality issues that might affect the analysis.
* We removed irrelevant columns that were not needed for our specific task.
* To prepare the data for machine learning and graph creation, we performed the following transformations:
* Encoded categorical columns ('protocol\_type', 'service', 'flag') using Label Encoding.
* Normalized numerical features ('duration', 'src\_bytes', 'dst\_bytes', etc.) using Standard Scaling.
* Created a binary target variable ('target') where 1 indicates an attack and 0 indicates normal traffic.
* Split the data into training, validation, and test sets (60% training, 20% validation, 20% test).

# Dynamic Graph Creation

* In this critical step, we transformed the preprocessed data into a dynamic graph suitable for real-time anomaly detection.
* We initialized an empty dynamic graph structure.
* We iterated over the data using time windows, where each time window represents a snapshot of the network traffic.
* For each time window, we created a snapshot graph using NetworkX.
* Nodes in the snapshot graph represent entities (e.g., IP addresses), and edges represent interactions between entities.
* We calculated interaction counts and used them as edge weights.
* We added each snapshot graph to the dynamic graph structure.
* The result was a dynamic graph that represents the evolving network traffic over time.

# Visualization (Efficiency Considerations)

* While visualization is a valuable tool for understanding dynamic graphs, we acknowledge that the traditional approach using NetworkX and Matplotlib might be inefficient for very large graphs.
* To improve efficiency in visualization, especially for large graphs, we discussed alternative approaches such as graph sampling, graph metrics, interactive visualization tools, and graph aggregation.
* These approaches can help explore and understand the dynamic graph without overwhelming the visualization tool.