CyberShield AI: A Prototype Multi-Agent Framework for Simulated Cyber Threat Detection, LLM-Assisted Risk Analysis, and Explainable Automated Response Recommendations

D.Satya Priya[1], J.Naresh Kumar[2], D. Shravani[3], N.Sri Laxmi[4], V. Sravani[5]
[1,3,4,5] *Student (22N81A6205), Department of CSE (Cyber Security), Sphoorthy Engineering College, Hyd, India*
[2] *Asst.Prof, Department of CSE (Cyber Security), Sphoorthy Engineering College, Hyd, India*
Submitted: March 2026

***Abstract*—This paper presents CyberShield AI, a prototype multi-agent cybersecurity monitoring framework constructed to investigate the feasibility of coordinating modular AI-based components for automated threat detection, risk analysis, and structured response recommendation within a simulated network environment. The system comprises five purpose-built agents — Hunter, Analyst, Responder, Reporter, and Watchdog — organized as a sequential processing pipeline that operates over synthetically generated network access log data. The prototype does not ingest live network streams, execute autonomous remediation actions, or operate against real-world infrastructure; all evaluations are conducted within a controlled simulation environment using a synthetic dataset produced by an integrated log generation module.**

In controlled experimental evaluation, the pipeline processed 1,526 synthetic log records in 2.1 seconds, identifying 458 anomalous entries at a mean detection confidence of 75%. The platform exposes a seven-section Streamlit dashboard providing simulated threat metrics, geographic origin visualization of synthetic attack sources, per-agent conversational interfaces, incident-level SHAP feature attribution, counterfactual scenario analysis, and configurable report generation. A FastAPI REST backend with full OpenAPI documentation and Docker Compose containerization supports repeatable deployment of the prototype. By integrating ensemble machine learning anomaly detection, optional GPT-4-assisted risk narration, structured response planning heuristics, and post- hoc model explainability within a single modular workflow, CyberShield AI demonstrates a defensible design architecture for future research toward operationally viable security automation tooling, while acknowledging the substantial gap between prototype demonstration and production-grade deployment.

***Index Terms*—Multi-Agent Systems, Cybersecurity Prototype, Anomaly Detection, Large Language Models, Explainable AI, SHAP, Response Recommendations, Isolation Forest, Random Forest, Streamlit, FastAPI, Docker, Synthetic Dataset, Modular Pipeline.**

I. INTRODUCTION

The increasing frequency and complexity of cyber intrusions continue to impose considerable operational burdens on Security Operations Centres (SOCs) worldwide. Legacy intrusion detection mechanisms that rely on static signature repositories and deterministic rule engines are structurally constrained from recognizing attack variants that fall outside their catalogued knowledge base. This limitation contributes to escalating alert volumes, analyst desensitization to machine-generated notifications, and extended incident containment timelines in operational environments.

Machine learning offers a technically credible route beyond the architectural ceiling of signature- based defences by enabling detectors to construct behavioral baselines and extrapolate to previously unobserved attack patterns. However, a standalone classification model falls considerably short of the enriched output

that SOC workflows demand. When a detection model classifies an event as anomalous, the analyst responsible for investigation requires substantially more than a binary classification result: contextual understanding of the nature of the detected activity, an assessment of threatened assets, a recommended sequence of remedial actions, and an interpretable explanation of the model's decision process.

CyberShield AI is a prototype system designed to investigate the architectural feasibility of addressing this multi-output requirement through a coordinated ensemble of five modular Python components, referred to throughout this paper as agents. Each agent contributes a distinct analytical layer to a sequential pipeline: anomaly identification, risk evaluation, response plan recommendation, metric aggregation, and execution auditing. It is emphasized at the outset that CyberShield AI constitutes a research prototype evaluated exclusively against synthetic data within a simulated environment. The system does not interface with live network infrastructure, does not execute autonomous remediation actions, and does not constitute a deployable security product. Its purpose is to demonstrate a modular architectural design and to provide a foundation for future research toward operationally viable security automation.

The system is implemented in Python 3.11, exposes an interactive Streamlit dashboard and a FastAPI REST backend, and is containerized using Docker Compose for reproducible prototype deployment. GPT-4 integration via the OpenAI API is optionally available for threat narrative generation but is not a core dependency of the pipeline; the system operates fully in a rule-based fallback mode when API access is unavailable or administratively disabled. The principal technical contributions of this work are as follows:

- A five-stage modular agent pipeline for simulated threat detection, encompassing anomaly hunting, optional LLM-assisted risk analysis, structured response recommendation generation, metric aggregation, and pipeline integrity auditing.
- A Hunter Agent integrating three complementary detection modalities — statistical Z-score analysis, ensemble machine learning using Isolation Forest and Random Forest classifiers, and rule-based pattern matching — with configurable sensitivity thresholds.
- An optional Analyst Agent that constructs structured threat assessment records using OpenAI GPT-4, with a rule-based fallback mode ensuring pipeline continuity in the absence of API access.
- A Responder Agent that generates tiered response recommendation plans from structured incident records, without autonomous execution of any containment or remediation actions.
- An Explainability Viewer providing per-incident SHAP feature contribution visualizations, multi-step reasoning chain summaries, supporting evidence links, and interactive counterfactual scenario analysis for analyst-facing interpretability.
- A seven-section Streamlit dashboard, a FastAPI REST backend with OpenAPI documentation, and Docker Compose deployment for reproducible prototype execution.

