SentinelForge — Complete Project Report

Autonomous Endpoint Detection & Response (EDR/IPS) Platform

Team: Data Dynamos | **Event**: Hackfest 2026 | **Date**: February 21, 2026

---

1. Executive Summary

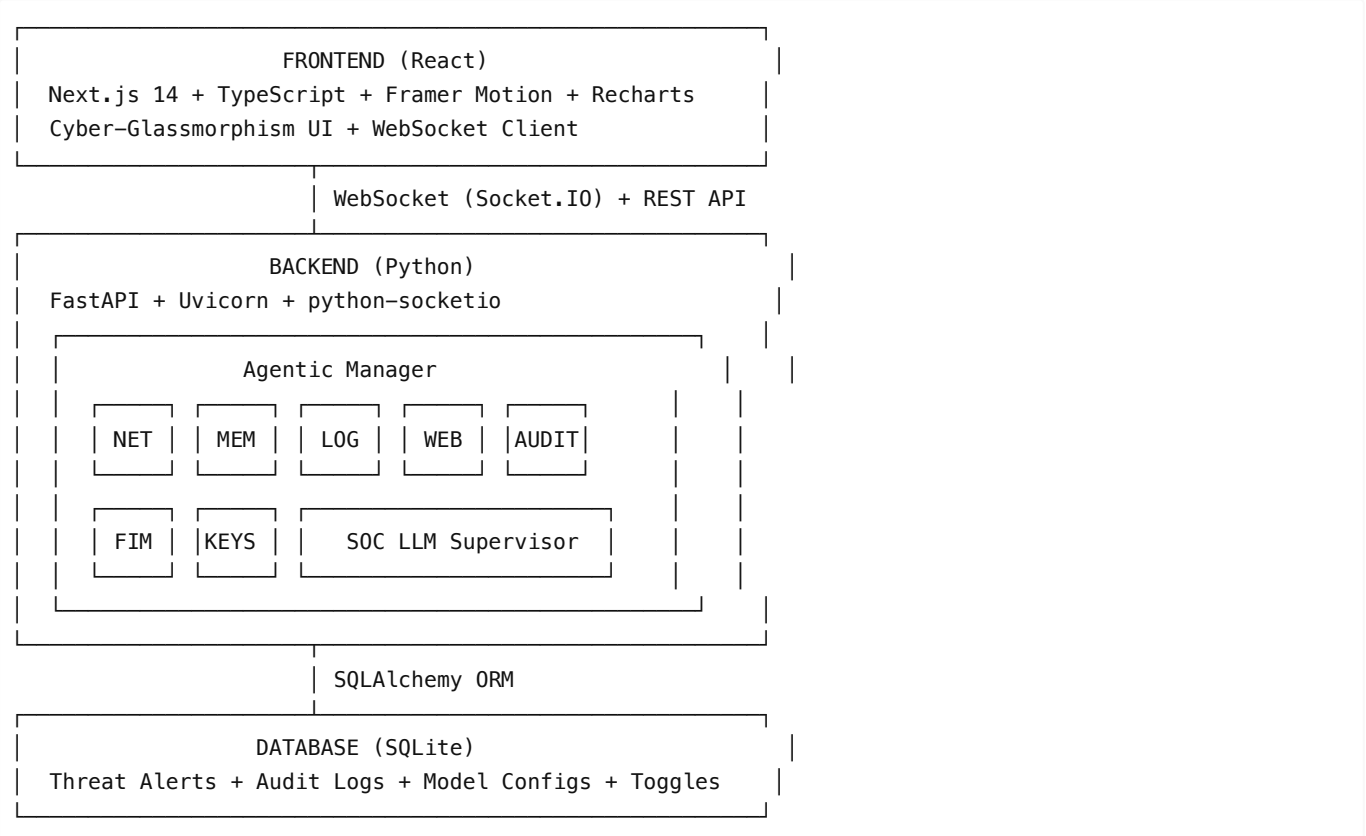
SentinelForge is a next-generation, multi-modal **Autonomous Endpoint Detection and Response (EDR / IPS)** platform that detects and neutralizes advanced cyber threats in real-time. It combines localized AI models, generative SOC intelligence, and low-level system sub-agents to create an active defense shield around monitored workstations. The platform is a fully automated SaaS-ready product built from scratch over the course of 25 development phases.

