Software Design Document (Version 2)

9/24/2025

Team Name: Cyber Recon

Sponsor: HighViz Security LLC

Team Mentor: Karthik Srivathsan Sekar

Team Members: Zachary Garza, Sean Weston, Jared Kagie, Christian Butler

Table of Contents

Introduction

Statistics have shown that 60% of small businesses shut down within six months following a cyberattack. In the US alone, the average cost of a data breach is currently $9.44 million. Many companies lack in-house capabilities to scan and prioritize vulnerabilities since they are understaffed and underfunded. This deficiency has made the majority of companies over-reliant on third-party cybersecurity professionals like HighViz Security LLC.

HighViz possesses the skills to discover and document vulnerabilities with automatic scanners like Nessus. Although these scans detect a wide array of issues, they also produce false positives, they give irrelevant information, and there is just too much information that manual evaluation becomes highly inefficient and time-consuming. Cyber Recon has a solution that utilizes artificial intelligence in automating vulnerability scanning and prioritization.

In the dynamically changing threat landscape of today, the imperative to refresh cybersecurity tactics is clearer than ever. As technology has been evolving at a high speed, cyberattacks have become more sophisticated and unpredictable, exploiting even the smallest vulnerabilities and placing organizations at significant financial and reputational risk. Using artificial intelligence and deep analytics, the proposed solution will translate raw vulnerability information into actionable guidance for faster decision-making and enhanced risk mitigation. This preemptive approach not only enhances the overall security position but also ensures that resources are directed at the most critical issues first.

The incorporation of machine learning and natural language processing in the assessment process is a paradigm shift for vulnerability management. By scanning complex Nessus files, enriching results with up-to-date threat intelligence, and prioritizing risks intelligently, the system enables security teams to work with greater accuracy and efficiency. Its hybrid architecture, designed for local deployment on MacBook Pro M2 hardware and supported with secure cloud collaboration, balances between both cutting-edge technology and practical usability. This approach ensures scalability for organizations of all sizes while maintaining strong guarantees of data privacy.

This design document outlines the architecture of the proposed AI-enabled tool. The solution inputs Nessus files, applies machine learning and natural language processing to analyze vulnerabilities, incorporates external threat feeds, and produces a prioritized list of risks accompanied by a concise report that communicates findings in a way that is accessible to non-technical stakeholders. The focus is on delivering a deployment model that empowers HighViz team members to operate locally while collaborating securely through the cloud. With this foundation in place, the sections that follow define the specific challenges HighViz Security faces and the architecture proposed to address them.

Implementation Overview

Cyber Recon is an artificial intelligence-powered vulnerability analysis and reporting solution that automates the lengthy process of parsing, prioritizing, and presenting vulnerability information. The vision behind the solution is to take raw scan data from scanners such as Nessus and turn it into an actionable, intelligent enhanced report that drives security assessments at higher speeds with less human error. The goal is not to just be more effective, but also to enhance accuracy and prioritization by taking advantage of AI, anomaly detection, and external threat intelligence feeds.

At an architecture level, the implementation makes use of a modular producer-consumer type pipeline. The data is fed in via the input layer, where raw .nessus XML files, CSVs or JSON scan results are processed by a custom NessusParser component. This parser is accompanied by retro hv-doctools libraries provided by HighViz to process scan formats reliably, converting them into structured CSVs and SQLite databases that form the basis for analysis. The data is then piped down to the VulnerabilityProcessor once structured. This is the systems communicator of contact between the machine learning models external API's and report tools.

The intelligence layer is fueled by the AI/ML engine, whose Random Forest classifier that is backed by TF-IDF text vectorization is used to determine the risk levels of vulnerabilities beyond CVSS scores. Aiding it is an Isolation Forest anomaly detector that finds patterns and outliers that may not be caught by normal scoring mechanisms. Both the in-house models are trained locally to protect data privacy and produce their outputs blended into a composite threat score integrating AI-based prediction with conventional factors and real time risk probability. Integration with threat intelligence adds an additional layer of benefit. Cyber Recon uses real-time lookups against the CISA KEV catalog, Exploit Prediction Scoring System (EPSS) and National Vulnerability Database (NVD). These additions enable the system to differentiate between vulnerabilities in the wild and theoretical ones. Results of these services are cached for performance and aggregated into the risk calculation framework.

