Attack Surface Analysis: Real Estate Analyzer

Project Name: Listings

Team: Ramsundhar and Jaiden

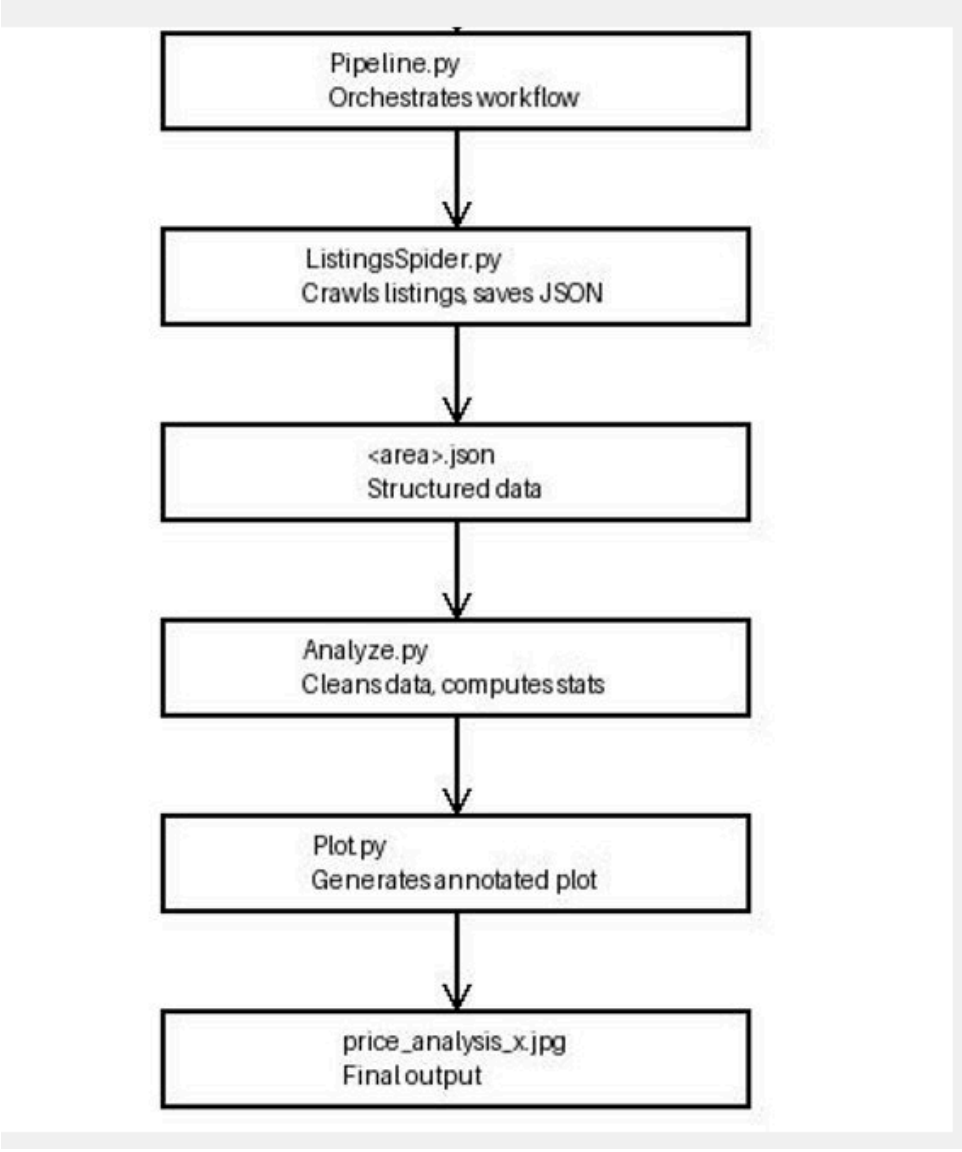
Date: 2025-12-05

Version: v1.0

- **Section 1: Software Architecture Model**
    - Presentation Tier (Client/API Layer)
        - This is the user-facing layer responsible for handling requests, displaying results, and providing the interface (App.py)
        - Receives Requests: Defines the FastAPI endpoint (/run-pipeline/download-plot) to accept the user's input (area).
        - Orchestration Trigger: Calls the next tier (Pipeline.py) to execute the business logic.
        - Delivers Results: Takes the final plot file from the Business Tier and returns it to the user as a downloadable file response (File Response).
    - Business Logic Tier (Application/Service Layer)
        - This is the core layer that coordinates the application's processes, enforces rules, and controls the flow of data (Pipeline.py).
        - Process Manager: Orchestrates the entire data flow (Scrape → Analyze → Plot).
        - Business Rules: Contains the logic for the sequence of operations, including calling external subprocesses (Spider.py, Analyze.py, Plot.py).
        - Helper Components: Analyze.py and Plot.py contain specific analysis and plotting logic, acting as service modules called by the pipeline manager.
    - Data Tier (Data Access/Resource Layer)
        - This layer is responsible for interacting with data sources, which in this case are external websites (scraping) and the file system (temporary JSON storage and final JPEG output)
        - Data Acquisition: Scrapes raw real estate data from the target website (arizonarealestate.com).
        - Data Persistence: Writes the acquired raw data to a temporary file (<area>.json), making the data available to the Business Tier for processing.
- **Section 2: Executive Summary**