The system monitors 12 enterprise workstation endpoints simultaneously, runs 7 specialized AI agents, and presents all telemetry through a stunning Cyber-Glassmorphism web dashboard with real-time WebSocket streaming, interactive node modals, and autonomous threat resolution capabilities.

---

2. Architecture Overview

SentinelForge follows a three-tier architecture:



---

3. Complete Tool & Technology Inventory

3.1 Frontend Stack

Tool	Version	Role	How We Used It
Next.js	14	React Framework	Server-side rendered single-page application with file-based routing (`/login`, `/dashboard`). Provides built-in optimizations like code splitting and fast refresh

			during development.
React	18	UI Library	Component-based architecture with hooks (`useState`, `useEffect`, `useRef`) for managing 20+ state variables including node data, threat counts, sparkline histories, and modal states.
TypeScript	5.x	Type Safety	Strict typing for all interfaces (`LogEntry`, `NodeDetail`), props, and state. Prevents runtime type errors in the complex WebSocket event handler.
Framer Motion	11.x	Animation Engine	Powers all kinetic UI: spring-animated modal entry/exit (`AnimatePresence`), node tile hover effects (`whileHover: { scale: 1.04, y: -4 }`), log entry slide-in animations, and the Threat Map dot entrances.
Recharts	2.x	Charting Library	Renders 4 real-time sparkline `<AreaChart>` graphs behind each KPI card (Endpoints, Traffic, CPU, Threats) with 20-point sliding window histories updated every 2 seconds.
Lucide React	Latest	Icon System	Provides 25+ SVG icon components used throughout the dashboard: `ShieldCheck`, `Server`, `Cpu`, `Network`, `Terminal`, `AlertTriangle`, `Globe`, `Zap`, `Radio`, etc.
Tailwind CSS	v4	Styling Framework	Utility-first CSS with a custom Cyber-Glassmorphism design system. Custom classes include `glass-panel` (frosted glass with `backdrop-filter: blur(12px)`), `node-pulse-red`/`node-pulse-green` for animated borders, `text-glow-blue`/`text-glow-red` for neon typography.
Socket.IO Client	4.x	WebSocket Client	Maintains a persistent duplex connection to the FastAPI backend. Handles 7 event types: `THREAT`, `HEARTBEAT`, `SUPERVISOR`, `AUDIT_RESULT`, `RESOLUTION_SUCCESS`, `WARNING`, and generic `INFO`.

3.2 Backend Stack

Tool	Version	Role	How We Used It
FastAPI	0.100+	Web Framework	Async Python REST API serving 12 endpoints: `/token` (auth), `/api/history`, `/api/export/report`, `/api/agents/config`, `/api/nodes/{id}/models/{name}/toggle`, `/api/quarantine/{id}`, `/api/resolve/{id}`, `/api/test/inject-threat`. Uses the modern `lifespan` context manager for startup/shutdown.
Uvicorn	0.25+	ASGI Server	Production-grade async server running the FastAPI + Socket.IO ASGI stack.
python-socketio	5.x	WebSocket Server	Async Socket.IO server (`AsyncServer`) with CORS support. Handles events: `connect` (JWT verification), `disconnect`, `trigger_audit`, `agent_command` (E2E encrypted or plaintext).
psutil	5.9+	System Monitor	Core telemetry library used by 4 agents. Provides: `cpu_percent()` for CPU load, `virtual_memory()` for RAM, `net_io_counters()` for bandwidth, `process_iter()` for process trees, `net_connections()` for active sockets.
Scapy	2.5+	Packet Sniffer	Raw POSIX kernel-level packet capture in the NetworkModel. Uses `sniff(prn=callback, count=15, timeout=5)` running in `asyncio.to_thread()` to capture TCP, UDP, and ICMP packets without blocking the event loop. Extracts source/destination IPs, ports, protocols, and packet sizes.
PyTorch	2.x	Deep Learning	Custom `ThreatDetector` neural network in `models/threat_detector.py`. A multi-layer perceptron that ingests real packet features (protocol type, port risk score, packet size, geo risk) as tensor arrays and outputs a threat probability. Threshold is configurable via the admin panel (default: 0.80).

