LOGSENTINEL

Intrusion Detection & Log Analysis Tool

Project Report

1. Introduction

Every server on the internet keeps log files that record every request, login attempt, and blocked connection. When an attacker tries to break into a system, they leave traces in these logs. The challenge is that log files can contain millions of lines — impossible to read manually.

LogSentinel is a browser-based tool that solves this problem. A user drops in any log file and immediately gets a full security analysis — no installation, no server, and no data sent anywhere. It supports 7 log formats, detects 26 threat patterns across 6 attack categories, and presents results as charts and a filterable alert table with export options.

2. Abstract

LogSentinel is built entirely with HTML, CSS, and JavaScript using the ES module system. The parser automatically detects the log format and converts each line into a structured object. The detection engine then runs 26 threat detection algorithms covering: Brute Force, Reconnaissance, Denial of Service, Exploitation, Privilege Escalation, and Blacklisted IP Detection. Results are shown in 5 interactive charts and a color-coded alert table. Alerts can be exported as JSON, CSV, or a standalone HTML report.

3. Tools Used

- HTML5 — Page structure. index.html contains markup only, no embedded scripts or styles.
- CSS3 — All styling in styles.css using CSS variables, flexbox, grid, and animations.
- JavaScript (ES Modules) — Logic split into 6 files: state.js, parser.js, detector.js, charts.js, ui.js, main.js.
- Chart.js 4.4.1 — Five charts (bar, line, doughnut). Loaded from Cloudflare CDN.
- VS Code + Live Server, Python HTTP server, Browser DevTools — for development and testing.

4. Steps Involved in Building This Tool

Step 1 — Architecture Planning: Mapped the data flow: drop file → parse → detect → visualize. Each module was assigned one job so changes in one area do not break others.

Step 2 — Log Parser (parser.js): Reads each line using regular expressions and auto-detects the format. Supports Apache CLF, Apache Combined, Apache Error Log, SSH Auth Log, System/Syslog, Firewall (UFW/iptables), and Application Logs. Extracts IP, time, method, URL, status code, user-agent, and more.

Step 3 — Threat Detector (detector.js): 26 detection functions across 6 categories. Category 1 (Brute Force): SSH/web login abuse, root attacks, credential stuffing, odd-hour logins. Category 2 (Recon): 404 floods, admin probing, port scans. Category 3 (DoS): HTTP floods, firewall floods, error storms. Category 4 (Exploitation): SQLi, XSS, directory traversal, command injection, malicious user-agents like sqlmap and nikto. Category 5 (Privilege Escalation): sudo failures, risky sudo commands, new users, password changes. Category 6 (Blacklist): 32 known-bad IPs cross-referenced across all log sources.

Step 4 — Charts (charts.js): Built 5 Chart.js visualizations: Top IPs bar chart (red for alerted IPs), Activity Timeline (5-min bins with threat overlay), HTTP Status distribution, Alert Categories doughnut, and Alert Severity doughnut.

Step 5 — Interface (ui.js): Drag-and-drop upload, animated progress bar, stat cards, dual filter bars (severity + category), live search, raw log viewer with color-coded lines, and three export formats.

Step 6 — Testing: Tested against real Apache access logs, SSH auth.log samples, OpenStack Nova logs, and manually crafted attack lines to verify each detector.

---

5. Conclusion

LogSentinel is a complete, working intrusion detection tool that runs entirely in a web browser with no backend required. It parses 7 log formats, detects 26 threat patterns across 6 attack categories, and presents results through 5 charts and a filterable table — all offline and exportable.

The code is organized into 6 modules with clear responsibilities, making it easy to extend. Adding a new log format requires only changes to parser.js; adding a new detector requires one function in detector.js.

This project demonstrates practical skills in web development, security analysis, regex pattern matching, time-window-based detection, and data visualization.

--- End of Report ---