NOC/CSOC: A Web-Based Video Stream Monitoring System for Network and Cybersecurity Operations Centers

Aayan Ansari, Gursneh Kaur, Nakul Ranka, Mohammad Rashid Pathan, Vedika Agrawal, Vinay Lunawat
Department of Computer Science and Engineering
Indian Institute of Technology Gandhinagar
Gandhinagar, Gujarat, India
Email: {aayan.ansari, gursneh.kaur, nakul.ranka, mohammadrashid.pathan, vedika.agrawal, vinay.lunawat}@iitgn.ac.in

***Abstract*—The NOC/CSOC web application enables real-time monitoring of video streams for Network and Cybersecurity Operations Centers, providing configurable dashboards for surveillance. Developed using Node.js, Express, and Node-Media-Server (NMS), it employs HTTP Live Streaming (HLS) with FFmpeg and integrates Elasticsearch for analytics visualized via Kibana and Grafana. JSON Web Tokens (JWT) ensure secure authentication with an admin-client approval mechanism. The current system supports a six-frame dashboard displaying a single stream across all frames. Exploratory efforts to integrate Firebase authentication, WebRTC, MediaSoup, Splunk, stream recording, and machine learning-based threat detection were attempted but not fully realized due to technical constraints. This paper details the system's architecture, implementation, challenges, and potential for NOC/CSOC applications.**

***Index Terms*—Video Streaming, Network Operations Center, Cybersecurity Operations Center, Node-Media-Server, HTTP Live Streaming, Elasticsearch, JSON Web Tokens, Kibana, Grafana, Real-Time Monitoring**

I. Introduction

Network Operations Centers (NOCs) and Cybersecurity Operations Centers (CSOCs) rely on advanced tools to monitor infrastructure and detect anomalies in real time. Video surveillance systems, inspired by platforms like Grafana and Kibana, are essential for visualizing operational and security data. The NOC/CSOC project delivers a web-based application for configurable, multi-stream video monitoring tailored to these environments.

This paper presents the NOC/CSOC system, which uses Node-Media-Server (NMS) for HLS-based streaming, FFmpeg for video processing, and Elasticsearch for analytics, with dashboards powered by Kibana and Grafana. Authentication is secured using JSON Web Tokens (JWT) with an admin approval mechanism. Currently, the system displays a single stream across six frames, with efforts ongoing to support independent streams. We discuss the system's design, implementation challenges, and lessons from unsuccessful integration attempts, including Firebase, WebRTC, and machine learning-based threat detection.

II. System Architecture

The NOC/CSOC system is a full-stack web application comprising backend, frontend, and external services for streaming and analytics.

A. Backend

Built with Node.js and Express, the backend manages authentication, streaming, and analytics:

- **Streaming**: NMS handles RTMP ingestion and converts streams to HLS using FFmpeg, serving them at a designated endpoint.
- **Authentication**: JWT secures access with admin and client roles. Admins approve client accounts, with passwords hashed using bcrypt and stored securely.
- **Analytics**: Stream metrics (e.g., frame rate, bitrate) are logged to Elasticsearch for visualization in Kibana and Grafana.

B. Frontend

The frontend, developed with HTML, CSS, and JavaScript, provides role-specific interfaces:

- **Admin Dashboard**: Displays six video frames, a playback timeline, and a client approval interface.
- **Client Dashboard**: Allows approved clients to monitor streams without administrative access, using flv.js for playback.
- **Authentication Interface**: Handles login and signup with JWT validation.

C. External Integrations

- **Elasticsearch**: Stores stream metrics for real-time analytics.
- **Kibana and Grafana**: Visualize metrics like bitrate and latency.
- **FFmpeg**: Converts RTMP streams to HLS for browser compatibility.

III. Implementation

The system was designed for modularity and scalability using open-source technologies.

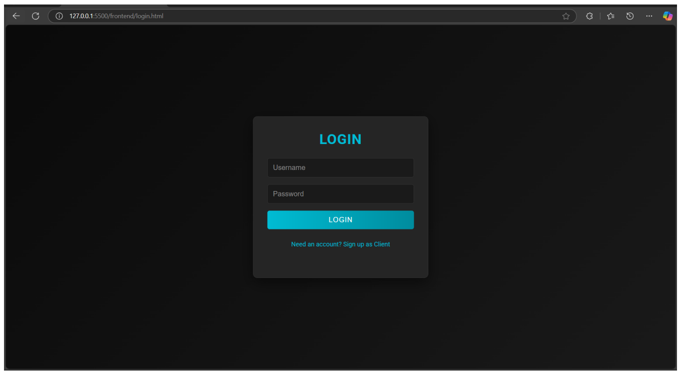

Fig. 1. Login interface for the NOC/CSOC system.

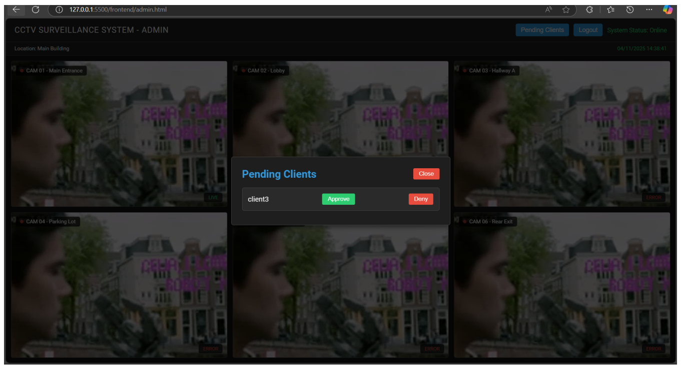

Fig. 2. Admin dashboard showing six video frames and client approval interface.