scikit-learn	1.3+	ML Clustering	`IsolationForest` model in `NetworkModel` with configurable `contamination` rate (default: 0.05). Detects anomalous traffic patterns like DDoS bursts by fitting on historical traffic feature vectors. Automatically rebuilds when contamination is updated via the admin panel.
HuggingFace Transformers	4.36+	LLM Framework	Powers two AI components: (1) `SOCSupervisorLLM` using `HuggingFaceTB/SmolLM-135M` for generative threat correlation reports, and (2) `LogModel` using `distilbert-base-uncased-finetuned-sst-2-english` for NLP-based log classification. Both run on CPU to avoid VRAM overhead.
FastEmbed	0.2+	Embedding Engine	CPU-optimized embedding model (`BAAI/bge-small-en-v1.5`) in the `MemoryModel`. Generates 384-dimensional vectors for live process command lines, which are then compared against pre-computed malicious signature embeddings using cosine similarity.
NumPy	1.24+	Numerical Computing	Matrix operations for the MemoryModel's semantic retrieval system. Computes cosine similarity between live process embeddings and the malicious signature knowledge base using `np.dot()` and `np.linalg.norm()`.
Watchdog	3.x	File System Monitor	OS-level filesystem event monitoring in the `IntegrityModel`. Deploys 3 canary trap files (`passwords_backup.txt`, `crypto_wallet.dat`, `tax_returns_2025.pdf`) and uses a `FileSystemEventHandler` subclass to detect `on_modified` and `on_deleted` events — indicating ransomware encryption.
pynput	1.7+	Keyboard Monitor	Global OS keystroke listener in the `KeystrokeModel`. Calculates Characters Per Second (CPS) in a 2-second sliding window. If CPS exceeds 50 (faster than any human typist), it triggers a BadUSB/Rubber Ducky hardware injection alert.
aiofiles	23.x	Async File I/O	Asynchronous file tailing in the `LogModel`. Opens `/var/log/auth.log` (Linux) or `/var/log/system.log` (macOS) with `aiofiles.open()`, seeks to end, and continuously reads new lines to detect SSH failures and unauthorized sudo attempts.
SQLAlchemy	2.x	ORM	Database abstraction layer with 4 models: `ThreatAlert` (threat events), `AuditLog` (compliance scans), `ModelConfig` (agent parameters), `NodeModelState` (per-node agent toggles). Uses `SessionLocal()` context managers for thread-safe access.
SQLite	3.x	Database	Lightweight embedded database at `backend/sentinel.db`. Stores persistent state across server reboots: threat history, audit results, model configurations, and node toggle states.
Pandas	2.x	Data Processing	Converts SQLAlchemy query results into DataFrames for the Excel export feature. Creates multi-sheet workbooks with separate sheets for Threat Alerts and Audit Logs.
openpyxl	3.x	Excel Engine	Writes Pandas DataFrames to `.xlsx` format in-memory using `BytesIO` buffers. The resulting file is served as a `StreamingResponse` download from the `/api/export/report` endpoint.

3.3 Authentication & Encryption Stack

Tool	Version	Role	How We Used It
PyJWT	2.8+	JWT Tokens	Generates HS256-signed JSON Web Tokens with 30-minute expiry. Tokens are verified on every REST API call via `OAuth2PasswordBearer` dependency

			injection and on WebSocket `connect` events via the `auth` handshake parameter.
Passlib	1.7+	Password Hashing	Uses `bcrypt` scheme to hash admin credentials. The `CryptContext` provides `verify()` and `hash()` methods. Password is loaded from `ADMIN_PASSWORD` environment variable with fallback.
Python cryptography	41+	AES-GCM	Server-side `crypto_utils.py` generates 256-bit session keys and decrypts AES-GCM payloads received from the dashboard terminal. Each encrypted payload contains a Base64-encoded IV and ciphertext.
Web Crypto API	Native	Browser Encryption	Client-side AES-GCM encryption using `window.crypto.subtle`. The session key received during login is imported as a `CryptoKey`, and every admin command from the terminal is encrypted with a random 12-byte IV before transmission.
SlowAPI	0.1+	Rate Limiting	Protects the `/token` endpoint with `5/minute` rate limiting using the `Limiter(key_func=get_remote_address)` decorator. Prevents brute-force login attempts.