The remainder of this paper is structured as follows. Section II surveys relevant prior work spanning conventional cybersecurity tooling, anomaly detection, LLM integration in security contexts, and explainable AI. Section III presents the proposed multi-agent prototype architecture. Section IV describes the implementation methodology, synthetic dataset construction, and technology stack. Section V reports experimental results and discussion. Section VI presents a dedicated limitations analysis. Section VII concludes with a summary of contributions and directions for future research.

II. BACKGROUND AND RELATED WORK

A. Conventional Intrusion Detection Architectures

Rule-based intrusion detection systems (IDS) such as Snort [16] and Suricata [17] operate by matching observed network packets or log entries against curated repositories of previously documented malicious signatures. While these tools provide reliable detection coverage for catalogued attack types, they are architecturally incapable of identifying zero-day exploits or novel attack variants absent from their signature libraries. The static nature of their rule sets means that sufficiently motivated adversaries can systematically probe and circumvent detection boundaries once signature patterns become

known.

Stateful inspection firewalls extended conventional packet filtering by maintaining session context across connection lifecycles, but remained bounded by deterministic rule sets that fail to generalize beyond anticipated threat scenarios. Security Information and Event Management (SIEM) platforms represented a further step by aggregating and correlating multi-source log and event data to surface potential intrusion sequences. However, SIEM deployments are widely associated with high false- positive rates that impose sustained analyst triage burden [18], and their correlation rules require continuous expert maintenance to remain effective as threat landscapes evolve.

CyberShield AI was conceived as a prototype that complements these established tool categories by introducing a machine learning detection layer operating above raw log telemetry, with the explicit goal of reducing the manual enrichment burden through automated analysis recommendations rather than raw alert escalation.

B. Machine Learning Approaches to Anomaly Detection

The application of machine learning to network intrusion detection has been extensively studied over the past two decades. The Random Forest algorithm [2] established early credibility on network intrusion benchmarks through its resilience to noisy and high-dimensional feature spaces, achieved through ensemble bagging over independently trained decision tree classifiers. Its capacity to produce per-feature importance scores provides a natural foundation for explainability, which CyberShield AI exploits through SHAP-based decomposition of Random Forest predictions.

Isolation Forest, introduced by Liu, Ting, and Zhou [3], offered a complementary unsupervised strategy by exploiting the structural observation that anomalous records are few in number and compositionally distinct from the majority population, rendering them disproportionately susceptible to early isolation under recursive random partitioning. This property is well-suited to security telemetry, in which malicious events constitute a small minority of logged activity. The combination of supervised Random Forest classification and unsupervised Isolation Forest anomaly scoring within CyberShield AI's Hunter Agent is designed to provide complementary coverage across labeled and unlabeled anomaly types.

Statistical Z-score deviation analysis provides a lightweight third detection modality for numeric feature distributions in log metadata, flagging records whose feature values deviate substantially from the population baseline under configurable threshold parameters. The integration of these three complementary modalities is intended to reduce the probability that anomalies detectable by one method are systematically missed by the others.

C. Large Language Models in Cybersecurity Contexts

The emergence of transformer-based large language models (LLMs) has introduced a qualitatively distinct capability to security tooling: the capacity to synthesize structured detection outputs into coherent, human-readable threat narratives that non-specialist analysts can interpret without requiring deep familiarity with the underlying model internals. Ali and Kostakos [1] demonstrated through the HuntGPT prototype that combining a Random Forest classifier with GPT-3.5 contextual narration improved the accuracy and confidence of analyst triage decisions relative to working from raw classifier outputs alone. These findings provided direct motivation for the optional Analyst Agent in CyberShield AI, which engages GPT-4 to transform anomaly records enriched with SHAP feature contributions into structured threat assessments.

It is important to note that CyberShield AI's use of GPT-4 is bounded to narrative generation and structured risk classification; the LLM does not perform detection, does not access external data sources during inference, and does not execute any system actions. The quality of LLM-generated threat narratives is inherently contingent on the accuracy of the upstream anomaly detection and feature enrichment stages, and the system's rule-based fallback mode ensures that the pipeline remains functional when GPT-4 access is unavailable.

More broadly, the integration of LLMs into security automation workflows has attracted growing research interest. Sun et al. [9] surveyed LLM and agent-based approaches to security automation, identifying prompt engineering, retrieval-augmented generation, and fine-tuning as promising directions for domain

adaptation. Molleti et al. [10] examined automated threat detection and response using LLM agents, highlighting the importance of structured output formats and fallback mechanisms in operational contexts. CyberShield AI's design reflects these considerations through its structured JSON prompt construction and configurable fallback operation.

D. Automated Response and Security Orchestration

Security Orchestration, Automation, and Response (SOAR) platforms automate incident management workflows through pre-authored response playbooks, reducing the manual overhead of repetitive triage and containment tasks. However, the playbook-driven paradigm is inherently constrained by its reliance on pre-anticipated scenarios; attack types not foreseen during playbook authoring cannot be handled without manual intervention or playbook extension. CyberShield AI's Responder Agent addresses this limitation at the prototype level by dynamically constructing response recommendation plans from the threat profile produced by the Analyst Agent, without requiring every scenario to be pre-scripted. It is critical to note that these plans constitute analyst-facing recommendations only; no automated remediation or system modification is performed by the prototype.

E. Explainable AI in Security Systems

The deployment of machine learning models in security-critical workflows has elevated explainability from an academic research topic to an operational and regulatory requirement. Analysts who receive a binary anomaly flag without supporting rationale face the same interpretive uncertainty as those working with conventional alert systems, and organizations deploying AI in security contexts increasingly face regulatory audit requirements that demand traceable decision rationales.

