Will Young
Student ID: 924230057
Week 3 Report
2/13/2026

Hello Professor,

Below is an update for my week 3 progress In addition, I have pushed my updates to my github repo: https://github.com/Woyoung21/LastMile-Sec. For week 3, the main tasks were to develop the ingestion pipeline parsers and find a model to be used for the “Mapping” and “Reporter” models.

For the ingestion pipeline, I created two functional parsers utilizing regex for both CSV files and PCAP files. However I struggled with creating a one size fits all parser for PDF files. This does make sense, as most PDF’s are primarily in natural language, and their length and formatting is not uniform, each document is substantially different than the previous one. The goal for all of the parsers was to normalize important data to be shipped off to the Reporter agent for summation. Below is an image of the fields I thought were important to extract from each event in the ingestion pipeline. During my testing, I used both files from old security scans and public data sets of past breaches.Again, for both PCAP and CSV files, we were able to populate the majority of these fields, certainly enough to feed to the Reporter agent and have them create a natural language summation sentence of the event. However the PDF normalization did not go as planned. I attempted a few different revisions of the parser, but because each report was so different and unique, it did not work as planned. Typically the only field that would fill was the raw_excerpt and that was the entire event.

```
{
  "id": "",
  "source_type": "",
  "source_file": "",
  "source_hash": "",
  "timestamp": "",
  "report_date": ,
  "findings": [
    {
      "id": "",
      "severity": "",
      "title": "",
      "description": "",
      "affected_assets": [
        {
          "identifier": "",
          "asset_type": "",
          "details":
        },
        {
```

```
        "identifier": "",
        "asset_type": "",
        "details":
      }
    ],
    "raw_excerpt": "{}",
    "cve_ids": [],
    "cvss_score": ,
    "recommendations": [],
    "references": [],
    "source_ip": "",
    "destination_ip": "",
    "protocol": "",
    "timestamp_observed":
  }
```

This led me to searching for an alternative option to normalize PDF files, which is a very important part of this project. I found a project by Google called LangExtract (https://developers.googleblog.com/introducing-langextract-a-gemini-powered-information-extraction-library/). Whereas the parser we built needs an exact regex match to route to the right JSON categories, LangExtract leverages Google's Gemini to understand what you are looking for based on the input, and generates the structured output you define. It does this by using a “few shots” which are structured example inputs and expected responses. I will test this product out this upcoming week, I just didn’t want to max out my free Gemini API allotment before SFHACKS this weekend. Eric, Bilal & I are participating again and I will be incorporating Gemini into our project.

For the agent selection, we will stick with this model from hugging face as our Reporter agent (https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3). With over a million monthly downloads, this would seem to be a very stable and dependable model for summation. This model will be fed the normalized json packet from our parsers and then create a tight natural language summary of the event it is looking at. It will then pass this onto the Mapping agent.

For the Mapping agent, I will be using this model from hugging face (https://huggingface.co/basel/ATTACK-BERT). It has a very large monthly download metric meaning the community is actively using it. This model takes in the summarization it received from the Reporter agent and creates an embedding vector. We then compare those embedded vectors with our embedded vectors representing MITRE ATT&CK technique descriptions to get a similarity value. We can populate the top 3 matches, compare their values, and select the highest score for our MITRE ID.