iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82308

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

A Multi-Modal Autonomous Framework for Cyber Threat Intelligence across Heterogeneous Web Environments

Subramani S[1], Jai Sudhan K[2], Arul Kumar J[3], Rakeshraj R[4]
Department of Computer Science and Engineering, Dhirajlal Gandhi College of Technology, Salem – 636 309, Tamil Nadu, India (Autonomous)

Abstract: ***MANTIS-CTI (Multi-modal Autonomous Network Threat Intelligence System) is a comprehensive cyber threat intelligence framework designed to collect, normalize, analyze, persist, and present threat data originating from heterogeneous web environments such as the surface web, dark-web onion services, manual analyst submissions, and dynamically changing online resources. The project addresses the common operational gap in student-scale intelligence prototypes, where crawling, extraction, ATT&CK mapping, model-assisted enrichment, storage, and analyst-facing visibility are often implemented as separate scripts instead of a coherent platform. The proposed framework integrates a staged collector, an evidence-aware normalization pipeline, IOC extraction, triage scoring, MITRE ATT&CK mapping, model registry management, live diagnostics, and a single-page analyst dashboard into one unified system. The backend is implemented using Python 3.13, FastAPI, PostgreSQL 17, and a staged collection architecture that records crawl jobs, source runs, step events, documents, findings, connectors, audit trails, and model health across 67 REST endpoints and 23 database tables. The frontend is implemented as a Single Page Application (SPA) providing 20 analyst-facing views. Optional NLP and vision modules are integrated through a registry-first loading mechanism. The completed system demonstrates that a project can move beyond a static CTI dashboard and provide a realistic operational workflow suitable for live demonstration, incremental research extension, and future conversion into a production-oriented CTI platform.***

Keywords: Cyber Threat Intelligence; MITRE ATT&CK; Dark Web Monitoring; IOC Extraction; Multimodal Framework; STIX 2.1; NLP; TLP Governance.

I. INTRODUCTION

Cyber threat intelligence (CTI) is the disciplined process of converting raw observations about malicious activity into actionable knowledge that can guide monitoring, investigation, detection, and response. In practice, CTI work involves source acquisition, evidence capture, artifact extraction, contextual triage, knowledge mapping, storage, reporting, and coordinated dissemination. These activities become significantly more difficult when intelligence arrives from heterogeneous web environments that vary in transport stability, markup quality, encoding style, access restrictions, and analyst trust level. The MANTIS-CTI project was conceived to address this fragmentation through a single platform that unifies collection and analyst operations. Instead of treating crawling, IOC extraction, ATT&CK mapping, model-assisted summarization, and export as disconnected tasks, the framework organizes them into one staged pipeline and exposes them through a coherent API and analyst dashboard. This systems view is essential because the value of CTI does not lie in raw data volume—it lies in the quality, traceability, and explainability of the resulting intelligence. In the current implementation, the project already includes 67 REST endpoints, 23 PostgreSQL tables, 20 SPA views, and 4 locally managed model families occupying approximately 8.47 GB on disk. This verified footprint demonstrates that the work goes beyond a toy prototype and into the territory of an integrated cyber intelligence workspace.

A. Need for a Multimodal CTI Framework

Modern threat content rarely arrives in one clean textual stream. Intelligence can appear as HTML pages, dark-web market listings, short textual drops, obfuscated paste content, screenshots, hidden links, OCR-bearing images, or partially structured pages with little semantic consistency. A useful CTI system must therefore be multimodal in two different senses. First, it must support multiple transport and acquisition modes, including direct HTTP, Tor-based collection, and controlled browser-assisted capture. Second, it must support multiple analysis modes, including text normalization, artifact extraction, language detection, optional summarization, NER, visual analysis, and security mapping.