- The Real Estate Data Pipeline application is a 3-Tier system that uses a synchronous FastAPI interface to trigger a Python pipeline of subprocesses (including Scrapy) for data extraction, analysis, and plot generation. The most critical vulnerabilities are rooted in its reliance on synchronous blocking calls for the pipeline execution, which leads to a severe Denial of Service (DoS) risk, and the use of un-sanitized user input (area parameter) in subprocess arguments, leading to Command/Path Injection risk. The attack surface primarily consists of the public API endpoint, the subprocess execution boundary, and the temporary file system storage.

- **Section 3: System Architecture Overview**



- Primary components:
    - Client/End User: Initiates the POST request.
    - FastAPI Application (App.py): Presentation Tier - Receives requests and serves the final JPEG file.
    - Pipeline Manager (Pipeline.py): Business Logic Tier - Orchestrates subprocess.run() calls for Scrapy, analysis, and plotting scripts.
    - Scrapy Spider (Spider.py): Data Tier - Handles data acquisition from the external website (arizonarealestate.com).
    - Data Analysis/Plotting Scripts (Analyze.py, Plot.py): Business Logic - Perform data processing and image generation.
    - Local File System: Data Tier - Temporary storage for raw JSON and final JPEG output.
- Data flows:
    - Client → FastAPI (App.py): HTTP POST request with {"area": “tempe"}.
    - FastAPI → Pipeline Manager (Pipeline.py): Synchronous function call run_full_pipeline(“tempe").
    - Pipeline Manager → Scrapy (Spider.py): Subprocess Execution (sends area as command-line argument).
    - Scrapy → Local File System: Writes raw data to tempe.json.
    - Pipeline Manager → Analysis/Plotting: Executes Analyze.py and Plot.py (which read tempe.json and save price_analysis_tempe.jpeg).
    - FastAPI → Client: Returns the price_analysis_tempe.jpeg file as a FileResponse.
    - Pipeline/App → Local File System: Secure deletion of temporary files.
- **Section 4: Asset Data Classification**

Data	**Owner**	**Description**	**Sensitivity**	**Location**
Raw Listing Data (JSON)	Team	Scraped Listings (May include PII like addresses)	Medium	Local Filesystem
Plot (JPEG)	Team	Price Analysis JPEG	Low	Local Filesystem
User Input	DevOps	Area to scrape,	Medium	Pipeline/Code

		rate limits		
Execution Environment	Operations	Scraping, logs, errors, processing	Low	Local Filesystem