The SHAP framework [4], grounded in cooperative game theory Shapley value decomposition, provides theoretically principled per-prediction explanations expressed as signed per-feature contribution scores. SHAP has demonstrated effectiveness across tree-based models including Random Forest [2] and gradient boosting [13], and its TreeExplainer implementation provides computationally efficient exact Shapley value computation for these model families. CyberShield AI's Explainability Viewer exposes SHAP outputs at both the model population level and per-incident level, supplemented by counterfactual analysis, evidence links, and multi-step reasoning chains, providing a layered interpretability interface designed for analyst use without requiring knowledge of the underlying mathematical framework.

Ribeiro, Singh, and Guestrin [6] introduced LIME as an earlier model-agnostic local explanation approach; while LIME provides broader model compatibility than SHAP's tree-specific variant, SHAP's theoretical grounding in Shapley values and its exact consistency guarantees make it the more appropriate choice for a system where decision auditability is a primary design objective.

III. PROPOSED MULTI-AGENT PROTOTYPE ARCHITECTURE

A. Architectural Overview

CyberShield AI is structured as a five-agent modular pipeline coordinated by a central Python orchestrator module. The term "agent" is used throughout this paper to denote a discrete, self- contained Python component with a defined input interface, processing responsibility, and output schema; it does not imply autonomous goal-directed behavior, distributed execution, or inter-agent communication beyond the structured state object passed sequentially through the pipeline. This distinction from autonomous multi-agent systems in the artificial intelligence literature is acknowledged explicitly.

Each agent receives the accumulated processing state from its predecessor, contributes its specialized analytical output to that state, and forwards the augmented state to the following stage. The pipeline accepts as input a corpus of synthetic log records retrieved from a SQLite database via SQLAlchemy ORM models, and produces as output a collection of structured anomaly records, incident assessments, response recommendation plans, and aggregated threat metrics, all persisted to the database at pipeline completion. Figure 1 illustrates the high-level data flow through the five- agent pipeline.

Figure 1: CyberShield AI – Five-Agent Sequential Pipeline Data Flow

Stage	Agent	Input	Output
1	Hunter	Raw synthetic log records	Anomaly records with confidence scores
2	Analyst	Anomaly records + SHAP features	Structured incident assessments
3	Responder	Incident assessments	Tiered response recommendation plans
4	Reporter	All pipeline outputs	Aggregated metrics and dashboard data
5	Watchdog	Agent execution metadata	Audit log with performance flags

The pipeline is triggered from the Live Monitoring section of the Streamlit dashboard. Pipeline execution is synchronous within the active web session; there is no background scheduling, streaming ingestion, or persistent daemon process. A FastAPI REST backend exposes all pipeline endpoints and persisted data under a documented OpenAPI schema accessible at localhost:8000/docs, enabling programmatic integration alongside the interactive dashboard at localhost:8501.

B. Hunter Agent

The Hunter Agent constitutes the computational entry point of the pipeline and is responsible for scanning all available synthetic log records to identify entries that deviate from expected behavioral norms under one or more detection modalities. Three complementary detection strategies are implemented, each configurable through the Detection Settings panel within the Settings section of the dashboard.

Statistical Detection

Statistical detection applies a configurable Z-score threshold (default: 3.00) to the numeric feature distributions of log metadata. For each numeric feature in the log schema, the agent computes the population mean and standard deviation across the full log corpus, then flags records whose feature values exceed the threshold multiple of standard deviations from the mean. This modality is computationally lightweight and captures distributional outliers that may indicate unusual session behavior, abnormal data transfer volumes, or atypical login timing.

Machine Learning Detection

Machine learning detection employs two independently trained models. An Isolation Forest model provides unsupervised anomaly scoring by recursively partitioning the feature space using randomly selected split attributes and thresholds; anomalous records, being compositionally sparse and distinct, require fewer partitions to isolate and receive correspondingly lower anomaly scores. A Random Forest classifier trained on labeled synthetic log data provides supervised anomaly classification, producing per-record class probabilities that are used as confidence scores. The ML Contamination Rate parameter (default: 0.10) controls the expected proportion of anomalous records used during Isolation Forest fitting. Both models are trained by executing the src/ml/trainer.py script prior to pipeline deployment and serialized to the models/ directory via joblib for efficient loading at runtime.

Rule-Based Detection

Rule-based detection applies explicit pattern-matching logic for a set of defined attack signature categories. Credential brute-force detection flags records where the number of failed authentication attempts within a session exceeds a configurable threshold (default: 5). Unusual hour detection flags login events occurring before the configurable Unusual Hour Start parameter (default: 2 AM). Additional rules target behavioral anomalies, insider threat indicators based on administrative action patterns, and suspicious access patterns derived from IP reputation and geographic implausibility heuristics.

Upon completion, the Hunter Agent writes each identified anomaly to the anomaly records table in the database with full feature vectors, event timestamps, originating IP addresses, associated user account identifiers, confidence scores, and detection method attributions. In the experimental run documented in this paper, the Hunter Agent scanned 1,526 synthetic log records and identified 458 anomalous entries.

C. Analyst Agent

The Analyst Agent processes each anomaly record produced by the Hunter Agent to generate a structured risk assessment. The agent's behavior is governed by whether GPT-4 API access is enabled in the Settings configuration.