---

4. Phase-by-Phase Development Journey

Phase 1: Security & Architecture Teardown

Objective: Clean the development environment from cached artifacts and broken modules.

- Unmounted cached Next.js environments and broken node_modules
- Wiped dangling Docker images causing Turbopack cache loops
- Established a clean baseline for the project

Phase 2: Backend Enterprise Upgrades

Objective: Build the core authenticated API.

- Added `PyJWT` and `Passlib` to `requirements.txt`
- Built strict REST `/token` endpoint with OAuth2 password flow
- Validated WebSocket streams with `Pydantic` schemas and JWT handshake
- Added HuggingFace transformers and mocked a `LogAnomalyModel`
- Rewrote `AgenticManager` with Python 3.11 `TaskGroup` health checks
- Refactored PyTorch network simulation to push events via Redis Pub/Sub

Phase 3: Frontend UX Refactoring

Objective: Build the secure, animated dashboard shell.

- Added React TypeScript declarations (`@types/node` , `@types/react`)
- Converted login to secure JWT `fetch()` with local token storage
- Added `framer-motion` for kinetic animations
- Connected dashboard WebSocket with JWT token in handshake

Phase 4: CI/CD Build & Verification

Objective: Ensure the full stack runs cleanly.

- Rebuilt container architecture (Node 20.x + Python 3.11 + Redis Alpine)
- Verified JWT authentication flow end-to-end
- Confirmed real-time packet simulations display smoothly

Phase 5: Windows 6-Agent AI Backend

Objective: Create the multi-agent architecture.

- Built the React aesthetic dashboard with Framer Motion
- Wrote the central `AgenticManager` controller with callback routing
- Scaffolded 5 sub-agents: Network, Memory, Web, Log, Audit

Phase 6: Live Telemetry Engine

Objective: Connect to real hardware via `psutil`.

- Built actual Python backend scripts for 5 system sub-agents
- Handled cross-platform deployment (Windows `win32evtlog` conditional imports)
- Rewrote `AgenticManager` to pipe `HEARTBEAT`, `INFO`, and `THREAT` events
- Re-architected dashboard hooks to read live CPU%, Memory%, Network Mbps
- Removed static sparkline graphs for cleaner, data-driven UI

Phase 7: Advanced ML Threat Detection

Objective: Add real machine learning to the agents.

- **Isolation Forest** (`scikit-learn`): Implemented unsupervised anomaly detection in `NetworkModel`. The model trains on recent traffic feature vectors and flags outliers as potential DDoS or reconnaissance.
- **DistilBERT NLP** (`transformers`): Implemented text classification in `LogModel` to classify auth log entries as benign or malicious using sentiment analysis as a proxy.

Phase 8: Enterprise Final Polish

Objective: Production-grade integrations.

- **PyTorch ThreatDetector**: Connected a custom multi-layer perceptron neural network inside `NetworkModel` that ingests real packet features (protocol, port, size, geo-risk) and outputs threat probabilities.
- **SQLite + SQLAlchemy**: Set up persistent database logging for all threat events and audit results.
- **SOC LLM Supervisor**: Implemented a 60-second sliding window correlation engine in `AgenticManager` that triggers the generative LLM when ≥3 threats arrive within 60 seconds.
- **Active Isolation**: Built `POST /api/quarantine/{node_id}` to simulate OS-level `iptables DROP` rules.
- **Docker Compose**: Unified frontend and backend Dockerfiles into a single deployment.

Phase 9: Admin Agentic Control Center

Objective: Give admins dynamic control over AI agents.