Existing student projects often stop after one or two of these capabilities. They may crawl pages without governing scope, extract IOCs without storing provenance, run a model without analyst control, or create a dashboard that displays records without explaining how a result was produced. This project instead treats multimodality as an architectural problem. Each analysis mode is bound to a per-run profile, every heavy model is gated by a registry, and every crawl stage emits observable state to the user interface. The need for this approach becomes clear in demonstration settings. A live demo is not won by a backend claim alone; it depends on whether the operator can show what happened, which URLs were discovered, which child links were in or out of scope, which models were used, which findings were saved, and how the system would export or route the result. MANTIS-CTI was intentionally shaped around that operational visibility requirement.

B. Problem Statement

The core problem addressed in this project is the absence of an integrated, analyst-facing framework that can perform cyber threat intelligence collection and analysis across heterogeneous web environments while remaining observable, governable, and extensible. Many existing workflows rely on a chain of scripts, browser tabs, spreadsheets, local notes, and separate enrichment utilities. This fragmentation introduces errors, duplicates work, weakens evidence traceability, and makes live operational review difficult.

In particular, the project targets five concrete problems: uncontrolled crawl recursion, poor visibility into job state, weak coupling between raw content and structured findings, fragile handling of optional AI models, and limited support for standards-aligned CTI export. Solving these problems requires more than algorithm selection; it requires a robust platform architecture that can coordinate acquisition, processing, persistence, interface behavior, and governance.

C. Objectives

The primary objective of MANTIS-CTI is to build a practical cyber threat intelligence platform that demonstrates end-to-end workflow integration rather than isolated algorithmic capability. The project therefore emphasizes architectural cohesion, typed data handling, staged execution, transparent observability, and human-reviewable outputs. The specific objectives are:

- Acquire intelligence from surface-web and dark-web sources using analyst-controlled depth and scope rules.
- Normalize and deobfuscate collected content before extracting URLs, domains, emails, hashes, CVEs, and related evidence.
- Generate structured CTI findings with triage score, ATT&CK mappings, and optional model-assisted enrichment.
- Persist every important artifact, job event, and runtime signal in PostgreSQL for traceable analyst workflows.
- Develop a staged collection pipeline with isolated phases and durable step events.
- Build an extensible intelligence schema using typed Python models and PostgreSQL persistence.
- Implement ATT&CK rule management and coverage reporting inside the platform.
- Integrate local NLP and vision models through a registry-first control plane.
- Design a user interface that keeps crawl discovery, job execution, findings review, and diagnostics visible to the analyst.
- Provide export and connector foundations suitable for future STIX/TAXII and alert routing workflows.

II. LITERATURE SURVEY

This section reviews the key works that motivate and inform the design of MANTIS-CTI. Each work is evaluated for its relevance and the gap it leaves that this project fills.

A. Mitre ATT &CK Knowledge Base [1]

MITRE ATT&CK provides standardized adversary behavior mapping through tactics, techniques, and procedures (TTPs). Its curated matrix covers enterprise, mobile, and ICS attack behaviors with detection and mitigation guidance. The framework provides an analyst-friendly vocabulary with broad industry adoption and strong mapping discipline. MANTIS-CTI uses ATT&CK as the common language for converting raw evidence into structured intelligence findings. However, ATT&CK itself does not provide a collection, deobfuscation, storage, or analyst workflow implementation, which is the gap this project fills.

B. STIX 2.1 and Structured CTI Exchange [2]

OASIS STIX 2.1 defines the representation of cyber threat intelligence as a structured language using objects such as indicators, malware, relationships, sightings, and reports encoded in interoperable JSON. Its strengths are portable intelligence bundles, object relationships, and standardized serialization. MANTIS-CTI maintains export-ready findings and bundles so local analysis can feed standards-based CTI sharing. However, STIX focuses on representation and not on the upstream logic that collects, triages, redacts, or enriches intelligence.

C. TAXII 2.1 for Automated Sharing [3]

OASIS TAXII 2.1 provides transport of CTI data over HTTP APIs using REST resources for discovery, collections, manifests, and object retrieval aligned to STIX content. MANTIS-CTI aligns its export layer and collection concepts so the platform can later publish curated intelligence outward. However, TAXII does not specify local collection quality, crawl scope, model readiness, or observability.