The last pipeline stage addresses standardized reporting. We use openpyxl libraries and matplotlib visualization libraries, and the system generates Excel reports with comprehensive findings, summary statistics, and charts. CSV and JSON outputs elsewhere allow other platforms to be integrated with, and interactive dashboards using Plotly in the future.

Cyber Recon is written mainly in Python 3.9+, utilizing Pandas and NumPy for data processing, scikit-learn for machine learning functionality, and SQLite for simple storage of parsed scans. Central configuration via a YAML file is used so that teams can quickly adjust model parameters, API endpoints, and process options without altering code. Security is designed into the system naturally by running all AI processing on the local machine, i.e., sensitive vulnerability data never exits the client environment.

Architectural Overview

Cyber Recon uses a modular, pipeline-driven architecture designed to transform raw vulnerability scan data into actionable security intelligence through the integration of artificial intelligence and real-time threat intelligence. The system operates as a series of interconnected processing layers, each responsible for specific data transformation and enrichment while maintaining data privacy through local processing capabilities. The architectural design of the system has five distinct layers in which the vulnerability data flows, being Input Processing, Data Pipeline Management, Threat Intelligence Integration, AI-Integrated Analysis, and Intelligence Reporting. This approach ensures clear separation of concerns while enabling easy and fast data flow from raw Nessus output to concise security reports.