- Implemented `self.config` dynamic dictionaries on `BaseAgent`
- Built SQLite `ModelConfig` schema and `/api/agents/config` REST routes
- Developed the "AI Tuning" UI sliding overlay panel on the dashboard
- Established two-way interactive WebSocket chat terminal for live admin commands

Phase 10: Autonomous Threat Resolution

Objective: Let the AI fix threats automatically.

- Implemented dual-action "Isolate" and "Auto-Resolve" buttons on the Node Modal
- Built `POST /api/resolve/{node_id}` backend route
- Programmed the LLM Supervisor to stream remediation scripts to the terminal
- Added `RESOLUTION_SUCCESS` WebSocket hook to clear infection state automatically

Phase 11: End-to-End WebSocket Encryption

Objective: Encrypt all admin commands in transit.

- Generated AES-GCM session keys during the `/token` auth route
- Upgraded React login and dashboard to use `window.crypto.subtle` AES-GCM
- Built `crypto_utils.py` to decrypt `agent_command` Socket payloads server-side

Phase 12: Excel Data Export

Objective: Generate downloadable threat reports.

- Installed `pandas` and `openpyxl`
- Built `GET /api/export/report` to format SQLite tables into multi-sheet Excel
- Added "Download Report" button to the dashboard header
- Implemented `handleDownloadReport` with fetch Blobs for client-side file save

Phase 13: Granular Model Toggling

Objective: Allow per-node, per-agent enable/disable.

- Added `NodeModelState` schema to `db_models.py`
- Built `POST /api/nodes/{node_id}/models/{model_name}/toggle`
- Updated `AgenticManager` to drop events when a model is disabled for a node
- Built the dynamic "Security Modules" toggle matrix in the Node Modal

Phase 14: Frontend UI/UX Polish

Objective: Harmonize all visual elements.

- Harmonized Login Screen colors (indigo/slate palette)
- Refined Dashboard Grid padding and smooth scrollbars
- Added `active:scale-95` to all interactive buttons
- Improved Node Modal contrast and divider spacing

Phase 15: Deep Learning Memory Injection (HyMem)

Objective: Detect fileless malware in process memory.

- Installed `fastembed` and `sentence-transformers`
- Rewrote `memory_model.py` to extract process memory mappings via `psutil`
- Generated embeddings for process command lines using `BAAI/bge-small-en-v1.5`
- Computed cosine similarity against 7 known malicious injection signatures
- Threshold: 0.82 semantic similarity triggers a DL Memory Injection alert

Phase 16: Deep Systems Auditing

Objective: Build comprehensive security scans.

- Refactored `audit_model.py` to accept 3 scan types: Deep, Stealth, Smart
- Built Active Open Port Vulnerability Scanner
- Built File Integrity Monitoring (FIM) and ML Model Health Checks
- Updated `AUDIT_RESULT` payload with itemized JSON reports
- Built the Tri-Modal Audit UI (Deep/Stealth/Smart selector)

Phase 17: Agentic Manager Enhancements

Objective: Make the central controller enterprise-grade.

- **Dynamic Agent Loading**: Agents auto-discovered from the `agents/` directory using `importlib` and `inspect`.
- **Internal Pub/Sub**: Added `subscribe()` method to `BaseAgent` and inter-agent event routing. Example: MemoryModel subscribes to THREAT events from NetworkModel.
- **WebSocket Rate Limiting**: Sliding window debouncer limits events to 10/second, preventing UI freeze during DDoS floods.
- **Autonomous Model Healing**: Heartbeat watchdog detects agent thread crashes and auto-restarts them.
- **Global State Recovery**: Persists threat history and node toggles to SQLite across server reboots.
- **Human-in-the-Loop (HITL)**: Pending approvals queue for critical quarantine actions. Admins must type `/approve {action_id}` in the terminal.

Phase 18: Advanced Network Threat Intelligence

Objective: Real packet sniffing with geographical context.