When GPT-4 mode is active, the agent constructs a structured JSON prompt for each anomaly record incorporating the anomaly category, detection confidence score, the top SHAP-derived feature contributions from the trained Random Forest model, the originating IP address, the associated user identifier, and the event timestamp. This prompt is submitted to the OpenAI GPT-4 model via the configured API key using a temperature of 0.2 for response consistency and a maximum token budget of 512 to maintain acceptable processing latency. The model returns a structured assessment comprising a threat narrative describing the probable nature and intent of the detected activity, a severity tier classification mapped to one of four levels (Critical, High, Medium, or Low), and an affected-asset profile identifying the systems, user accounts, and data repositories implicated by the threat.

When GPT-4 mode is disabled or API access is unavailable, the Analyst Agent activates a rule-based fallback mode that maps anomaly category and confidence score to pre-defined severity tiers and generates template-based threat summaries. This fallback mechanism ensures pipeline continuity without external API dependency. In the evaluated experimental run, 458 anomaly records were processed and converted to 458 structured incident entries.

D. Responder Agent

The Responder Agent translates each structured incident record produced by the Analyst Agent into a tiered response recommendation plan. It is essential to emphasize that these plans are analyst- facing recommendations only; the Responder Agent does not interact with any network infrastructure, does not execute firewall rules, does not terminate sessions, and does not perform any automated system modification. All generated plans are written to the database and presented in the dashboard for human review and discretionary implementation.

The agent evaluates each incident's severity tier and threat category through a heuristic tiered logic engine to construct a prioritized action sequence spanning four recommendation phases: Containment (e.g., block the source IP address, isolate the affected endpoint), Eradication (e.g., revoke compromised credentials, remove unauthorized session artifacts), Recovery (e.g., restore affected services, patch exploited vulnerability vectors), and Post-Incident Review (e.g., preserve forensic evidence, update detection signatures). Recommendations for Critical-severity incidents are flagged URGENT and annotated with estimated implementation effort levels to support analyst prioritization.

In the evaluated experimental run, 458 recommendation plans were generated, each linked to its originating anomaly record and incident assessment via shared identifiers in the database schema.

E. Reporter Agent

The Reporter Agent aggregates the outputs of the upstream pipeline stages into structured metrics and populates the Home Dashboard visualization layer. It computes total threat counts segmented by risk tier, daily threat volume timelines derived from event timestamps, attack category frequency distributions, and ranked listings of the most frequently occurring suspicious IP addresses. All computed metrics are persisted to the database at pipeline completion and rendered through Plotly- based interactive visualizations including donut charts, line graphs, horizontal bar charts, and a world map of synthetic attack origin locations derived from IP geolocation data embedded in the synthetic log records.

The Reports section of the dashboard provides a supplementary capability to generate structured narrative documentation in three formats: Executive Summary (a concise non-technical overview suitable for organizational leadership), Technical Deep-Dive (a comprehensive analysis including detection method performance and attack vector distributions), and Compliance Format (an audit-structured document with supporting evidence chains). Reports are rendered in HTML or plain text and scoped to a configurable date range.

F. Watchdog Agent

The Watchdog Agent operates at the meta-level of the pipeline, auditing the execution behavior of the preceding four agents for signs of operational

anomaly. It records each agent's processing duration, output record volume, and error count, and flags deviations from baseline profiles such as agents completing atypically faster or slower than expected, producing zero output records on non- empty input sets, or returning structurally malformed records. The System Logs panel within the Settings section presents the complete timestamped audit trail of every agent execution, enabling retrospective review of pipeline health and performance across all historical runs.

G. Inter-Agent Coordination and Orchestration

The five agents are invoked sequentially by a Python orchestrator module (src/agents/orchestrator.py) that passes accumulated pipeline results through a shared in-memory state dictionary. There is no asynchronous messaging, publish-subscribe communication, or distributed coordination between agents; the pipeline executes as a single synchronous Python function call within the active web session. This design choice prioritizes simplicity, debuggability, and reproducibility for a prototype research system over the concurrency or fault-tolerance features that a production deployment would require.

The Live Monitoring section of the dashboard presents each agent as a status card showing its current operational state (Ready, Running, or Complete with a result summary) and a progress bar for the overall pipeline. Pipeline execution concludes with a summary panel displaying the count of logs processed, anomalies identified, incidents assessed, and recommendation plans generated, alongside the total wall-clock execution duration.

Agent configuration parameters including Z-score threshold, ML contamination rate, failed login threshold, unusual hour start, and detection method activation toggles are exposed in the Detection Settings panel and persisted to the database across sessions. The Alert Simulation panel within Settings supports validation of email notification workflow configuration without dispatching live messages, supporting safe evaluation exercises.

IV. Methodology

A. Synthetic Dataset Construction

CyberShield AI operates exclusively on synthetic network access log data generated by an integrated log generation module (src/data/log_generator.py) included in the project repository. The decision to use a synthetic dataset rather than a publicly available benchmark dataset reflects the prototype nature of the system and the requirement to control class distributions and feature characteristics independently of the constraints of any specific published corpus.

The log generator produces labeled network access records incorporating the following feature attributes: event timestamp, source IP address, user account identifier, session duration in minutes, bytes transferred per session, login hour of day, failed authentication attempt count within session, and six binary indicator features — is_suspicious_ip, is_anomalous_hour, is_failure, is_admin_action, is_data_export, and location_risk score. Geographic anomalies are simulated by associating login events with IP addresses in geographically implausible locations relative to each synthetic user's established behavioral baseline.

The generator supports configurable class imbalance parameters that approximate the structural characteristics of realistic security telemetry, in which benign access events substantially outnumber attack-class records. For the experimental evaluation documented in this paper, the generator produced a corpus of 1,526 log records distributed across the attack categories shown in Table 1. It is emphasized that these records do not represent real network activity; all IP addresses, user identifiers, and behavioral patterns are synthetically constructed.

