Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Cortexia: Code-Driven Autonomous Threat Correlation and AI-Assisted Remediation

Aparna Chandran
Dept. of Computer Science & Engineering
Rajadhani Institute of Engineering and Technology
Trivandrum, India

Ajmal S Ahammed
Dept. of Computer Science & Engineering
Rajadhani Institute of Engineering and Technology
Trivandrum, India

Abhinav P
Dept. of Computer Science & Engineering
Rajadhani Institute of Engineering and Technology
Trivandrum, India

Anjali Suresh
Dept. of Computer Science & Engineering
Rajadhani Institute of Engineering and Technology
Trivandrum, India

Adithyan U
Dept. of Computer Science & Engineering
Rajadhani Institute of Engineering and Technology
Trivandrum, India

***Abstract* - Modern cybersecurity operations often suffer from isolated tooling and alert fatigue. In this paper, we present Cortexia, an autonomous, integrated cybersecurity platform implemented natively via a Python asynchronous backend (FastAPI) and a React TypeScript frontend. Bypassing disjointed theoretical models, the platform actively utilizes an `apscheduler` routine to execute cyclic vulnerability evaluations (Trivy, OSV, Semgrep) and multi-layered malware scans (ClamAV, YARA, VirusTotal). We detail the ThreatSentinel correlation engine, which programmatically fuses scan outcomes—such as YARA behavioral offsets and Trivy CVEs—into a unified SQLite persistent memory database. Furthermore, the architecture features PatchMaster, a remediation module algorithmically mapping identified exposures to exact package-management terminal commands. Finally, the platform embeds a "Blue Team Copilot" via an AI Router that retrieves live JSON scan metrics and injects them as prompt context into a local Ollama Large Language Model limit, bridging the gap between raw analytical data and human- readable incident response.**

Keywords - Cybersecurity, Threat Correlation, YARA, VirusTotal, FastAPI, Local LLM, Automated Remediation, SQLite.

I. INTRODUCTION

The transition to continuous deployment environments has expanded the attack surface, creating scenarios where traditional, isolated security scanners generate overwhelming alert noise. Modern Security Operations Centers (SOCs) require systems that do not merely generate logs, but autonomously correlate them into stateful incidents.Cortexia is developed to directly address these architectural shortcomings. Rather than functioning as a passive log aggregator, the system's backend (`auto_scanner.py`) orchestrates a deterministic execution cycle via `BackgroundScheduler`. It staggers four prima ry threaded pipelines: a Vulnerability Scan every 2 hours, a Malware Scan every 2 hours and 15 minutes, a Telemetry snapshot hourly, and the ThreatSentinel correlator. This ensures that the React- based frontend (`App.tsx`) receives a continuously prioritized and correlated feed of infrastructure health, significantly reducing the manual investigative load placed on security engineers.

II. SYSTEM ARCHITECTURE

Cortexia is divided into three distinct operational layers: The Analytical Scanners, the Stateful Correlator (Backend), and the Presentation Layer (Frontend / AI).

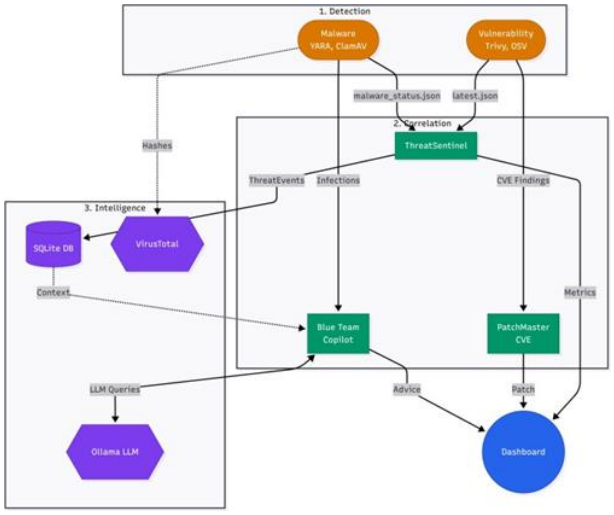

Figure 1: High-level architectural diagram of Cortexia showing the interaction between the APScheduler, Scanners, SQLite Memory, and AI Router.