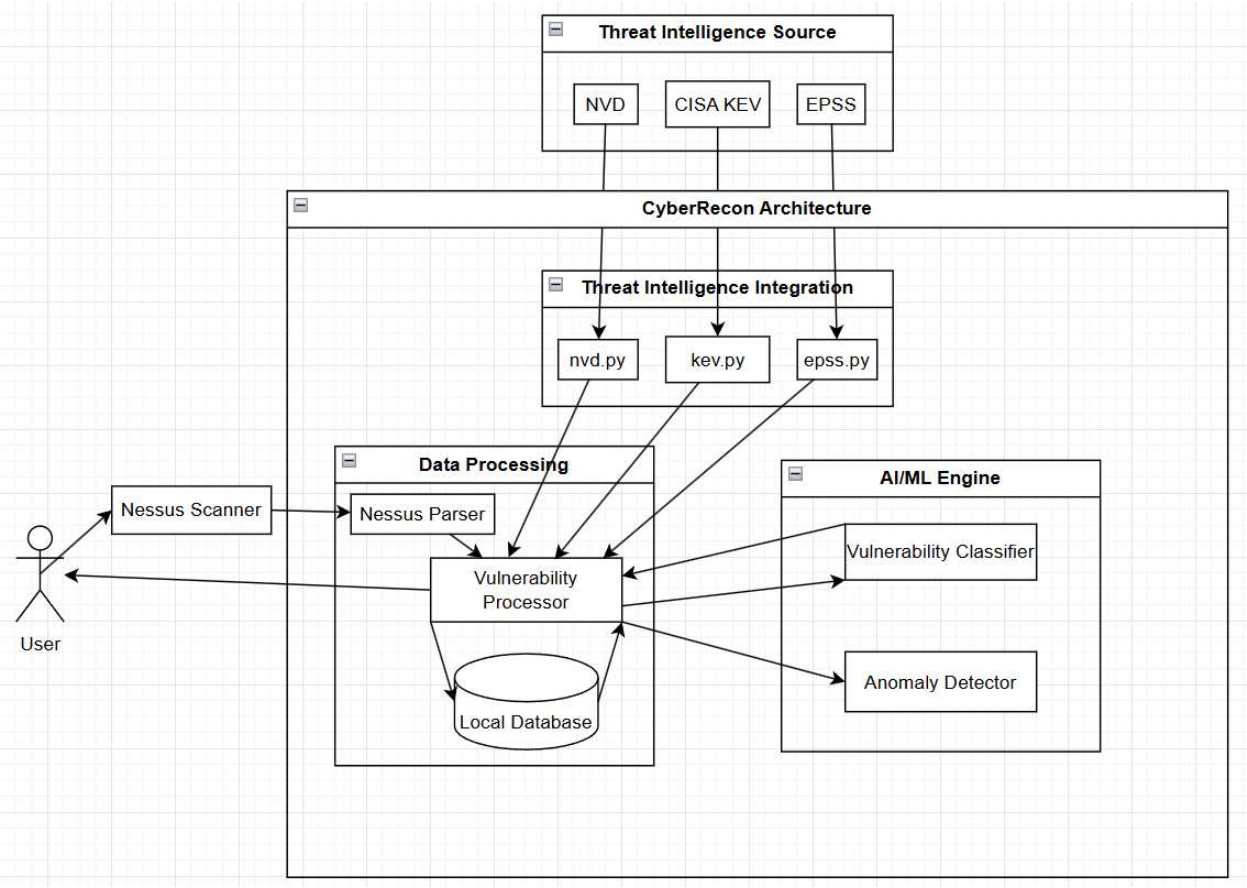

Figure 1: Architectural Diagram of Cyber Recon's system

The Input Layer of the system architecture is the files passed from the user in the form of the Nessus scanner file. The file is taken to the parser inside the Data Processing Pipeline to be turned into a more efficient CSV file to make data handling easier. The focus of the Data Processing is to allow for analyzation of the parsed Nessus data and to be the main flow of information throughout the system. As the model above shows, the Vulnerability Processor is a main source of workflow. The AI and Threat Intelligence are both called upon by the Vulnerability Processor and are used to help generate the reports in various formats such as CSV, Excel and JSON. Lastly, the Data

Processing uses an updater that keeps the data consistent for synchronizations between environments, as well as backing up information.

The Threat Intelligence integration connects the system to multiple sources outside of itself for use of external threat analysis data. Each of the three sources provide information from the National Vulnerability Database, Known Exploited Vulnerabilities, and Exploited Prediction Scoring System. The KEV Client queries CISA's catalog to identify vulnerabilities with confirmed real-world exploitation, while the EPSS Client retrieves probability scores indicating likelihood of future exploitation. The NVD Client provides official CVE metadata and CVSS scoring information. Each client implements caching mechanisms and rate limiting to ensure efficient operation, with all enriched data flowing back to the Vulnerability Processor for integration with internal analysis results.

The AI/ML Engine serves as the analytical intelligence core that processes the structured data received from the Vulnerability Processor. The engine employs two primary analysis components, a Vulnerability Classifier that uses Random Forest modeling with text analysis to predict risk levels beyond standard scoring, and an Anomaly Detector that identifies unusual threat patterns through Isolation Forest algorithms. A Learning System component enables continuous improvement by collecting feedback and retraining models based on real-world results.

Once all components have finished their evaluations, they report back to the Vulnerability Processor for the finishing touches. It combines all analytical data into final composite threat scores. It combines all the AI predictions, threat intelligence indicators, traditional CVSS scores, and anomaly detection results using weighted algorithms that reflect both theoretical severity and practical exploitation likelihood. The Vulnerability Processor then will create one of the three outputs of CSV, Excel or JSON to give back to the user for both ease of understanding as well as clear visual data presentation.

The system's communication flow is characterized by a producer-consumer pipeline pattern where the Vulnerability Processor is the primary source of communication. Data moves unidirectionally from input through parsing, then branches to parallel processing in the AI Engine and Threat Intelligence modules before reconvening for composite scoring and output generation. This centralized coordination ensures data consistency while enabling parallel processing for improved efficiency.

The architecture embodies multiple established patterns working in concert. The layered architecture provides clear separation between processing stages, while the pipeline and filters pattern guides data transformation through sequential enrichment phases. The plugin architecture enables easy addition of new threat intelligence sources. These patterns combine to create a modular, scalable system that maintains data privacy through local processing while leveraging external intelligence sources.

Module and Interface Descriptions

Data Processing Pipeline Module

The Data Processing Pipeline Module is the backbone of the entire system. It is responsible for the ingesting, parsing, and preparing of the data from a provided nessus file. This data will then be passed onto our AI/ML Engine Module, and finally the pipeline will be used to generate a final excel report.