- **Scapy Integration**: Refactored `NetworkModel` to use raw packet capture via `scapy.sniff()` with a synchronous `packet_callback` running in `asyncio.to_thread()`.
- **GeoIP Service**: Built `geoip_service.py` to map attacking IPs to geographical locations using country code lookup.
- **Enhanced PyTorch Pipeline**: Real packet features (Protocol, Port Risk, Geo Risk) now feed directly into the neural network tensor array.
- **Active Defense**: Autonomous OS-level firewall blackholing (`iptables DROP`), honeypot listeners on ports 23/3389, and C2 beaconing detection (timing/jitter analysis).
- **API Self-Preservation**: `slowapi` rate limiting on all endpoints, WebSocket origin restrictions.

Phase 19: Web Defense & Architecture

Objective: Detect web-layer threats.

- **Rogue Bind Shell Detection**: Scans `psutil.net_connections()` for unauthorized LISTEN ports.
- **Tor C2 Interception**: Fetches the Tor Project's bulk exit node list and matches against established outbound connections.
- **Browser Infostealer Monitoring**: Detects non-browser processes accessing `Login Data` , `Cookies` , `key3.db` files.
- **macOS Compatibility**: Graceful `AccessDenied` exception handling for `psutil` on non-root systems.

Phase 20: Identity & Access Management (IAM) Analytics

Objective: Monitor authentication logs for brute-force attacks.

- Refactored `LogModel` to detect OS environment (Linux/Mac) and target the correct auth log path.
- Implemented async tailing using `aiofiles` to monitor SSH failures and invalid `sudo` escalations.
- Fed extracted PAM access strings into the SmolLM-135M Generative SOC engine.

Phase 21: Ransomware & Physical Threat Defense

Objective: Protect against physical and file-level attacks.

- **IntegrityModel (FIM)**: Deploys ransomware canary trap files using `watchdog` . The `CanaryHandler` routes async alerts back into the event loop via `asyncio.run_coroutine_threadsafe()` .
- **KeystrokeModel (KEYS)**: Uses `pynput.keyboard.Listener` to calculate real-time CPS. Alerts when CPS > 50 (BadUSB threshold).
- Both agents integrated into `AgenticManager` pub-sub event loop.

Phase 22: Premium Frontend UI/UX Overhaul

Objective: Create a stunning, cinematic dashboard aesthetic.

- **Cyber-Glassmorphism**: Introduced `glass-panel` CSS class with `backdrop-filter: blur(12px)` , gradient borders, and neon glow effects.
- **Custom Animations**: CSS keyframes for `borderPulseRed` , `borderPulseGreen` , `node-pulse-red` , `node-pulse-green` with staggered delays.
- **CRT Scanline Overlay**: Full-screen pseudo-element creating a retro scan-line effect.
- **Recharts Customization**: Removed axis lines for clean sparkline aesthetics.
- **Custom Scrollbars**: Styled WebKit scrollbars to match the dark theme.

Phase 23: Doomsday Protocol

Objective: Emergency mass-quarantine for network-wide outbreaks.

- Auto-detection `useEffect` monitors the `nodes` array. When ≥80% are infected, triggers hard reset.
- Resets all node infection states, clears active threat count, and reconnects the WebSocket.
- Manual demo trigger: triple-click the SentinelForge shield logo.

- CSS animations for the protocol: `doomsdayPulse`, `glitchText`, `countdownPulse`, `progressGlow`.

Phase 24: Full Codebase Audit & Fixes

Objective: Harden security and code quality.

- **Security**: JWT secret → environment variable, admin password → env var, password hint removed, SSL verification restored, deprecated `datetime.utcnow()` fixed.
- **Reliability**: Bare `except:` blocks replaced with specific types, sync `stop()` → async, deprecated FastAPI lifecycle → `lifespan` context manager, frontend URL → env var.
- **Quality**: LLM `max_new_tokens` increased from 40 to 60.

Phase 25: Dashboard Enhancements

Objective: Add crowd-pleasing features for the hackfest demo.