III. THREAT DETECTION PIPELINE

The core analytical capabilities of Cortexia are divided into distinct deterministic engines that evaluate the filesystem structure.

A. Multi-Engine Malware Analysis

The malware detection pipeline fuses three specialized layers to calculate an aggregate risk score. First, a subprocess execution

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

of `clamscan` evaluates the local file system for known malicious signatures. Second, the custom YARA engine (`scanner.py`) utilizes `gitpython` to dynamica ly clone and pull the `Neo23x0/signature-base` repository on startup. It compiles these community signatures alongside six hardcoded built-in fallback rules (e.g., catching reverse bash shells or PowerShell obfuscation). When scanning, it extracts cryptographic hashes (MD5, SHA-1, SHA-256) and eva luates byte streams, returning exact rule namespaces and malicious payload string offsets. For files flagged by YARA, the system

initiates the third layer: the VirusTotal lookup service. Respecting free-tier API limits via programmatic thread suspension, the module posts the captured SHA-256 hashes to the VirusTotal v3 REST API, retrieving a consensus count of positive engine detections and categorizing the precise malware family.

B. Vulnerability Scanning (OSV & Trivy)

Simultaneously, the vulnerability pipeline leverages `ThreadPoolExecutor` to execute concurrent subprocess routines of `trivy`, `osv-scanner`, and `semgrep`. The system targets specific directories, outputting machine-readable JSON that catalogs Common Vulnerabilities and Exposures (CVEs), the corresponding software package, and the currently installed version versus the securely patched version.

C. Stateful Memory & Threat Correlation

A critical innovation in Cortexia is the translation of ephemeral scan data into persistent, stateful memory instances, bypassing the need for complex, heavy systems like FAISS in favor of a normalized, highly efficient SQLite deployment.

The database schema tracks two primary entities: `incidents` and `patterns`. By generating unique UUIDs for emerging threats, the script tracks the lifecycle status (`open`, `resolved`, `severity`) of every alert. Crucially, the `update_pattern` mechanism allows the system to aggregate duplicate alerts. Instead of flooding the dashboard with repeated YARA matches, the engine increments a `hit_count` integer and updates the `last_seen` timestamp for specific signatures, drastically mitigating alert fatigue.

Operating on a scheduled interval, the logic in `correlator.py` parses the JSON outputs from the diverse scanning layers to assign structured severity levels. For malware evaluations, Cortexia calculates a deterministic mathematical `Risk Score` (R) capped at a maximum severity weight, defined iteratively based on engine hits:

$$R = C + \min(30,5y) + \min(30,3v) \qquad (1)$$

Where $C=40$ if ClamAV detects an infection (else 0), y is the YARA target hit count, and v is the VirusTotal positive engine count. This guarantees that no single heuristic engine monopolizes the severity index.

IV. AUTOMATED REMEDIATION AND AI INTEGRATION

Detection is only the first phase of the incident response lifecycle. Cortexia implements programmatic remediation and natural language processing to accelerate time-to-resolution.

A. PatchMaster Advisor

Implemented within `advisor.py`, this module computationally identifies the underlying package ecosystem responsible for vulnerability. By evaluating the package type of string parsed from Trivy/OSV, the logic maps the CVE directly to actionable terminal commands. The system distinguishes between environments (`pip`, `npm`, `apt`, `yum`, `cargo`, `go`) and synthesizes precise execution strings, such as `pip install --upgra de [package]==[version]`. This entirely eliminates the analyst's need to manua ly research remediation steps for localized dependencies.