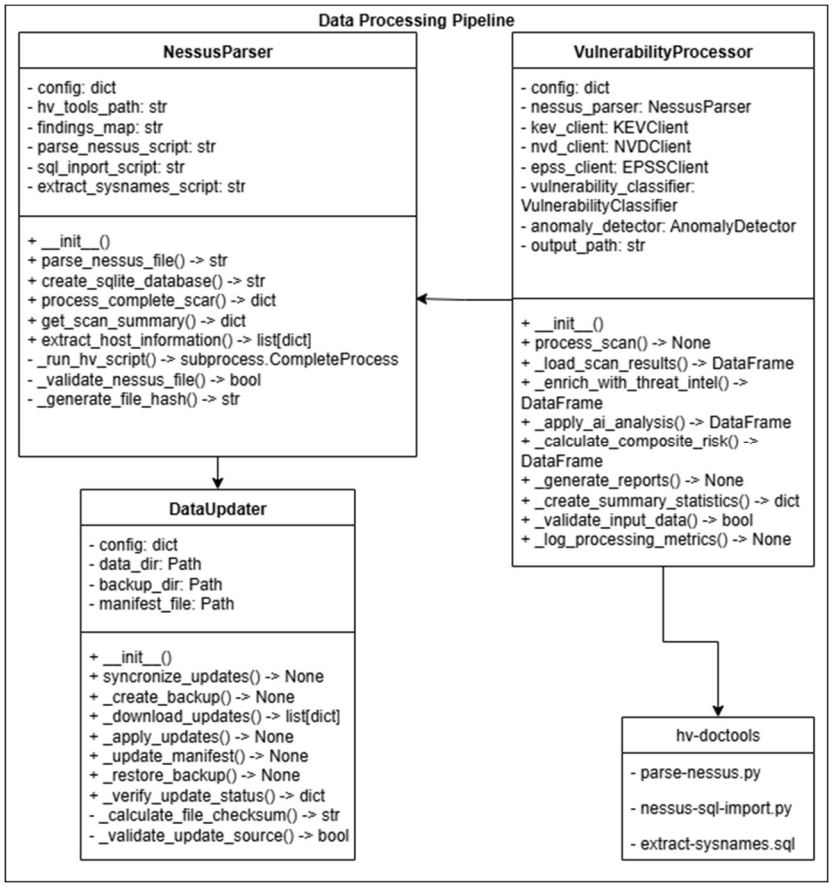

Figure 22: UML Diagram of the Data Processing Pipeline module

Starting with NessusParser, this class is designed to take in a nessus file and parse it into a CSV format to make it easier to use. When initialized, it can be provided a configuration file that provides various parameters, such as hv_tools_path which contains the path to the hv-doctools directory. Parse_nessus_file takes a nessus file as input and returns a path to the CSV file. Create_sqlite_database converts a CSV file, generated from parse_nessus_file, into an SQLite database to allow for more easily accessed data, as well as returning a path to the database. Process_complete_scan combines the previous methods, turning the nessus file into a CSV, then into an SQLite database, ultimately returning a dictionary containing a path to the CSV file, SQLite DB, and scan summary. Lastly, get_scan_summary extracts the scan's metadata without processing the data, returning start and end timestamps, number of scanned hosts, number of vulnerabilities, and the count by severity level.

VulnerabilityProcessor is used for the vulnerability analysis workflow from the input of the data to the output. It handles parsing, enriching, and analyzing the data, coordinating the AI/ML engine and the threat intelligence APIs. It also handles the creation of standardized reports in various formats, such as Excel, CSV, and JSON. When initializing the class, a configuration file is provided which includes the directory for the trained AI models, an anomaly detection sensitivity metric, or the config for the API. Process_scan takes the paths to the nessus file and output report, then runs through the vulnerability processing pipeline. This includes parsing the input file via NessusParser, loading/validating the data, enriching it with threat intelligence, applying the AI/ML analysis, calculating the composite risk scores, and then generating the final reports.

DataUpdater ensures that all threat intelligence is synchronized so that all data analysis is accurate between devices. Additionally, it handles creating and restoring backups of the data to ensure data is always available. When initialized, a configuration file is provided which contains the directory for the main data storage, as well as the list of threat intelligence sources. Synchronize_updates executes the data synchronization process, which involves creating a backup of the current data, downloading and verifying updates, updating the manifest with the new version, and restoring the backup if a failure occurs.

AI/ML Engine

The AI/ML Engine is the main meat of the system. After the data is parsed properly, the AI/ML Engine determines the vulnerability risk scores and anomaly detection. This helps determine what threats should be considered and what threats aren't a priority. All of this is completed locally to ensure total security for their client's data.

