# Assignment: Real-Time Intrusion Detection with Scapy + PyTorch

**Dataset**: Network Intrusion Dataset (Kaggle) https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

**Submission format**: A single Jupyter Notebook ( `.ipynb` ) that:

- runs top-to-bottom without errors,
- uses code cells for code,
- uses Markdown cells for explanations, design, conclusions, and organization.

---

## Task 1 — Dataset Exploration and Feature Understanding

1. Load the Network Intrusion Dataset and list all its features.
2. For each feature, provide:
    - name and data type (numeric, categorical, etc.),
    - a clear description of what it represents,
    - and a justification (for one or a set of feature) of how it could be relevant for distinguishing between normal traffic and intrusions / attacks.
3. Write a short summary (3–5 sentences) about which features likely come from packet- or flow-level network traffic, and which may be derived or aggregated.

---

## Task 2 — Identify Scapy-Computable Features

1. From your full feature list, select a **subset** of features that you believe can be **extracted in real time** via Scapy, by capturing live network packets or flows.
2. Create a table with the following columns:
    - Feature
    - Computable on Linux Server? (yes/no)
    - Computable on Raspberry Pi? (yes/no)
    - Reason (why or why not, considering packet scope, processing cost, privileges, hardware)
    - Validation snippet (a short Python / Scapy code idea that *if executed* would test whether you can extract that feature)
3. Justify each decision: why some features are not computable on the Pi, or why they demand more resources.

---

## Task 3 — Real-Time Scapy Extractor

1. Write a Scapy-based program that:
    - Detects whether it's running on a **Linux server** or a **Raspberry Pi**,
    - Sniffs live traffic,
    - Extracts the subset of features you identified in Task 2,
    - Logs them (e.g., to CSV) in real time.

---

## Task 4 — Prepare Platform-Specific Datasets

1. Using the original Kaggle dataset, build **two sub-datasets**:
    - one containing only the features you can compute in real time on a **Linux server**,
    - another containing only those computable on a **Raspberry Pi**.
2. Clean and preprocess both sub-datasets (missing values, encoding, normalization, etc.).

3. Describe in Markdown all preprocessing steps and decisions (why you drop or transform certain columns, how you handle imbalances, etc.).

---

## Task 5 — Train and Optimize PyTorch Models

On each of the two datasets, train **three different models** using PyTorch:

- **Model 1 – MLP (fully connected)**
- **Model 2 – Convolutional Network** (e.g., 1D or 2D conv, depending on data representation)
- **Model 3 – Self-Attention Network** (e.g., Transformer encoder block or custom attention)

For each model & dataset:

1. Define architecture and hyperparameters.
2. Implement training and validation loops.
3. Perform hyperparameter tuning.
4. Report training vs validation loss / accuracy curves.
5. Save the best-performing version of each model.

---

## Task 6 — Model Evaluation and Comparison

1. Evaluate all six models (3 × 2 platforms) on a held-out test set using:
    - Accuracy
    - Precision / Recall
    - F1-score
    - ROC-AUC (if applicable)
    - Confusion matrix
    - Inference time per sample
    - Model size (on disk)
    - Memory usage during inference
2. Present results in a clear table or DataFrame.
3. Provide an analysis (bullet points) comparing the models:
    - Which model works best on Linux vs Pi?
    - What trade-offs do you see (speed vs accuracy, memory, platform constraints)?

---

## Task 7 — Compare to Literature

1. Find **at least two recent (last 5 years)** research articles or preprints that use this (or a highly similar) intrusion detection dataset.
2. For each:
    - Summarize their methodology (features, model architectures, training regime),
    - Report their performance metrics,
    - Compare their results with yours, and discuss any differences (in features, preprocessing, model design, dataset split, etc.).

Provide full citations and links for each article.

---

## Task 8 — Deployment & Real-Time Integration

1. Choose the **best model** for each platform (Linux server and Raspberry Pi) based on your evaluation.
2. Prepare each model for deployment:

- Export in a suitable format (e.g., onnx, TorchScript, etc.),
- If applicable, apply optimization (quantization, pruning, etc.) to reduce size/latency.

3. Modify your Scapy real-time extractor (from Task 3) so that it:
   - sends the extracted feature vectors into your deployed model,
   - performs inference live,
   - triggers an alert or logs when an intrusion is predicted.
4. Test this live integration.
5. In Markdown, document your testing process, show example alerts or predictions, analyze false positives / negatives, and give final conclusions and recommendations for a production environment.

---