**Software Architecture**

**Problem Domain**

This project aims to provide a centralized Security Operations Center web platform that allows analysts to monitor, analyze, and manage cybersecurity threats efficiently.

Today, SOC analysts rely on multiple separate tools for tasks such as CVE tracking, IDS log analysis, phishing detection, blacklist management, and attack simulations, which leads to data fragmentation, time loss, and operational inefficiency.

To address this, the system integrates:

* Automated data collection (from sources like NVD, ExploitDB),
* AI-powered threat detection and anomaly analysis,
* Real-time dashboards and IDS integration,
* Phishing, malware, and password strength analysis,
* Cryptography and reporting tools.

The goal is to enable SOC analysts to perform faster, unified, and more effective security operations through a single integrated platform.

**Objective Criteria**

**Performance:** The system must process real-time IDS data, anomaly detections, and dashboard updates with minimal latency and high throughput.

**Scalability (Horizontal & Vertical):** Each module should scale independently to handle growing data and user loads, supporting both vertical and horizontal scaling.

**Integration Support:** Technologies must support easy integration with external sources (NVD, ExploitDB, Caldera) and APIs for ML/IDS modules.

**Cost:** Preference will be given to open-source, low-cost, and easily maintainable technologies to minimize infrastructure and operational expenses.

**Project Architecture Overview**  
The system is a modular, secure, and high-performance SOC platform designed for real-time monitoring, analysis, and reporting. It consists of several integrated layers:

* **Frontend**: Built with React, providing a dynamic and responsive interface for SOC analysts. Users can view dashboards, analyze IDS logs, monitor CVE updates, manage blacklisted IPs, and perform system actions in real time. It communicates with the backend via RESTful APIs and WebSockets (Django Channels) for live updates.
* **Backend**: Implemented using Django and Django REST Framework, the backend handles API requests, data flow, and business logic. Functionalities are organized into independent apps for modularity, maintainability, and extensibility.
* **Model Serving / ML**: Machine learning inference is evaluated across three approaches—Django internal inference, FastAPI microservice, or a dedicated model-serving setup—with the most efficient approach selected based on performance.
* **Database**: PostgreSQL provides high-performance data storage with strong indexing and native JSONB support, allowing efficient handling of complex security data.
* **Asynchronous Processing**: Celery with Redis handles background tasks such as CVE data fetching, periodic domain scans, report generation, and large IDS dataset processing.
* **Real-Time Communication:** Django Channels with Redis enables instant notifications for anomalies, blocked IPs, and updated metrics, ensuring immediate feedback to users.
* **Security & Authentication:** Security is integrated throughout, employing session-based authentication combined with two-factor authentication (2FA) to safeguard user accounts and system access.
* **Reporting & Monitoring:** This layer generates and presents system health and security reports, including PDF exports summarizing domain security posture, attack history, and detected vulnerabilities.
* **Deployment & Testing:** The system supports Dockerized deployment for consistent environments, AWS cloud deployment for scalability and production use, and local testing setups for development and validation.