Table 1: Synthetic Attack Category Distribution in CyberShield AI Evaluation Dataset

Attack Category	Record Count	Primary Detection Modality
Geographic Anomaly	2,728 (full historical)	Rule-based (geographic heuristic)
ML-Detected Anomaly	550	Isolation Forest / Random Forest
Credential Attack	550	Rule-based (brute-force threshold)
Behavioral Anomaly	521	Statistical Z-score
Insider Threat	264	Rule-based (admin action pattern)
Suspicious Access	264	Combined (multi-modality)
Account Compromise	44	Rule-based

A critical limitation of this dataset construction approach is that synthetic data cannot fully replicate the distributional complexity, temporal correlations, adversarial evasion patterns, or noise characteristics of real-world network telemetry. Consequently, the detection performance metrics reported in Section V reflect behavior on synthetic data only and should not be interpreted as indicators of expected performance against production network traffic. This limitation is examined further in Section VI.

B. Model Training and Serialization

Prior to pipeline execution, the machine learning models used by the Hunter Agent are trained by executing the src/ml/trainer.py script against the synthetic log corpus. The Isolation Forest model is fitted using scikit-learn's IsolationForest estimator with the contamination parameter set to the configured ML Contamination Rate (default: 0.10). The Random Forest classifier is fitted using scikit-learn's RandomForestClassifier with default hyperparameter values; no hyperparameter optimization was performed given the synthetic and controlled nature of the evaluation dataset. Both fitted estimators are serialized to the models/ directory using joblib for efficient deserialization at pipeline runtime.

SHAP values are computed using the SHAP library's TreeExplainer applied to the trained Random Forest model. Per-record Shapley value vectors are computed for each anomaly identified by the Hunter Agent and stored in the database alongside the corresponding anomaly record, enabling on- demand rendering of SHAP visualizations in the Explainability Viewer without rerunning the classifier. Population-level feature importance scores are derived from the Random Forest estimator's built-in feature_importances_ attribute.

C. Technology Stack

The complete implementation technology stack is as follows: Python 3.11 as the primary development language; Streamlit (>=1.32.0) for the seven-section dashboard frontend; FastAPI with uvicorn for the REST API backend; SQLAlchemy (>=2.0.0) with aiosqlite (>=0.20.0) for asynchronous database operations; SQLite as the persistence layer; scikit-learn (>=1.4.0) for machine learning model training and inference; SHAP (>=0.44.0) for explainability computation; OpenAI Python SDK (>=1.14.0) for optional GPT-4 integration; Plotly (>=5.20.0) and matplotlib (>=3.8.0) for data visualization; Jinja2 (>=3.1.0) for report template rendering; Faker (>=24.0.0) for synthetic data attribute generation; loguru (>=0.7.0) for structured operational logging; Docker and Docker Compose for containerized prototype deployment; and pydantic (>=2.6.0) with pydantic-settings for configuration management and validation.

Table 2: CyberShield AI Technology Stack Summary

Component Category	Technology / Library	Version
Dashboard Frontend	Streamlit	>=1.32.0
REST API Backend	FastAPI + uvicorn	Latest stable
Database ORM	SQLAlchemy + aiosqlite	>=2.0.0 / >=0.20.0

Persistence Layer	SQLite	Bundled with Python
ML Models	scikit-learn	>=1.4.0
Explainability	SHAP (TreeExplainer)	>=0.44.0
LLM Integration (Optional)	OpenAI Python SDK	>=1.14.0
Visualization	Plotly + matplotlib	>=5.20.0 / >=3.8.0
Containerization	Docker + Docker Compose	Latest stable
Configuration	pydantic + pydantic-settings	>=2.6.0
Synthetic Data	Faker	>=24.0.0
Logging	loguru	>=0.7.0

D. Docker Compose Deployment Configuration

Docker Compose defines two containerized services for prototype deployment. The FastAPI backend service (api) is exposed on port 8000 with hot-reload support via uvicorn and mounts the project source directory as a volume for iterative development. The Streamlit frontend service (streamlit) is exposed on port 8501 and is declared as dependent on the API service, ensuring correct startup sequencing. Both services load environment variables from a .env configuration file, with the SQLite database path and the API base URL injected as container environment variables. The OpenAI API key is optionally configured in the .env file; if absent, the Analyst Agent operates in rule-based fallback mode automatically.

The containerized deployment is intended to support reproducible prototype evaluation in controlled research environments. It does not incorporate production-grade security hardening measures such as API authentication, rate limiting, TLS termination, or secrets management, and should not be exposed to untrusted network environments without substantial additional security engineering.

E. Evaluation Approach

Given the synthetic and controlled nature of the evaluation environment, system performance was assessed across four operationally relevant dimensions rather than through standard classification metrics such as precision, recall, and F1 score. The absence of a ground-truth-labeled held-out test set derived from real network traffic precludes the computation of meaningful generalization performance estimates; reporting such metrics against the synthetic training distribution would risk presenting misleadingly optimistic performance indicators.

Detection throughput was measured as the number of synthetic log records processed per pipeline execution and the wall-clock time required for the complete five-agent pipeline from trigger activation to final summary display. Pipeline latency was recorded as the end-to-end elapsed duration from user-initiated pipeline execution to the appearance of all result metrics in the Live Monitoring section. Per-agent processing durations were individually logged by the Watchdog Agent and exposed in the System Logs panel. Dashboard completeness was assessed through the breadth of information
surfaced at each pipeline stage and the accessibility of configuration and explainability interfaces through the Streamlit UI.