A. Streaming Pipeline

NMS processes RTMP streams and outputs HLS segments using FFmpeg, stored in a media directory and served via Express. Stream metrics are collected periodically using FFprobe and indexed in Elasticsearch for analysis.

B. Authentication

JWT supports two roles:

- **Admin**: Manages streams and approves clients.
- **Client**: Views streams post-approval, with restricted access.

Client registration requires admin approval, implemented through dedicated API endpoints, ensuring compliance with data privacy standards. An example of the login interface is shown in Fig. 1.

C. Dashboards

Both admin and client dashboards display six frames labeled for locations (e.g., Main Entrance). Currently, a single HLS stream is shown across all frames, with HLS.js and flv.js enabling playback. A fullscreen mode with a timeline supports pausing and seeking within a simulated 10-minute window. The admin dashboard is depicted in Fig. 2.

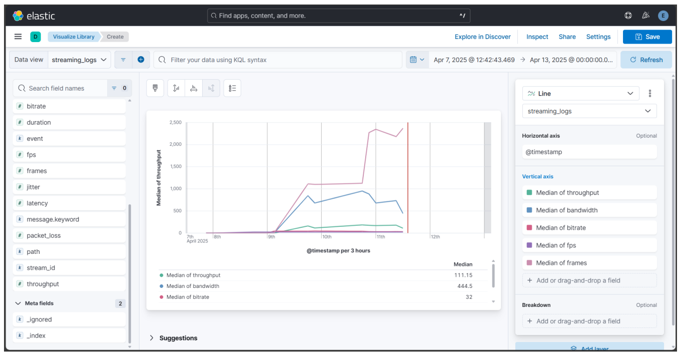

Fig. 3. Kibana dashboard visualizing stream metrics (throughput, bitrate, FPS) over time.

D. Analytics

Stream events and metrics (e.g., frame rate, bitrate) are logged to Elasticsearch and visualized in Kibana and Grafana, providing insights into stream performance. An example visualization is shown in Fig. 3.

IV. Challenges and Attempted Integrations

Several features were explored but not fully integrated:

- **Firebase Authentication**: Introduced latency, prompting a shift to JWT.
- **Multi-Stream Support**: Limited to a single stream due to NMS configuration issues.
- **Threat Detection**: A machine learning model for crowd counting was tested but not deployed due to performance constraints.
- **WebRTC and MediaSoup**: Explored for low-latency streaming but incompatible with the HLS setup.
- **Splunk**: Attempted for advanced analytics but not completed due to setup complexities.
- **Stream Recording**: Hindered by storage limitations.

These efforts highlighted the importance of aligning technologies with system requirements.

V. Evaluation

The system achieves stable HLS streaming, secure authentication, and effective analytics visualization. Key features include:

- **Streaming**: Reliable HLS delivery with low latency (2 seconds).
- **Authentication**: Robust JWT-based access control.
- **Analytics**: Real-time insights via Kibana and Grafana (see Fig. 3).

Limitations include the single-stream display and unintegrated features. Scalability for multiple streams remains untested, suggesting a need for further performance evaluation.

VI. Future Work

Planned enhancements include:

- Enabling unique streams per frame.
- Integrating optimized threat detection models.

- Implementing stream recording with efficient storage.
- Exploring WebRTC for low-latency applications.
- Completing Splunk integration for advanced analytics.

VII. Conclusion

The NOC/CSOC system offers a robust platform for video monitoring in operational centers, leveraging NMS, HLS, and Elasticsearch. Despite limitations like single-stream display, its secure authentication and analytics capabilities meet NOC/CSOC needs. Lessons from integration challenges underscore the importance of technology compatibility. Future improvements aim to enhance scalability and functionality, strengthening its operational value.

References

[1] Node-Media-Server, 2023. [Online]. Available: https://github.com/illuspas/Node-Media-Server

[2] Apple Inc., "HTTP Live Streaming (HLS)," 2023. [Online]. Available: https://developer.apple.com/streaming/

[3] Elastic, "Elasticsearch," 2023. [Online]. Available: https://www.elastic.co/elasticsearch/

[4] D. Bradley, "JSON Web Token (JWT)," IETF RFC 7519, May 2015.

[5] Elastic, "Kibana: Analytics and Visualization," 2025. [Online]. Available: https://www.elastic.co/kibana

[6] Grafana Labs, "Grafana: The Open Observability Platform," 2025. [Online]. Available: https://grafana.com/

[7] FFmpeg Developers, "FFmpeg: A Cross-Platform Solution for Streaming," 2025. [Online]. Available: https://ffmpeg.org/

[8] Bilibili, "flv.js: FLV Playback in HTML5," 2025. [Online]. Available: https://github.com/bilibili/flv.js

[9] Node.js Foundation, "Node.js," 2025. [Online]. Available: https://nodejs.org/

[10] Express.js, "Express: Web Framework for Node.js," 2025. [Online]. Available: https://expressjs.com/