Project Development Manual

Project Title: AI-Based Threat Intelligence Platform

Use Case:

Develop a backend-driven platform that collects, processes, and analyzes threat intelligence data using AI/ML techniques to deliver actionable insights to cybersecurity teams.

Milestone 1: Project Initiation and Planning

Objectives:

- Define project scope and key deliverables.
- Identify stakeholders and assign responsibilities.
- Prepare a development roadmap with timelines.
- Establish collaboration methods within the team.

Subtopics:

1. Defining Project Scope and Objectives:

- Purpose: To develop a backend system for collecting, analyzing, and interpreting cyber threat data.
- Key Goals:
  - Real-time threat data aggregation.
  - Intelligent threat pattern detection using ML.
  - Delivery of insights in a structured backend format.

2. Stakeholder Identification and Roles:

- Stakeholders:
  - Developers – system architecture and backend logic.
  - ML Engineers – model training and testing.
  - Data Engineers – data ingestion and preprocessing.
- Defined Roles:
  - Backend Developer – API integration, server-side logic.
  - Data Engineer – data sourcing and cleaning.
  - ML Specialist – model selection, training, and evaluation.

3. Project Plan Development:

- Break project into phases: Data collection → Preprocessing → Analysis → Output.
- Allocate time for research, coding, and testing.
- Assign tasks using a Kanban board or task tracker.

4. Communication Plan:

- Tools used: GitHub (code collaboration), WhatsApp (daily check-ins), Google Docs (documentation).
- Weekly status meetings to review progress and solve blockers.

Milestone 2: Data Collection and Integration

Objectives:

- Identify relevant data sources (open-source, CVE databases).
- Build scripts and mechanisms to collect data.
- Preprocess and format data for model compatibility.

Subtopics:

1. Source Identification and Selection:

- Selected Sources:
  - MITRE ATT&CK, CVE Details, public threat feeds (e.g., AlienVault OTX).
- Data types:
  - Indicators of Compromise (IP, Hashes, Domains).
  - Known vulnerabilities and exploit metadata.

2. Data Collection Mechanisms:

- Implemented automated data retrieval using APIs and Python scripts.
- Scheduled scripts using cron jobs for continuous updates.
- Logged raw data for audit and debugging purposes.

3. Data Preprocessing:

- Standardized formats into structured JSON/CSV.
- Removed noise, duplicate entries, and irrelevant fields.

- Normalized naming conventions for consistency.

Milestone 3: Threat Analysis and Insights Generation

Objectives:

- Apply machine learning models to detect anomalies and predict threat trends.
- Analyze historical data to identify emerging threats.
- Convert analytical output into structured backend results.

Subtopics:

1. Machine Learning Model Development:

- Used models like K-Means Clustering, Isolation Forest (for anomaly detection).
- Input data: Preprocessed threat logs and known incident reports.
- Model evaluation based on accuracy, recall, and detection rate.

2. Emerging Threat Detection:

- Focused on pattern deviations and correlation across threat sources.
- Designed for adaptability using retrainable models.
- No human labeling required (unsupervised learning approach).

3. Backend Insight Generation:

- Final insights generated as structured JSON output.
- Insights included:
  - Likely affected systems
  - Threat severity score
  - Suggested mitigation paths
- No front-end; results are stored in local/hosted backend storage.

Milestone 4: Testing, Deployment, and Maintenance

Objectives:

- Validate the functionality and accuracy of the backend system.
- Deploy system on local or cloud-based environment.

- Plan for regular updates and performance monitoring.

Subtopics:

1. Testing and Quality Assurance:

- Unit tests for data fetching and preprocessing.
- Manual validation of ML model predictions using test datasets.
- Verified accuracy and false positive rates of the threat alerts.

2. Deployment Strategy:

- Deployed project in a controlled test environment (e.g., XAMPP or Python Flask).
- Local server used to simulate API integration and storage.
- Ensured data backups and model reproducibility.

3. Maintenance and Updates:

- Documented process for adding new data sources.
- Regular review of model performance.
- Plan to update threat sources and retrain models quarterly.