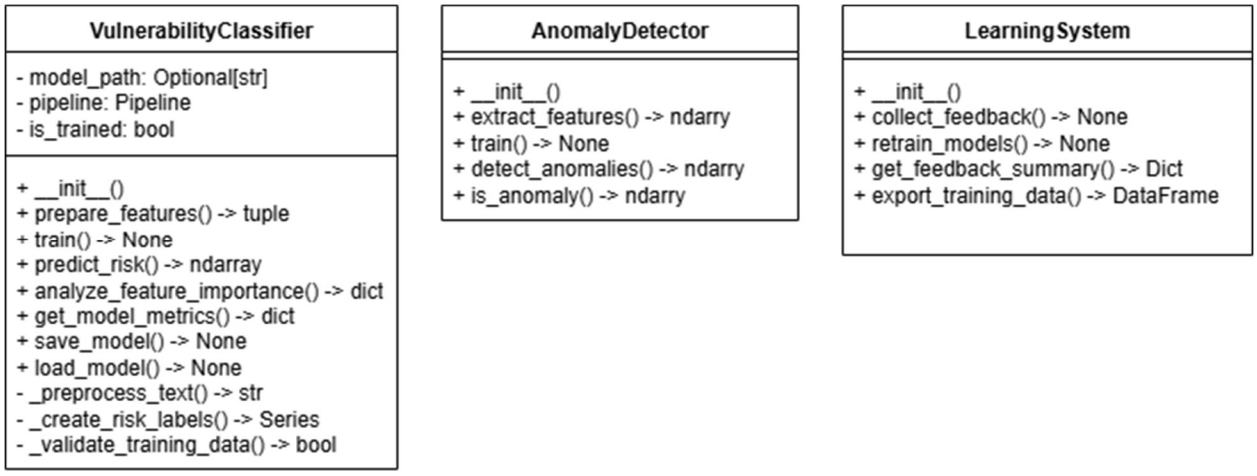

Figure 33: UML Diagram of the AI/ML Engine Module

When initializing VulnerabilityClassifier, the path to an optional pre-trained model can be provided. The ML pipeline that will be used by the class will include TfidVectorizer() and RandomForestClassifier(). Prepare_features is used to extract and prepare features from the provided vulnerability data when inputted with the vulnerability data that was found. After being used, it returns a feature matrix and the target labels. Predict_risk, when provided with vulnerability data, will create the risk probability scores for each of the vulnerabilities. What is returned is an array of risk scores for the vulnerabilities provided. Train, save_model, and load_model allow for the configuration of the model to be used during the vulnerability risk analysis, while get_mode_metrics allows the various aspects of the specified model to be inspected.

AnomalyDetector's main responsibility is to identify and report anomalies in the provided data. Specifically, it is designed to detect rare/novel threats that may have been missed by standard scoring.extract_features returns the numberical features of the data being analyzed. Train will train the isolation forest model specifically for finding anomalies in the vulnerability data. Detect_anomalies will return all of the anomaly scores, while is_anomaly will return return a boolean array of anomaly classification.

LearningSystem allows the model to continuously improve based on real-world feedback. This feedback will come from employees when new data may need to be introduced to the system, so that the model can be trained on potentially unseen data. Collect_feedback willl store any feedback given by testers. Retrain_models will retrain the models based on the collected feedback, ensuring the model has access to the newest information provided. Get_feedback_summary allows for any statistics on the feedback submitted to be seen, and export_training_data allows all feedback to be collected so that it may be analyzed.

Threat Intelligence Integration

The Threat Intelligence Integration Module is responsible for enriching vulnerability data with up-to-date external threat intelligence feeds. While the Data Processing Pipeline ensures that raw scanner outputs are normalized and structured, this module provides the critical context that transforms those outputs from static lists into real-world diagnosed security intelligence. Its primary role is to assess whether a vulnerability is actively being exploited, how likely it is to be weaponized, and whether it has significant impact to the real world.

Within the Cyber Recon architecture, this module acts as a bridge between internally generated findings and the wider cybersecurity ecosystem. The AI/ML Engine produces estimated risk scores based on training and statistical analysis, but without external enrichment, these scores could overlook emerging threats. By integrating authoritative sources like the CISA Known Exploited Vulnerabilities (KEV) catalog, the Exploit Prediction Scoring System (EPSS), and the National Vulnerability Database (NVD), the Threat Intelligence Module adds dynamic and evidence-based context to each vulnerability record.

The importance of this module is obvious; it ensures that the system does not merely prioritize based on "theoretical" severity, like the CVSS scores, but also incorporates live exploitation data. For example, a medium-severity vulnerability might be assigned high urgency if KEV lists it as actively exploited, while a critical vulnerability without known exploits may be downgraded in priority. In this way, the module ensures security teams are allocating their resources to the issues that matter most right now.

Additionally, the module supports caching and rate-limiting functionality. Since external lookups can be slow or limited by API restrictions, results are cached locally in structured formats for quick reuse. This live querying makes the module reliable, even when handling large vulnerability datasets.