V. RESULTS AND DISCUSSION

A. Pipeline Throughput and Execution Performance

The CyberShield AI pipeline demonstrated consistent processing throughput across repeated evaluation runs against the 1,526-record synthetic log corpus. The complete five-agent pipeline executed in 2.1 seconds from trigger activation to final summary display in the March 24, 2026 evaluation run, with subsequent runs completing in 2.0–2.3 seconds across repeated executions, indicating stable performance within the prototype's operating envelope.

Per-agent processing durations recorded in the System Logs reveal the Hunter Agent as the most computationally intensive pipeline stage, requiring 1,705ms in the referenced run due to the combined execution of Isolation Forest and Random Forest inference alongside Z-score and rule-based analysis over the full log corpus. The Analyst Agent required 40ms in rule-based fallback mode (GPT- 4 was not enabled in the benchmark run to eliminate network latency variability); Responder required 8ms;

Reporter required 3ms; and Watchdog required 0ms (below timer resolution). These durations reflect operation on a synthetic dataset of approximately 1,500 records and should not be extrapolated to larger corpora or production telemetry volumes without further profiling.

Table 3: Pipeline Execution Summary – Benchmark Evaluation Run (March 24, 2026)

Performance Metric	Recorded Value	Notes
Synthetic Log Records Processed	1,526	Full synthetic corpus
Anomalies Identified	458	Across all three detection modalities
Incidents Assessed	458	Rule-based fallback mode (GPT-4 disabled)
Response Plans Generated	458	Analyst recommendations only; no auto-execution
End-to-End Pipeline Duration	2.1 seconds	Wall-clock time, local hardware
Mean Detection Confidence	75%	Across all 458 anomaly records
Hunter Agent Duration	1,705 ms	Dominant pipeline stage
Analyst Agent Duration	40 ms	Rule-based fallback mode
Responder Agent Duration	8 ms	–
Reporter Agent Duration	3 ms	–
Watchdog Agent Duration	0 ms	Below timer resolution

Performance Metric	Recorded Value	Notes
Peak Flagged IP (event count)	185.234.218.21 (884)	Synthetic IP address

B. Anomaly Detection Characteristics

The 458 anomalies identified by the Hunter Agent represent 30.0% of the 1,526-record synthetic corpus, consistent with the class imbalance parameters configured in the synthetic data generator. The dominant anomaly category was unusual_time, reflecting the Unusual Hour Start rule applied to after-hours login activity, which accounted for the majority of high-confidence detections. Confidence scores for the top-ranked anomaly records ranged from 97% to 99%, with the combined multi-method detection approach attributing the highest-confidence classifications to records where two or more of the three detection modalities reached concurrent agreement.

The cumulative threat statistics displayed on the Home Dashboard, drawn from the full historical synthetic dataset across all pipeline runs, show 4,921 total threat records, of which 384 were classified as Critical severity, 3,202 as High, and 335 as Medium, with 75% mean detection confidence across all runs. The Daily Threat Timeline chart shows a detection peak concentrated in the February 23 to March 1 window followed by a declining trend through mid-March, which reflects the temporal distribution of log records produced by the synthetic generator rather than any real network activity pattern.

It is important to reiterate that these figures characterize the system's behavior on synthetic data only. The detection modalities have not been evaluated against independently labeled real-world network traffic, and no claims regarding false positive rates, false negative rates, or generalization performance on unseen real data are advanced by this paper.

C. Explainability Viewer Analysis

The Explainability Viewer provides two levels of model interpretability output derived from the trained Random Forest classifier.

At the population level, the Feature Importance (Random Forest) chart shows that session_duration_min (importance: 0.209), bytes_transferred (0.209), and bytes_log (0.201) constitute the three strongest predictive features for anomalous activity classification in the synthetic dataset, collectively accounting for approximately 62% of total feature importance. login_hour (0.084),

is_suspicious_ip (0.077), and location_risk (0.066) contribute supplementary discriminative signal. The remaining six features contribute importance scores below 0.055 each. This distribution reflects the feature engineering choices embedded in the synthetic data generator and may differ substantially from feature importance distributions observed on real network telemetry.

The Detection Confidence Distribution histogram presents a bimodal distribution with a pronounced cluster in the 0.70–0.75 confidence band and secondary mass at the 1.0 boundary. The 0.70–0.75 cluster corresponds to records flagged by a single detection modality at moderate confidence, while the 1.0 boundary mass represents records where all three modalities reached concurrent agreement and the Random Forest classifier assigned near-certain anomaly probability.

At the per-incident level, the Explainability Viewer populates a Reasoning Chain panel with a multi- step structured account of the detection logic, a SHAP Feature Contributions waterfall chart identifying the dominant contributing feature (bytes_transferred at a raw Shapley contribution of
+67,800 in the illustrated example), an Evidence Links section citing the specific feature values that activated detection thresholds, and a Counterfactual Analysis panel with interactive Streamlit sliders for Failed Attempts, Location Risk Score, Login Hour, and a Suspicious IP indicator toggle. Adjusting these controls dynamically recomputes a composite risk score and updates the Predicted Risk Level display, enabling analysts to explore the sensitivity of the detection boundary and develop practical intuition about the model's decision behavior in the simulated environment.

D. LLM-Assisted Risk Analysis (Optional Mode)

During evaluations conducted with GPT-4 API access enabled, the Analyst Agent successfully processed all 458 anomaly records, generating structured incident entries with threat narratives, severity classifications, and affected-asset profiles. The structured JSON prompt format produced consistent severity tier assignments when submitted with temperature 0.2. The Chat with Agents section of the dashboard exposes a conversational interface allowing analysts to query any of the four named agents in natural language, with pre-configured shortcut queries such as "Why was IP 185.220.101.47 flagged?" and "Explain the brute-force detections" lowering the interaction threshold for non-specialist users.