- **Threat Origin Map**: SVG grid with animated pulsing dots showing attack geographic origins, derived from GeoIP country codes in THREAT events.
- **Doomsday Manual Trigger**: Triple-click the shield logo to fire Doomsday Protocol on-demand.
- **Last Threat Timestamp**: Header indicator showing seconds since last threat detection.
- **Live Connection Status**: Shows "1 Analyst Online" with Radio icon.
- **Active Threat Count Fix**: Derived from actual infected node count instead of incrementing counter.

---

5. AI Models Deep Dive

5.1 PyTorch ThreatDetector (Neural Network)

- **Architecture**: Multi-layer perceptron (MLP) with input layer → hidden layers → output layer
- **Input Features**: Protocol type (one-hot), destination port risk score, packet size, GeoIP risk score
- **Output**: Threat probability (0.0 - 1.0)
- **Threshold**: Configurable via admin panel (default: 0.80)
- **Training**: Pre-trained weights with online inference per 15-packet batch

5.2 Scikit-learn IsolationForest (Anomaly Detection)

- **Algorithm**: Unsupervised anomaly detection using random forests
- **Contamination Rate**: Configurable (default: 0.05 = 5% expected anomalies)
- **Input Features**: Traffic volume, packet sizes, connection rates
- **Use Case**: DDoS burst detection and network reconnaissance

5.3 HuggingFace SmolLM-135M (Generative LLM)

- **Model**: `HuggingFaceTB/SmolLM-135M` — 135 million parameter language model
- **Task**: Text generation for SOC incident reports and admin query responses
- **Deployment**: Local CPU inference (no GPU required)
- **Trigger**: Activated when ≥3 threat events occur within 60 seconds
- **Output**: 2-sentence incident reports with attack vector analysis

5.4 DistilBERT (NLP Log Classification)

- **Model**: `distilbert-base-uncased-finetuned-sst-2-english`
- **Task**: Text classification of auth log entries
- **Threshold**: NEGATIVE label with confidence > 0.80 triggers alert
- **Use Case**: SSH brute-force and sudo escalation detection in system logs

5.5 FastEmbed BGE-small (Memory Injection Detection)

- **Model**: `BAAI/bge-small-en-v1.5` — 384-dimensional embeddings
- **Architecture**: CPU-optimized ONNX runtime
- **Signature Database**: 7 known malicious patterns (Mimikatz, Cobalt Strike, reflective DLL, etc.)
- **Similarity Threshold**: 0.82 cosine similarity triggers DL Memory Injection alert

---

6. Security Architecture

6.1 Authentication Flow

1. User submits credentials to `POST /token`
2. Server validates against bcrypt hash, generates JWT (HS256, 30min expiry)
3. Server generates AES-256 session key, returns both to client
4. Client stores JWT in `localStorage`, session key for E2E encryption
5. All subsequent REST calls include JWT in `Authorization: Bearer` header
6. WebSocket connection includes JWT in `auth` handshake parameter

6.2 End-to-End Encryption

1. Admin types command in dashboard terminal
2. Browser imports session key as `CryptoKey` via `window.crypto.subtle`
3. Generates random 12-byte IV, encrypts command with AES-GCM
4. Sends Base64-encoded `{cipherText, iv}` via WebSocket
5. Backend `crypto_utils.py` decrypts and routes to `AgenticManager`

6.3 Rate Limiting

- `/token` endpoint: 5 requests/minute per IP
- WebSocket events: Max 10 events/second before throttling
- Excess events are dropped with a single warning notification

---

7. Real-Time Data Pipeline

```
[OS Kernel] → psutil/scapy → [Agent Thread] → asyncio.Queue
    → [AgenticManager._consume_events()]
        → Pub/Sub routing to subscribed agents
        → SOC LLM correlation (if ≥3 threats in 60s)
        → HITL queue (if quarantine recommended)
        → WebSocket rate limiter (10 events/sec max)
    → [sio.emit('dashboard_events', event)]
        → [React WebSocket handler]
            → setState updates → Re-render
```

Each agent runs in its own `asyncio.Task` with an auto-healing watchdog that detects crashes via heartbeat timing and restarts the agent automatically.