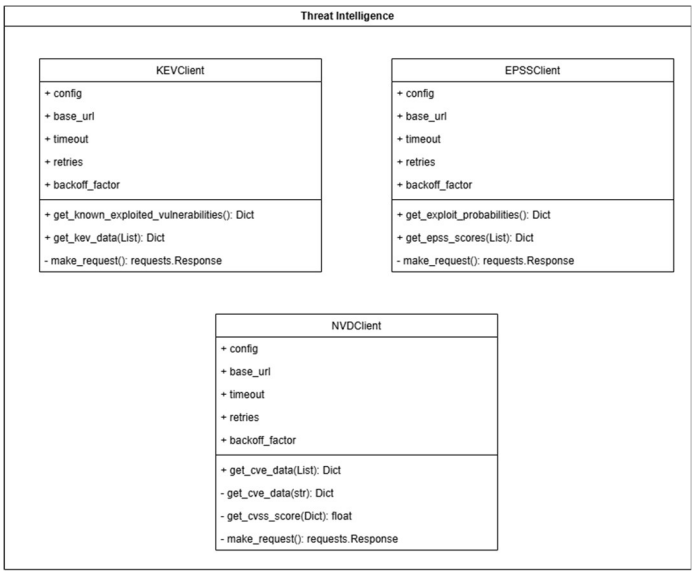

Figure 44: UML Diagram of the Threat Intelligence module

The Threat Intelligence Integration Module contains three classes, KEVClient, EPSSClient, and NVDClient that each provide focused API for its respective threat intelligence source. Together, they form the enrichment backbone of our solution, offering standardized methods for querying vulnerabilities, retrieving enrichment data, and caching results for performance.

KEVClient (kev.py)

Responsibilities: Interfaces with the CISA KEV catalog to identify vulnerabilities with confirmed exploitation. Provides details such as exploitation timelines and remediation deadlines. Implements local caching and retry logic to minimize API overhead.

Key Public Methods:

- __init__(config: Optional[Dict] = None), Initializes the client with API and caching configuration.
- get_known_exploited_vulnerabilities() -> pd.DataFrame, Retrieves the full KEV catalog and returns it as a Pandas DataFrame for downstream use.
- get_kev_data(cves: List[str]) -> Dict[str, bool], Accepts a list of CVE IDs and returns a dictionary mapping each CVE to a Boolean indicating whether it is listed in KEV.

EPSSClient (epss.py)

Responsibilities: Connects to the Exploit Prediction Scoring System (EPSS) to obtain exploitation probability scores and percentile rankings for CVEs. Designed to efficiently handle large CSV downloads and update scoring with the latest daily threat intelligence.

Key Public Methods:

- __init__(config: Optional[Dict] = None), Initializes the client with API and caching config options.
- get_exploit_probabilities(cves: List[str]) -> Dict[str, float], Returns a dictionary mapping CVEs to their probability of exploitation (0–1 scale).
- get_epss_scores(cves: List[str]) -> Dict[str, Dict], Provides a richer result set, including both probability and percentile ranking for each CVE.
- download_epss_data() -> pd.DataFrame, Downloads and parses daily EPSS dataset, stored as a Pandas DataFrame for reuse.

NVDClient (nvd.py)

Responsibilities: Interfaces with the National Vulnerability Database (NVD) to retrieve official CVE metadata, CVSS scores, and impact classifications. Provides support for CVSS (v3.1, v3.0, v2.0) **and** includes rate-limiting logic to comply with NVD API usage.

Key Public Methods:

- __init__(config: Optional[Dict] = None), Initializes client with API credentials, rate-limiting policies, and caching options.
- get_cve_data(cves: List[str]) -> Dict[str, Dict], Accepts a list of CVEs and returns a dictionary mapping each CVE to its associated metadata and CVSS scores.
- get_cvss_score(cve_item: Dict) -> float, Extracts and returns the CVSS score from an NVD API response object.

Command-Line Tools

The Command-Line Tools and Utilities Module is the primary user-facing module of the Cyber Recon system. While the data processing, AI/ML Engine, and Threat Intelligence Integration modules provide the core functionality, the command-line tools serve as the orchestrators that expose that functionality in a consumable way. The tools allow security analysts and developers to interact with the system in a consistent, scriptable and replicable way. By exposing a uniform interface, the tools are straightforward to integrate into existing workflows, automated pipelines, or test setups.