The GPT-4 integration introduces external API dependency with associated cost, variable response latency (typically 1–5 seconds per record in testing), and availability risk. For the benchmark pipeline execution reported in Table 3, GPT-4 was disabled and the Analyst Agent operated in rule-based fallback mode to isolate pipeline latency from network variability. This deliberate separation illustrates that the core detection and response recommendation pipeline is fully functional without the optional LLM component.

E. Response Recommendation Quality

The 458 response plans generated by the Responder Agent covered the full spectrum of attack categories identified by the Hunter and Analyst agents. Plans for Critical-severity Credential Attack incidents were labeled URGENT and specified concrete source IP addresses as targets for recommended containment actions, demonstrating that the heuristic tiered logic engine successfully propagates structured threat intelligence from the Analyst Agent into actionable recommendation content. Plans for Medium-severity Suspicious Access incidents specified less urgent containment measures with longer recommended implementation timelines.

The quality of response recommendations is fundamentally bounded by the heuristic nature of the Responder Agent's tiered logic engine and the accuracy of the upstream Analyst Agent's severity classifications. No formal evaluation of recommendation quality against expert-defined ground-truth playbooks was conducted; such an evaluation would require domain expert annotation of a representative incident set, which is identified as a priority for future work.

VI. LIMITATIONS

A rigorous representation of CyberShield AI requires explicit acknowledgment of the substantive limitations that bound the scope and generalizability of the results presented in this paper. The following limitations are identified and discussed in detail.

A. Synthetic Dataset Constraints

The most fundamental limitation of the CyberShield AI prototype is its exclusive reliance on synthetic network access log data generated by the bundled log generator module. Synthetic data, by construction, cannot replicate the full distributional complexity of real network telemetry. Real-world network logs exhibit temporal autocorrelation structures, long-tail event distributions, protocol-specific artifacts, adversarially crafted evasion patterns, and noise characteristics that are absent from or only crudely approximated in synthetically generated records. The feature distributions, class imbalance ratios, and inter-feature correlation structures in the synthetic dataset reflect deliberate design choices by the system authors rather than empirical observations from operational network environments.

Consequently, the detection performance characteristics reported in Section V — including the 458 anomalies identified, the 75% mean confidence, and the feature importance distributions — are descriptive of the prototype's behavior on its own synthetic training data only. These figures should not be interpreted as indicators of expected false positive rates, false negative rates, detection coverage, or confidence calibration on real-world network traffic. The system has not been validated on any publicly available intrusion detection benchmark dataset, and such validation constitutes a necessary prerequisite for any claims regarding generalization performance.

B. Absence of Real-World Deployment Validation

CyberShield AI has not been deployed against, or evaluated within, any real network infrastructure or operational SOC environment. The prototype has not been tested with live log ingestion, real network packet streams, production-scale telemetry volumes, or actual adversarial inputs generated by real attack tooling. The system's architecture does not include the log normalization, schema validation, encoding, and sanitization components that would be required to ingest heterogeneous real-world telemetry from diverse network sources.

Bridging the gap between prototype demonstration on synthetic data and operational deployment would require, at minimum: development of telemetry ingestion adapters compatible with standard log formats (e.g., CEF, LEEF, syslog, EVTX); evaluation and recalibration of detection thresholds against real traffic baselines; adversarial robustness testing against evasion-aware attack inputs; and integration with existing SOC tooling stacks. None of these requirements have been addressed in the current prototype.

C. No Authentication or Access Control Layer

The current prototype does not implement any authentication, authorization, or access control mechanism for either the Streamlit dashboard or the FastAPI REST API. The dashboard is accessible to any user with network connectivity to the host on port 8501, and all API endpoints are unauthenticated. The OpenAI API key, if configured, is stored in a plaintext .env file without encryption or secrets management. These characteristics are acceptable for a single-user prototype research system operating in an isolated local environment but render the system unsuitable for deployment in any shared, networked, or untrusted environment without substantial security engineering additions including at minimum: API key authentication, role-based access control, TLS termination, and secrets management infrastructure.

D. Scalability Constraints

The CyberShield AI prototype incorporates several architectural characteristics that impose explicit scalability ceilings incompatible with production telemetry volumes. The SQLite persistence layer is a file-based, single-writer database engine optimized for embedded applications; it does not support concurrent write operations from multiple processes and exhibits significant performance degradation at record volumes exceeding the low tens of thousands. Migration to a multi-writer relational database engine such as PostgreSQL would be a minimum prerequisite for any deployment context requiring concurrent pipeline execution or multi-user dashboard access.

The sequential synchronous pipeline architecture, while appropriate for a single-user prototype, imposes a fundamental throughput ceiling: each pipeline execution processes a static snapshot of the log corpus within a single synchronous function call. High-volume production deployments processing tens of thousands of log events per second would require a streaming pipeline architecture with asynchronous agent execution, message queue-based inter-stage communication, and horizontal scaling of

computationally intensive stages such as ML inference and SHAP computation. SHAP value computation, currently performed synchronously within the pipeline and stored for on- demand retrieval, introduces non-trivial latency for large anomaly sets. The TreeExplainer algorithm scales linearly with the number of records and the number of trees in the Random Forest estimator; for anomaly sets substantially larger than the 458 records in the evaluation corpus, SHAP computation would require either asynchronous background processing or sampling-based approximation to maintain acceptable dashboard response times.