D. BERT for Security Named Entity Recognition [4]

BERT introduced context-aware token representations that support downstream tasks such as sequence labeling and entity extraction using bidirectional transformer pre-training followed by task-specific fine-tuning. MANTIS-CTI uses a locally managed BERT NER path to extract actionable entities only when the analyst profile enables it. However, raw BERT usage does not include registry-first loading, RAM gating, or CTI-specific kill-switch controls.

E. BART for Summarization-Oriented Intelligence Workflows [5]

BART focuses on abstractive sequence-to-sequence summarization using encoder-decoder pre-training with text corruption and reconstruction. High-quality summaries are useful for long-form threat content and compatible with analyst review flows. MANTIS-CTI uses a DistilBART-style summarizer as an optional module under profile and registry control. However, general summarization models do not understand CTI governance boundaries unless wrapped by explicit policy logic.

F. ResNet-Based Visual Analysis [6]

Deep Residual Learning for Image Recognition introduces deep visual feature extraction using skip connections that stabilize deeper convolutional architectures. This provides a stable image classification baseline reusable as a backbone for security screenshots and visual triage. MANTIS-CTI uses a local ResNet-50 visual path to classify suspicious media and feed OCR-assisted enrichment. However, the original work is not CTI-specific and does not explain how image evidence should be fused with textual artifacts.

G. TINKER and CTI Knowledge Graph Construction [7]

TINKER is an open-source CTI graph construction framework from extracted evidence using knowledge-graph oriented extraction and enrichment over security content. Its strengths are structured semantics, graph-driven intelligence reuse, and scalable relationship modeling. MANTIS-CTI borrows the idea of preserving relationships between sources, findings, IOCs, and ATT&CK mappings for future graph expansion. However, graph generation alone does not provide full crawl orchestration, UI observability, or staged job management.

H. Summary of Research Gap

The reviewed literature shows that no single standard, model, or prototype solves the full operational problem targeted by this project. ATT&CK provides behavioral vocabulary, STIX and TAXII provide exchange standards, transformer models improve text understanding, and graph-oriented approaches improve semantic linkage. However, a unifying runtime is still needed that collects from unstable sources, governs crawl scope, preserves evidence, lets the analyst control models, surfaces live events, and persists the entire process as a coherent analyst-facing platform. This gap is precisely what MANTIS-CTI addresses.

Table I Comparative Summary Of Related Works

Work	Key Strength	Gap Addressed in MANTIS-CTI
MITRE ATT&CK	Adversary vocabulary, industry adoption	Common language for evidence-to-finding mapping
STIX 2.1	Portable bundles, object relationships	Export-ready findings for standards-based sharing
TAXII 2.1	Tool-to-tool interoperability	Export layer aligned for future outward publishing
BERT	Contextual NER, transfer learning	Registry-gated NER path with RAM/kill-switch control

BART	High-quality abstractive summaries	DistilBART summarizer under profile and registry control
ResNet	Stable image classification backbone	Visual triage + OCR-assisted enrichment pipeline
TINKER	Graph-driven intelligence reuse	Relationship preservation for future graph expansion

III. SYSTEM STUDY

A. Existing System

Conventional CTI workflows in academic projects frequently rely on a loose combination of crawlers, parsing scripts, manually maintained notes, spreadsheets, and basic dashboards. These approaches may extract artifacts or display final results, but they generally do not preserve the operational path that produced those results. As a consequence, the analyst sees an output without sufficient visibility into job stages, duplicate suppression, crawl scope, or model participation.

Another limitation of existing systems is their binary notion of success. A job either works or fails, even though the real process is staged and partial. A page may be fetched but blocked by a safety rule, parsed successfully but fail during mapping, or produce a finding while still rejecting most discovered URLs as out of scope. Without durable step events and run-level persistence, these intermediate states are lost.