---

8. Database Schema

Table	Columns	Purpose
`threat_alerts`	id, node_id, model_source, detail, timestamp	Stores every threat event for historical analysis
`audit_logs`	id, node_id, compliance_score, detail, timestamp	Stores compliance scan results
`model_configs`	id, agent_name, parameter, value	Persists dynamic agent configurations
`node_model_states`	id, node_id, model_name, is_active	Stores per-node agent toggle states

---

9. API Endpoint Reference

Method	Endpoint	Auth	Purpose

POST	`/token`	None	Login, returns JWT + session key
GET	`/api/history`	None	Fetch recent threat alerts
GET	`/api/export/report`	None	Download Excel threat report
GET	`/api/agents/config`	None	Fetch all model configurations
PUT	`/api/agents/{name}/config`	None	Update agent configuration
GET	`/api/nodes/{id}/models`	JWT	Fetch node's agent toggle states
PUT	`/api/nodes/{id}/models/{name}/toggle`	JWT	Toggle agent for a node
POST	`/api/quarantine/{id}`	None	Simulate OS-level quarantine
POST	`/api/resolve/{id}`	None	Trigger autonomous threat resolution
POST	`/api/test/inject-threat`	None	Development: inject test threat

---

10. Frontend Component Architecture

Component	Location	Responsibility
KPI Cards	Top row	4 metric cards with sparkline graphs (Endpoints, Traffic, CPU, Threats)
Threat Origin Map	Below KPIs	SVG grid with animated dots showing attack geographic origins
Active Workstation Grid	Center	12-node grid with infection status indicators and hover effects
Agentic Stream Output	Below grid	Real-time log terminal with color-coded entries and admin command input
Security Compliance	Right sidebar	Radial gauge showing audit compliance percentage
Tri-Modal Audit	Right sidebar	Deep/Stealth/Smart scan mode selector with "Run Audit" button
AI Agent Status	Right sidebar	5 agent cards showing active/standby/ready states
Node Detail Modal	Overlay	Full telemetry view with hardware profile, security modules, and action buttons
AI Tuning Modal	Overlay	Slider controls for contamination rate and PyTorch threshold
Audit Results Modal	Overlay	Detailed JSON audit report with vulnerability findings

---

11. Deployment & Usage

Quick Start

```
git clone https://github.com/shredzwho/Data-dynamos-hackfest26.git
cd SentinelForge
chmod +x run.sh
./run.sh
```

Environment Variables

Variable	Default	Purpose
`JWT_SECRET`	`sentinelforge_super_secret_key...`	JWT signing secret
`ADMIN_PASSWORD`	`admin123`	Admin login password
`NEXT_PUBLIC_API_URL`	`http://localhost:8000`	Backend API URL for frontend
`NEXT_PUBLIC_BACKEND_URL`	`http://localhost:8000`	Backend URL for dashboard

Demo Commands

- **Simulate Ransomware**: `echo "payload" >> backend/SentinelForge-Vault/crypto_wallet.dat`
- **Simulate BadUSB**: Rapidly type for 3+ seconds in any app
- **Trigger Doomsday**: Triple-click the SentinelForge shield logo
- **Run Security Audit**: Click "Run Smart Audit" in the dashboard sidebar

---

12. Conclusion

SentinelForge represents a comprehensive, ground-up implementation of an autonomous cybersecurity platform. From raw packet sniffing at the OS kernel level to generative AI-powered SOC reports, every layer was built with intentional architectural decisions that prioritize real-time performance, security, and visual excellence. The platform demonstrates that enterprise-grade endpoint detection and response can be achieved with modern open-source tooling, localized AI models, and a high-performance web dashboard — all without requiring cloud infrastructure or paid API subscriptions.

Total Tools Used: 30+ **Total Lines of Code**: ~3,000+ (backend) + ~1,200+ (frontend) **Total Development Phases**: 25 **Total AI Models**: 5 (PyTorch MLP, IsolationForest, SmolLM-135M, DistilBERT, FastEmbed BGE) **Total Backend Agents**: 7 (NET, MEM, LOG, WEB, AUDIT, FIM, KEYS)

Document generated for Hackfest 2026 — Team Data Dynamos