B. Context-Aware AI Router (Blue Team Copilot)

To bridge deterministic JSON data and human usability, the backend includes a dedicated AI routing endpoint (ai_router.py). Instead of using generic prompts, the system applies a structured Retrieval-Augmented Generation (RAG) approach. When an analyst submits a query via the React dashboard, the backend reads live outputs from latest.json and malware_status.json, injecting key metrics such as critical vulnerabilities, incident titles, YARA hits, and VirusTotal results into the system prompt. This enriched context is sent to a local LLM (e.g., qwen2.5:7b-instruct) via Ollama, ensuring the Blue Team Copilot delivers accurate, context- aware, and hallucination-free guidance aligned with the system's current state.

V. SYSTEM EVALUATION AND RESULTS

To measure the effectiveness of the autonomous pipeline, Cortexia was deployed against test directories containing obfuscated payloads and vulnerable packages.

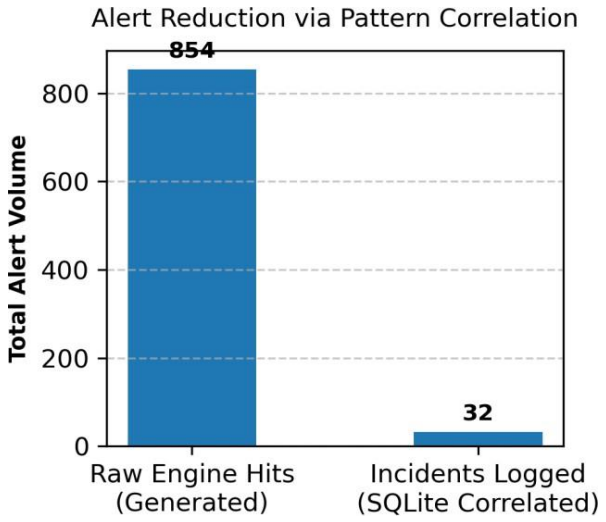

Figure 2: Alert volume reduction through stateful pattern correlation.

The system successfully executed end-to-end cycles without locking the Node/React UI stream. The YARA engine dynamically compiled Neo23x0 community rules and flagged string offsets in under 30 seconds for standard binaries, while the VirusTotal lookup gracefully handled rate-limiting via thread sleeping. Furthermore, the UI accurately rendered the AI Copilot's contextual guidance pulled from the active `latest.json` schemas.

VI. CONCLUSION

The codebase underlying Cortexia demonstrates a highly

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

functional, integrated defense architecture. By algorithmica lly scheduling and fusing static analysis (YARA), public intelligence (VirusTotal), and CVE vulnerability detection (Trivy), the system generates high-fidelity alerts. The deployment of a local SQLite state database prevents the classic issue of unstructured alert flooding. Furthermore, the programmatic synthesis of remediation commands via PatchMaster and the context-injected Blue Team Copilot proves that automation and Large Language Models can be reliably utilized to augment human Incident Response, provided they are strictly bolted to deterministic, code-driven analytics.

ACKNOWLEDGMENT

The authors would like to thank [Aparna Chandran, Asst. Professor] for their support. The authors also acknowledge the use of open-source resources, including the Neo23x0 YARA signature-base and the VirusTotal public API, which provided the foundational intelligence metrics for this autonomous correlation engine.

REFERENCES

[1] K. Bennouk, M. El Yazid, and A. El Omri, “A comprehensive review And assessment of cybersecurity vulnerability detection methodologies,” Journal of Cybersecurity and Privacy, vol. 4, no. 4, p. 40, 2024.

[2] P. Santos, R. Martins, F. Rodriguez, and A. Kumar, “A systematic review of cyber threat intelligence,” Sensors, vol. 25, no. 14, p. 4272, 2025.

[3] A. Dijk, R. Meier, and C. Melella, “Next steps in cyber blue team automation,” in Proc. Int. Conf. Cyber Conflict, 2025.

[4] YARA. "The pattern matching swiss knife for malware researchers". https://github.com/VirusTotal/yara (Accessed 2026).

[5] Ollama. "Get up and running with large language models locally". https://ollama.com (Accessed 2026).

[6] A. Bensaoud, J. Kalita, and M. Bensaoud, “A survey of malware detection using deep learning,” arXiv preprint arXiv:2407.19153, 2024.