**Log Analysis for Threat Detection**

**Project Summary**

**Task Selection**

The goal of this project was to explore how prompt engineering can enhance cybersecurity automation specifically, the task of detecting potential threats from log entries. Logs contain system level activities such as login attempts, malware alerts, or suspicious behavior. Analyzing them manually is time consuming, so leveraging a large language model (LLM) like GPT-4 offers a scalable solution.

**Design Process**

The project was structured in a systematic, repeatable way:

1. **Dataset Creation**

A dataset of 5 synthetic but realistic log entries was generated. Each entry reflected a different cybersecurity scenario (e.g., brute-force attempts, malware detection, benign login).

1. **Prompt Variant Design**

Three distinct prompt templates were created using Jinja2:

* **Instruction-based:** Straightforward commands to analyze logs.
* **Role-based:** Framed GPT as a SOC (Security Operations Center) analyst.
* **Few-shot:** Included example outputs to guide response formatting.

1. **Prompt Execution**

Each prompt was tested against all log entries using GPT-4 via the OpenAI API. A Python script managed the template rendering, API calls, and output collection.

1. **Evaluation Rubric**

A 4-point evaluation rubric was applied to each output:

* Detection Accuracy
* Clarity of Analysis
* Severity Classification
* Actionability

1. **Prompt Refinement**

The best performing variant (role based) was refined for improved clarity and explicit structure, then re-evaluated across the same dataset.

**Results**

* The role-based prompt consistently outperformed the others, scoring highest in detection accuracy and actionability.
* After refinement, its average evaluation scores improved by +2 points on multiple log entries.
* The few-shot prompt performed reasonably well, but lacked consistency in output structure.
* Instruction-based prompts were clear, but often less contextual.

**Supporting Files**

* Scripts:  
  log\_analysis\_runner.py – Executes log evaluations  
  log\_analysis\_template.j2 – Contains all prompt variants
* Data:  
  log\_samples.csv – Dataset of 5 synthetic log entries
* Results:  
  analysis\_results.csv – Outputs from all prompt runs
* Documentation:  
  Includes prompt variant descriptions, rubric, refined prompt, evaluation comparisons, and this summary

**Key Takeaways**

* Prompt design dramatically influences LLM performance on cybersecurity tasks.
* Role-based prompts improve contextual accuracy and actionable guidance.
* Iterative testing and rubric-based evaluation are essential for reliable prompt optimization.
* LLMs can be reliable log analysis assistants when guided with clearly structured instructions.