**Network Traffic Classifier / Intrusion Detection System Using NSL-KDD Dataset**

# Objective

The primary goal of this project is to develop a robust Network Traffic Classifier Intrusion Detection System (IDS) that can classify network traffic as either normal or malicious using deep learning techniques.   
This project leverages the NSL-KDD dataset to train and test a deep learning model, which is later deployed through an interactive user interface using Gradio. The model is designed   
to identify and prevent potential intrusions or attacks in network traffic.

# Importance of the Project

Network security is a critical aspect of any organization, especially in today's world where cyber threats are constantly evolving. An Intrusion Detection System (IDS) serves as a   
first line of defense by identifying suspicious activities and potential attacks. The ability to detect these intrusions in real-time can prevent significant data breaches and loss.   
This project aims to enhance the capabilities of IDS by using machine learning to automate and improve accuracy in detection.  
  
Using the NSL-KDD dataset, which is an improved version of the KDD Cup 1999 dataset, ensures that the model is built on data that is free from redundant and duplicate records, providing a more realistic evaluation of the model's performance.

# End Goal

The ultimate aim is to build a functional, real-time intrusion detection / Network Traffic Classification model that not only accurately identifies malicious traffic but also offers an intuitive user interface.   
The trained model is deployed using Gradio, allowing end users to input network traffic data and receive immediate feedback on whether the traffic is normal or indicative of a possible intrusion. For testing purpose sample CSV files containing data of 5 packets of network traffic have to be uploaded.

# Data Preprocessing

The NSL-KDD dataset contains various features such as duration, protocol\_type, service, flag, src\_bytes, dst\_bytes, and others. One of the key steps in any machine learning project is data preprocessing.

This involves:  
  
A- Label Encoding: Since the dataset contains both numerical and categorical features, categorical features like protocol\_type and flag need to be encoded as numbers.

B- Normalization: To ensure that all features contribute equally to the model, we normalize the data. This is handled using StandardScaler.

C- Train-Test Split: The dataset is split into training and testing subsets (80% training and 20% testing), ensuring that the model can be properly validated during the training process.

# Model Building

The deep learning model is built using TensorFlow's Keras API. The architecture consists of multiple dense layers, with dropout layers included to prevent overfitting.

Here's a breakdown:  
  
Input Layer: The input layer matches the number of features in the dataset.  
Hidden Layers: Two hidden layers with ReLU activation functions are used to learn complex patterns in the data.  
Output Layer: A single neuron with a sigmoid activation function is used in the output layer to provide binary classification.  
Training: The model is trained over 20 epochs with a batch size of 64, ensuring that it has enough time to learn the features in the dataset without overfitting.

# Evaluation

Once the model is trained, it is evaluated on the test set using accuracy and classification reports. Metrics such as precision, recall, and F1-score are crucial for understanding how   
well the model is distinguishing between normal and malicious traffic.

# Gradio Interface

The model is then integrated into a Gradio interface, which allows users to input network traffic data and receive a real-time prediction of whether the traffic is normal or malicious.   
Gradio provides a simple and user-friendly interface, making the model accessible to users who may not have a technical background.

# Current and Future Use Cases

1. Network Monitoring: The model can be used in an organization's network to continuously monitor traffic for intrusions.  
2. Cloud-Based Security: This model could be adapted to cloud environments to detect threats in cloud-hosted networks.  
3. IoT Security: Expanding the model to monitor IoT devices can protect against various types of network-based attacks.  
4. User Authentication Monitoring: The model could be extended to detect abnormal user authentication patterns.

# Future Innovations

1. Real-Time Data Integration: Integrating real-time data streams would provide instant feedback on potential threats.   
2. Feature Engineering: Deriving additional features from network traffic data could improve detection capabilities.  
3. Model Optimization: Experimenting with other machine learning models could lead to a more robust detection system.