E. Optional LLM Dependency and Quality Bounds

The GPT-4-based Analyst Agent introduces an external dependency on the OpenAI API that carries associated financial cost, variable response latency, and availability risk. The quality of LLM- generated threat narratives is contingent on the accuracy of upstream anomaly detection and SHAP enrichment; if the Hunter Agent produces false positive anomaly records, the Analyst Agent will generate plausible-sounding but fundamentally incorrect threat assessments without any mechanism for self-correction. The rule-based fallback mode mitigates availability risk but produces substantially less informative incident records than GPT-4 narration. No systematic evaluation of the accuracy, consistency, or factual correctness of GPT-4-generated threat narratives has been conducted within this prototype.

F. Response Recommendations Are Not Autonomous Actions

The Responder Agent generates textual response recommendation plans that are presented to human analysts through the dashboard interface for discretionary implementation. The prototype does not implement any mechanism for automated execution of these recommendations; no firewall rule modifications, session terminations, account lockouts, or network configuration changes are performed by the system. The distinction between recommendation generation and autonomous response execution is fundamental to the current prototype's scope, and the absence of autonomous execution is not a temporary limitation to be addressed in future work but a deliberate design boundary reflecting the ethical and operational risks of autonomous security action in prototype systems.

VII. CONCLUSION AND FUTURE WORK

This paper has presented CyberShield AI, a prototype multi-agent cybersecurity monitoring framework that investigates the architectural feasibility of coordinating five modular Python components — Hunter, Analyst, Responder, Reporter, and Watchdog — within a sequential detection pipeline for automated threat identification, structured risk assessment, and response recommendation generation in a simulated network environment. The prototype was evaluated against a synthetic log corpus of 1,526 records, identifying 458 anomalous entries in 2.1 seconds at a mean detection confidence of 75%, and generating 458 structured incident assessments and corresponding response recommendation plans. The Explainability Viewer provides Random Forest feature importance rankings, per-incident SHAP waterfall charts, multi-step reasoning chain summaries, supporting evidence references, and interactive counterfactual scenario analysis, equipping analysts with layered model interpretability support.

The modular agent-based architecture, FastAPI REST backend, Streamlit dashboard, and Docker Compose deployment configuration collectively demonstrate a coherent design template for research into coordinated security automation pipelines. The optional GPT-4 integration illustrates a viable pathway for enriching structured anomaly records with human-readable threat narratives, while the configurable rule-based fallback mode demonstrates that the core pipeline retains functional value independent of LLM availability.

The limitations identified in Section VI — synthetic data dependency, absence of real-world deployment validation, lack of authentication infrastructure, scalability constraints, and the non- autonomous nature of response recommendations — define a clear and substantive research agenda for future work. Priority directions include: integration of live telemetry ingestion adapters compatible with standard log formats and SIEM data sources; evaluation and recalibration of detection models against publicly available labeled intrusion detection benchmark datasets; implementation of streaming

pipeline architecture with asynchronous agent execution for high-volume deployment contexts; migration of the persistence layer from SQLite to a production-grade relational database; incorporation of reinforcement learning feedback mechanisms in the Responder Agent to improve recommendation quality over time from analyst engagement; extension of LLM integration to explore domain-adapted security language models as cost-effective alternatives to general-purpose GPT-4; expansion of detection coverage to cloud-native attack surfaces and API abuse patterns; and rigorous adversarial robustness evaluation of the detection pipeline against evasion-aware synthetic attack inputs and prompt-injection scenarios targeting the LLM integration layer.

CyberShield AI makes no claims to production readiness, real-world deployment viability, or generalization performance on operational network traffic. Its contribution lies in demonstrating a defensible and extensible design architecture for coordinated security automation research, providing a reproducible prototype implementation as a foundation for future empirical investigation, and articulating an honest and detailed characterization of the limitations that must be addressed before such architectures can be considered for operational deployment.

REFERENCES

[1] Ali, S., & Kostakos, P. (2023). HuntGPT: Integrating machine learning-based anomaly detection and LLM-powered explanation for network intrusion detection. arXiv preprint arXiv:2309.16021.

[2] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[3] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), 413–422. https://doi.org/10.1109/ICDM.2008.17

[4] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems (NeurIPS), 30, 4765–4774.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[6] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144. https://doi.org/10.1145/2939672.2939778

[7] OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.

[8] McKinney, W. (2010). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference (SciPy 2010), 51–56.

[9] Sun, P., Zhao, Y., & Chen, J. (2025). AI-augmented SOC: A survey of LLMs and agents for security automation. Computers, 5(4), 95. MDPI. https://doi.org/10.3390/computers5040095

[10] Molleti, R., Reddy, V., & Kumar, A. (2024). Automated threat detection and response using LLM agents. World Journal of Advanced Research and Reviews, 24(2), 079–090.

[11] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357.

[12] NIST. (2018). Framework for improving critical infrastructure cybersecurity. Version 1.1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.CSWP.04162018

[13] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM:A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems (NeurIPS), 30.

[14] FastAPI Documentation. (2024). Sebastián Ramírez. https://fastapi.tiangolo.com

[15] Streamlit Documentation. (2024). Snowflake Inc. https://docs.streamlit.io

[16] Roesch, M. (1999). Snort – Lightweight intrusion detection for networks. Proceedings of the 13th USENIX Conference on System Administration (LISA), 229–238.

[17] Albin, E., & Rowe, N. C. (2012). A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium Workshops, 122–127.

[18] Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. Procedia Computer Science, 88, 300–305.