This module provides three specific command-line tools. The first is process_nessus_scan.py, which runs the complete end-to-end vulnerability analysis workflow. This includes parsing raw .nessus XML scan data, checking its integrity, enriching it with threat intelligence feeds (KEV, EPSS, NVD), applying AI-based classification and anomaly detection, calculating composite risk scores, and generating reports in several formats such as Excel, CSV, and JSON. This script is the largest of the three tools, constructed for full production use.

The second utility is parse_nessus.py, which is a lightweight parser for situations where a complete analysis is not required. It performs XML-to-CSV and SQLite conversions so that users can readily obtain structured Nessus scan results. The tool is useful for preprocessing data or for analysts to feed outputs to other security tools without the full workflow.

The third use case is main.py, the primary entry point of the system. It provides three execution modes: process, which runs the full pipeline with AI and threat enrichment; update, which synchronizes local data with the latest threat intelligence feeds; and classify, which applies the AI classifier to pre-parsed datasets without performing parsing or enrichment. By integrating these options, main.py reduces user interaction, eliminates redundancies and allows consistent configuration management across workflows.

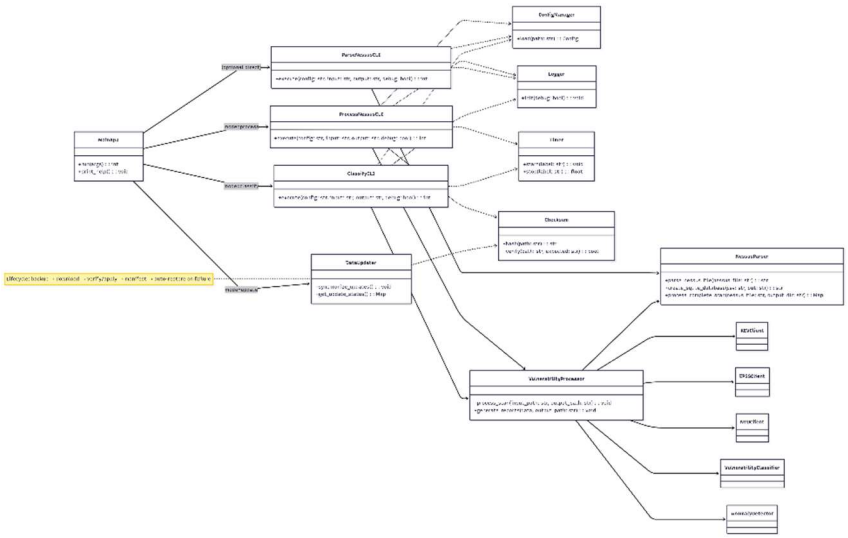

Figure 5: UML Diagram of the Command Line module

The command-line tools module offers a consistent interface on three modes of execution: process, update, and classify. These are invoked by the main program and share a common argument list: --config (YAML file), --input (source file such as a .nessus scan or pre-processed CSV), --output (result directory or file), and --debug (verbose logging). This uniform design ensures users can include Cyber Recon in scripts, automation pipelines, or manual workflows with minimal modifications.

In process mode, the software executes the full pipeline: it ingests raw .nessus XML, checks the data for consistency, enriches results with KEV/EPSS/NVD knowledge, applies AI-driven classification and anomaly detection, computes a composite risk score, and generates reports in Excel, CSV, and JSON formats. The update mode focuses on maintaining up-to-date threat intelligence, using the DataUpdater with checksum and backup verification to ensure data integrity even if a source fails. Meanwhile, classify mode streamlines experimentation by applying AI models directly to pre-parsed datasets, allowing fast re-scoring or model evaluation without re-running the entire pipeline.

Collectively, these services articulate the module's public promise: to allow users to process scans from beginning to end, keep their intelligence feeds current, or quickly apply AI-based risk scoring. By using explicit arguments, deterministic results, and built-in checks, the interface allows analysts to confidently rely on Cyber Recon for production pipelines and research probes.

Implementation Plan

The implementation timeline extends across the semester; however, for clarity and because this document was written towards mid-semester, it is presented beginning at Week 4.

Phase 1: Environment and Foundations (Week 4)

We will first begin by creating the environment for our software solution. Concurrently, the Data Processing Pipeline and Artificial Intelligence/Machine Learning Engine will begin to be constructed. Since these three are the main backbones of our solution, these sections will benefit from being created first.

