CyberSec Monitor 360

Real-time SOC Dashboard (Django + DRF + Channels + SQLite)

1. Problem & Goal

Security teams need near real-time visibility into alerts and incident response workflow. This project demonstrates a SOC monitoring prototype with live data.

2. Key Features (Demo)

• Live dashboard: severity % today, last scan, event timeline hover, threat map, incident types. • Alerts & Cases pages with filters. • Case details: tasks (1–5), verdict (TP/FP/Duplicate/Other), dispatch (Telegram/SIEM), playbooks wiki. • Rules with custom query templates and SLA guidance.

3. Architecture

Frontend	Django Templates + Bootstrap + Chart.js + WebSocket client
Backend	Django + DRF (REST) + Channels (WebSocket)
DB	SQLite (Alerts, Cases, Tasks, Rules, Clients, Employees, Dispatch)
Data	Async log generator (asyncio + aiohttp) posts events to /api/ingest/

4. Five Incident Scenarios

Brute Force, SQL Injection, XSS, Path Traversal, Suspicious Windows Service install. Each scenario auto-creates a case with tasks and links to a playbook.

5. SLA Reaction Time (Simplified)

Critical: 15–60 min • High: 1–4 h • Medium: 4–8 h • Low: 24–72 h

6. Syllabus Coverage

Topic	Where used
Python basics / control	Project structure, services, filtering logic
Data types / collections	Enums, JSON fields, aggregations by severity/type/hour
Async	asyncio generator + WebSocket push notifications
Modules & files	apps (core/soc), notebook, JSON ingestion
OOP	Django models + service layer
SQLite	db.sqlite3
Web/REST + Django	DRF endpoints + templates
Data science	Jupyter notebook: notebooks/analytics_today.ipynb

7. How to Run (Windows)

1) pip install -r requirements.txt 2) python manage.py migrate 3) python manage.py seed_demo_data 4) daphne -p 8000 config.asgi:application 5) python manage.py run_log_generator --count 75

8. GitHub

Upload the repository and include the link in the report. (The submission ZIP contains all sources + PDFs.)