# **Software Requirement Specification (SRS)**

## **Cloud-Based Threat Intelligence Dashboard**

## **1. Introduction**

### **1.1 Purpose**

The purpose of this SRS document is to define the comprehensive requirements for developing a Cloud-Based Threat Intelligence Dashboard. This platform is designed to aggregate, analyze, and visualize real-time threat intelligence data from multiple external sources. The system aims to provide cybersecurity analysts and IT security teams with actionable insights and alerts to enhance their threat detection and response capabilities.

### **1.2 Scope**

The Cloud-Based Threat Intelligence Dashboard will:

* Aggregate cyber threat data from external APIs like VirusTotal, AlienVault OTX, Shodan, and AbuseIPDB.
* Store and organize threat intelligence data using MongoDB Atlas for scalable NoSQL storage.
* Visualize threat trends, attack sources, and malicious indicators through an interactive and real-time dashboard built with React.js.
* Provide alerts on trending threats and anomalies using basic heuristics and pattern matching.
* Deploy on a cost-effective cloud platform (preferably GCP for its generous free tier) to ensure scalability and high availability.
* Serve cybersecurity analysts and IT security teams to enhance proactive threat monitoring.

### **1.3 Definitions, Acronyms, and Abbreviations**

* **API**: Application Programming Interface
* **GCP**: Google Cloud Platform
* **AWS**: Amazon Web Services
* **IP**: Internet Protocol
* **NoSQL**: Non-relational database
* **CI/CD**: Continuous Integration and Continuous Deployment
* **MongoDB Atlas**: Managed cloud database service
* **Heuristics**: Rules or methods for data analysis without complex algorithms or ML models

### **1.4 References**

* [VirusTotal API Documentation](https://developers.virustotal.com/)
* [AlienVault OTX API](https://otx.alienvault.com/api)
* [Shodan API](https://developer.shodan.io/)
* [AbuseIPDB API](https://www.abuseipdb.com/api)
* [MongoDB Atlas Documentation](https://docs.atlas.mongodb.com/)

### **1.5 Overview**

This document outlines the system features, functional and non-functional requirements, design constraints, external interfaces, and planned system architecture for the Threat Intelligence Dashboard. It serves as a guide for developers, testers, and stakeholders involved in the project.

## **2. Overall Description**

### **2.1 Product Perspective**

The Threat Intelligence Dashboard is a cloud-native application designed to provide real-time cyber threat insights. It acts as an aggregation platform that collects data from multiple external threat intelligence sources, normalizes it, and visualizes it for easy analysis. This project is designed to be lightweight and cost-effective, suitable for deployment on cloud platforms such as GCP, AWS, or Azure.

It is an independent system but can be integrated with existing Security Information and Event Management (SIEM) tools or threat management platforms using RESTful APIs. The application consists of the following major components:

* **Frontend**: React.js for an interactive and responsive user interface.
* **Backend**: Node.js with Express framework for RESTful APIs and data processing.
* **Database**: MongoDB Atlas for flexible and scalable NoSQL storage.
* **Cloud Platform**: GCP (preferred for its cost-effectiveness), AWS, or Azure for deployment.

### **2.2 Product Functions**

* **Threat Feed Aggregation**: Collects real-time threat data from multiple external APIs, normalizes the data, and stores it in MongoDB Atlas.
* **Threat Database**: Organizes and securely stores threat intelligence data for efficient retrieval and analysis.
* **Real-Time Dashboard**: Visualizes threat trends, geographical attack sources, and malicious indicators with interactive charts and maps.
* **Alert System**: Generates and notifies users about potential threats and anomalies based on predefined rules and heuristics.
* **Basic Threat Analysis**: Utilizes simple heuristics and pattern matching for anomaly detection without complex ML models.
* **Cloud-Native Deployment**: Leverages cloud platforms for scalable, secure, and highly available deployment.

### **2.3 User Classes and Characteristics**

* **Cybersecurity Analysts**: Experienced users who analyze threat data, investigate anomalies, and respond to alerts.
* **IT Security Teams**: Monitor threat intelligence in real-time and investigate suspicious activity.
* **System Administrators**: Manage user roles, access controls, and maintain system security and uptime.

### **2.4 Operating Environment**

* **Frontend**: React.js (hosted on Firebase Hosting, S3, or Azure Static Web Apps).
* **Backend**: Node.js with Express (deployed on GCP Cloud Run, AWS Lambda, or Azure Functions).
* **Database**: MongoDB Atlas (managed NoSQL cloud database).
* **Cloud Platforms**: GCP (preferred for free tier), AWS, or Azure for scalability.

### **2.5 Design and Implementation Constraints**

* Cloud-native design to minimize infrastructure management and operational costs.
* Use MongoDB Atlas for database due to its scalability and cloud integration.
* Cost-effective solution leveraging free-tier cloud resources where possible.
* Security compliance with HTTPS communication, role-based access control, and input validation.

### **2.6 User Documentation**

* **User Guide**: Navigation and usage instructions for the dashboard.
* **Developer Documentation**: API documentation, deployment guide, and architecture overview.
* **Technical Support Documentation**: Troubleshooting common issues and FAQs.

## **3. Functional Requirements**

* **Threat Feed Aggregation**: The system shall collect threat intelligence data from external sources such as VirusTotal, AlienVault OTX, Shodan, and AbuseIPDB using their respective APIs.
* **Data Normalization**: The system shall normalize threat data to a common schema before storage.
* **Threat Storage**: The system shall store threat intelligence data securely in MongoDB Atlas.
* **Real-Time Dashboard**: The system shall display real-time threat data, trends, and visualizations on a user-friendly dashboard.
* **Alert Generation**: The system shall generate alerts based on predefined rules and heuristics for detecting anomalies.
* **User Authentication and Authorization**: The system shall require users to authenticate before accessing the dashboard, ensuring role-based access control.
* **Data Refresh and Sync**: The system shall periodically refresh and sync threat intelligence data from external sources.
* **Export Reports**: The system shall allow users to export threat intelligence reports in PDF or CSV formats.

## **4. Non-Functional Requirements**

* **Performance**: The system shall provide real-time updates with a maximum delay of 5 seconds for new threat data.
* **Scalability**: The system shall be scalable to handle increasing amounts of threat data and user requests.
* **Security**: The system shall use HTTPS for data transmission and implement role-based access control.
* **Usability**: The dashboard shall have an intuitive UI with responsive design for compatibility with desktops, tablets, and mobile devices.
* **Availability**: The system shall maintain 99.9% uptime by leveraging cloud services with auto-scaling and redundancy.
* **Maintainability**: The codebase shall be modular and documented for easy maintenance and future updates.
* **Compliance**: The system shall comply with data protection regulations such as GDPR for user data privacy.

## **7. Conclusion**

The Cloud-Based Threat Intelligence Dashboard is designed to provide real-time threat intelligence aggregation, analysis, and visualization. By leveraging cloud-native services and cost-effective deployment strategies, the platform delivers scalable, high-performance threat monitoring without the complexity of advanced AI models. This document provides a comprehensive outline of the system’s features, requirements, and design architecture, ensuring a structured development process. The dashboard is poised to be an essential tool for cybersecurity analysts and IT security teams, enhancing proactive threat detection and incident response.

This SRS serves as a definitive guide for developers, stakeholders, and testers, ensuring alignment throughout the project lifecycle. With its modular architecture and cloud-native deployment, the system is flexible enough for future enhancements and integrations. The project is positioned to offer valuable insights into emerging cyber threats while maintaining a cost-effective and scalable solution.