Phase 2: Other Module Development (Week 5 - 7)

After the main backbone is constructed for our software solution, other modules are to be developed on top of this. The Public Interface module needs to have the Data Processing module finished before development, explaining its placement on the Gantt Chart (fig 3). This is also why the Threat Intelligence module is being developed after the Data Processing module as well. While the testing and development of these are going on, the implementation and testing of the Artificial Intelligence/Machine Learning Engine is still taking place.

Phase 3: Integration and System Finalization (Week 8 - 9)

Due to all the individual modules being developed, final integration methods can now be assigned to the team members to create a cohesive product. The software solution requires lots of testing, since all these parts need to coexist and work together in tandem. Implementation will occur at the same time as the final testing of the command-line tools, as they will be finished around the same time.

Phase 4: Final Testing, Polish, and Documentation (Week 10+)

Our software solution needs to be done by Week 9, per class instructions and guidelines. Thus, this allows further final polish for the client as well as final presentations and user manual creation. This leaves system testing to be finished around Week 10, and final polish lasting throughout the rest of the semester. By Week 15, the codebase will be finished and ready to deploy.

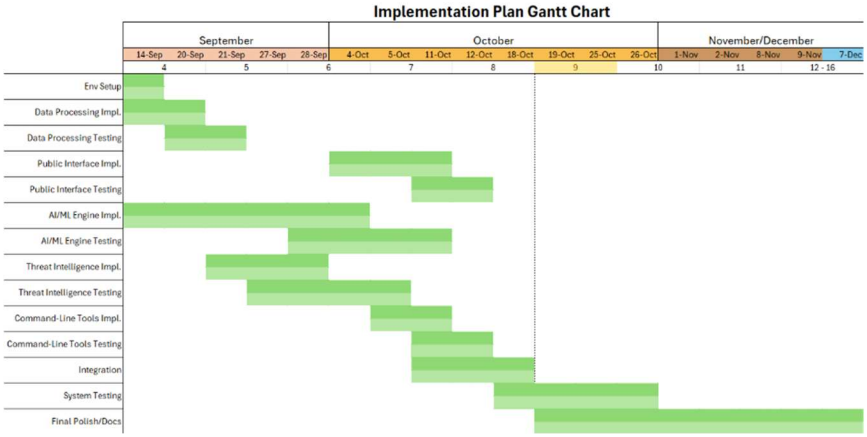

Figure 66: Gantt Chart showing Cyber Recon's Implementation Plan

For this project, the work is distributed among the team as follows: Zach will write the initial environment setup and the integration of the command-line tools, Christian will write the data processing implementation and will also be responsible for system integration, Sean will write the AI/ML engine implementation, and Jared will write the threat intelligence and be responsible for the final polish and documentation. All test phases, from data processing, AI/ML engine, threat intelligence, command-line tools, up to full system testing, will be a shared responsibility across the entire team to ensure thorough coverage and overall quality assurance.

Conclusion

Overall, the main objective of this document is to illustrate the impacts, purposes, and planning decisions in developing our software solution. By stepping through the architectural overview, the module descriptions, and the staged implementation plan, we have shown how Cyber Recon is designed to transform raw vulnerability data into actionable intelligence for HighViz Security LLC and their clients. Each design decision was made with both practical and strategic considerations in mind. This helps balance local execution, secure collaboration, and the integration of machine learning and natural language processing.

The architectural structure emphasizes clarity and modularity, ensuring that each component works both independently and as part of a larger, cohesive whole. This modularity not only simplifies testing and debugging but also allows team members to contribute in an efficient and meaningful manner without sacrificing quality. The implementation plan further demonstrates how these modules will come together over the next weeks, highlighting the approach of building the backbone, developing features, and performing final integration.

From a broader perspective, the importance of this project extends beyond its technical details. Cyber Recon directly addresses a growing need in the cybersecurity field. By automating tedious manual processes and enhancing them with contextual threat intelligence, our solution allows HighViz Security to help clients better secure their domains, while enabling HighViz to more efficiently assess their clients.

In short, the combination of robust architecture, careful planning, and team collaboration makes our solution an asset for improving vulnerability identification and management. As we move into the implementation and testing phases, we are confident that the project will not only meet its immediate goals but will also provide lasting value to HighViz and their clients. With this design as our foundation, we are optimistic about the project’s contribution towards AI-driven solutions for cybersecurity management.

Appendix: Figures