- **Section 5: Trust Boundaries**

Boundary ID	**From**	**To**	**Crossing Mechanism**	**Assets Exposed**
1	Public Internet	Spider	HTTP(S) requests	Remote real estate webpage
2	Spider	JSON Storage	Local Filesystem	Raw Scraped JSON files
3	Analyzer/Plot	JPEG Storage	Local Filesystem	Processed stats and plots
4	FastAPI/User	Pipeline	API/Background	Area input access to JSON and plots

- **Section 6: Attack Surface Components**
    - Network
        - FastAPI Endpoint: The public facing POST /run-pipeline/download-plot route.
        - External Traffic: Outbound HTTP requests from Scrapy to https://arizonarealestate.com/*.
        - Server Ports: The Uvicorn listening port (e.g., 8000).
    - Software (Applications & APIs)
        - Subprocess Execution: The subprocess.run() calls in Pipeline.py.
        - Input Validation Logic: The basic if not area_name: check in App.py.
        - File I/O: Reading/writing *.json and *.jpeg files (e.g., file path concatenation).
    - Human (Users & Administrators)
        - End User: Supplying the area input value to the API.

        - System Administrator: Managing the deployment environment and user privileges.
    - Execution environment (OS, containers, middleware)
        - Local File System: The directory where temporary files are created and accessed.
        - Uvicorn/ASGI Server: The web server handling concurrent requests.
    - External dependencies (libraries, APIs, frameworks)
        - Scrapy: Used for web crawling.
        - FastAPI/Pydantic: Used for the API interface.
        - Pandas/Matplotlib: Used for data analysis and plotting.
- **Section 7: Threat Attack Vectors**

Component	**Element**	**Threat/Attack Vector**	**Potential Impact**	**Likelihood**
Spider	Outbound requests	Malicious websites servicing crafted HTML or large payloads	Resource Exhaustion, DOS, and invalid data	Medium-High
Parser	JSON/Matplotlib	Malformed input causing exceptions	Worker crash or pipeline failure	Medium
Storage	JSON/Plots	Accidental Storage of sensitive PII	Privacy or compliance violation	Medium
Human	FastAPI input	Malicious or invalid input	Unauthorized or repeated scraping	Medium
Environment	Python VENV	Vulnerable Python packages	Remote code execution or supply-chain compromise	Low-Medium

Network	Target Sites	IP blocking, captcha challenges	Loss of data coverage	High

- **Section 8: Attack Surface Summary**

Exposure	**Why It Matters**	**Severity (1-5)**	**Remediation Priority**
Outbound fetched to untrusted hosts	Direct interaction with potential attacks	5	High
Raw JSON data storage	May contain PII	4	High
FastAPI endpoints	Can trigger pipeline execution or access stored data	4	High

- **Section 9: Lessons Learned and Future Work**
    - Synchronous I/O is a Security Flaw: The synchronous design where the API thread blocks for the entire pipeline duration creates a severe single point of failure and makes the application vulnerable to basic DoS attacks.
    - External Input Requires Extreme Caution: Relying on un-sanitized user input (area) to construct file names and command arguments is a fundamental security flaw that creates Command/Path Injection vectors.
    - Asynchronous Task Queue: Refactor the system to use a service like Celery or Redis Queue. The API should submit the pipeline job and return a job ID immediately, moving the blocking work out of the web server's main threads, thus mitigating the DoS threat.
    - Robust Input Validation: Implement strict whitelisting for the area parameter, allowing only alphanumeric characters and disallowing any characters that could be used for path traversal (/, ., \) or command injection (;, |, &).
    - Authentication/Authorization: Add an API key or token-based system to secure the /run-pipeline/download-plot endpoint and enable accurate rate limiting per user.
- **Section 10: Revision**
    - v1.0 — 2025-12-4 — Converted example to Real Estate Crawler