Our project focuses on building an **attack prediction system** that analyzes Windows event logs to predict potential cyberattack scenarios using the MITRE ATT&CK framework. The pipeline begins by **extracting embeddings from logs**—either through full log vectorization or entity-level embedding—and then performs **retrieval over a custom-built vector space of MITRE techniques**. This retrieval is followed by **attack scenario construction using an attack graph**, which connects related techniques based on adversarial behavior. Finally, the system queries a **large language model (LLM)** to reason over the retrieved techniques and provide **probabilistic predictions** of attack progressions. So far, we have processed the EVTX-ATTACK-SAMPLES dataset by converting EVTX files to CSV format, updating MITRE technique mappings to the latest ATT&CK version (v14), and cleaning the dataset to retain only relevant files. We then explored a second dataset, the Atomic Red Team logs from Security Datasets, which we cleaned, processed, and combined with the first dataset. However, after analysis, we found that even **with both datasets combined, we only cover a limited portion of the MITRE matrix**.

To address the limited coverage of MITRE techniques in existing datasets, we designed a **Hybrid Log Generation Strategy** that enriches our training data by generating realistic and diverse synthetic logs. This approach leverages both **structured rule-based generation** and the **creative flexibility of language models**, while maintaining log realism through pretraining and validation. Here's the step-by-step workflow:

1. Pretrain on Real Windows Logs

- We begin by **pretraining our model on large volumes of real Windows event logs** (e.g., from EVTX-ATTACK-SAMPLES and Atomic Red Team).
- This allows the model to **learn the structure, syntax, and field relationships** typical of Windows logs (e.g., `EventID`, `CommandLine`, `Image`, etc.).
- As a result, the model becomes capable of **understanding and generating logs** that closely mirror real-world data formats.

2. Rule-Based Log Template Creation

- Using **MITRE ATT&CK detection rules** or **Sigma rules**, we generate **base log templates** for specific techniques (e.g., T1059.001 PowerShell Execution).
- These templates are **accurate, human-defined, and technique-aligned**, ensuring each synthetic log reflects known adversarial behavior.

3. AI-Augmented Log Diversification (via LLM)

- We feed the rule-based logs into a **large language model (e.g., GPT-4)** with prompts like:

  > "Given this base log for technique T1059.001, generate 5 realistic variations that maintain the core behavior but differ in parameters or context."

- The model generates **syntactic and semantic variations**—changing file paths, parameters, command-line flags, usernames, etc.—while preserving the core technique.
- These logs simulate the natural diversity found in real-world systems

4. Output Validation and Filtering

- To prevent hallucinations or malformed outputs, we apply **post-generation validation** using:
  - **Regex or YARA rules**
  - **Field-schema checks**
  - (Optional) fine-tuned classifiers for semantic consistency
- This step ensures only **syntactically valid and behaviorally accurate logs** are retained.

5. Feedback Loop and Embedding Optimization

- Generated logs are **embedded into the vector space** and used in downstream tasks like RAG and attack scenario prediction.
- Logs that perform poorly in retrieval or prediction can be filtered out.
- We can use performance metrics to **refine prompt strategies or fine-tune the model**, improving generation quality over time.

Why We Chose This Approach

- **Limited Coverage**: Real datasets only cover a fraction of MITRE techniques.
- **Balanced Control and Diversity**: Rule-based logs give us control; LLMs give us variety.
- **Improved Generalization**: Synthetic logs help the model generalize to unseen attack techniques.
- **Data Completeness**: Ensures our embedding space is rich, varied, and full-spectrum.