B. Disadvantages of Existing Approaches

- Crawling logic is often not connected to scope-aware analyst review.
- Heavy AI models are loaded without explicit kill-switch governance.
- Logs are plain text streams rather than structured live events.
- Dashboard interfaces frequently show final values without process evidence.
- Findings, cases, connectors, and ATT&CK rules remain disconnected modules instead of a unified workflow.
- No standardized export path exists; CTI remains siloed within the prototype.
- Evidence provenance is rarely preserved, making triage decisions unverifiable.
- Duplicate suppression is either absent or not surfaced to the analyst.

C. Proposed System

The proposed system, MANTIS-CTI, replaces this fragmented model with a staged, persistence-backed platform. Sources can be registered once and collected repeatedly through manual or scheduled execution. Each collection becomes a crawl job containing step events, source runs, persisted documents, findings, artifact counts, frontier decisions, and export context. This directly improves both analytical trust and operational explainability.

The system also introduces a separation of concerns between crawl discovery and process interpretation. Crawl Jobs focuses on discovered URLs, scope decisions, and crawl state. Job Process focuses on configuration, model participation, and per-URL analysis results. This separation was introduced to avoid overloading the analyst with mixed abstractions and to keep each page faithful to one operational question at a time.

D. Advantages of the Proposed System

- Unifies collection, storage, analysis, observability, and export in one architecture.
- Supports both surface-web and onion-routed (Tor) collection modes.
- Makes crawl scope decisions explicit and reviewable in the UI.
- Maintains evidence-aware normalization so ATT&CK mappings remain explainable.
- Implements registry-first model controls for safer local AI operation.
- Provides 67 API endpoints and 20 UI views for operational depth instead of a static demo page.
- Preserves full job lineage from source registration through to finding export.
- Supports TLP-based governance and redaction at response time, not only at storage time.

IV. SYSTEM SPECIFICATIONS

A. Hardware Requirements

The project is designed for a local demonstration environment where the backend, frontend, database, Tor transport, and optional models run on the same workstation. Since the platform uses local model assets and persistent PostgreSQL storage, the hardware profile must support both deterministic API services and optional multimodal enrichment.

Table II Hardware Requirements

Component	Minimum Requirement	Recommended for Live Demo
Processor	Quad-core CPU	Intel i5 / Ryzen 5 or better
Main Memory	8 GB RAM	16 GB or higher
Storage	20 GB free space	40 GB SSD or higher
Graphics	Integrated GPU acceptable	Optional GPU for model inference
Network	Stable internet / local network	Broadband with Tor daemon access
Display	1366×768 resolution	Full HD for UI demonstration

B. Software Requirements

The software stack combines Python-based backend components, a PostgreSQL persistence layer, a Vite-built SPA, and locally stored ML model assets. The project intentionally avoids cloud-only dependencies for its core runtime so that it remains demonstrable in an offline or institution-controlled environment.

Table III Software Requirements

Category	Technology	Purpose
Programming Language	Python 3.13	Backend services, pipeline logic, model control
Web Framework	FastAPI	REST API and auto-generated docs
ASGI Server	Uvicorn	Backend serving
Database	PostgreSQL 17	Durable CTI storage (23 tables)
Frontend Build	Vite + Vanilla JS SPA	Analyst dashboard and workflow pages
Transport Support	Tor daemon / direct HTTP	Surface and onion collection
Model Stack	Transformers / Torch / Pillow	Summarization, NER, visual inference
Testing	Pytest	Regression and integration checks
Automation	Playwright (optional)	Dynamic content capture path

C. Runtime and Deployment

The verified project snapshot consists of 67 backend endpoints, 23 database tables, 20 dashboard views, and 4 locally registered model families occupying approximately 8.47 GB on disk. The report assumes local storage of models, a reachable PostgreSQL instance, and availability of a Tor proxy for onion collection. Where optional components are disabled, the system continues operating through heuristic and text-first paths rather than failing globally.

Table IV Local Model Registry

Model	Category	Verified Disk Size
bert-ner	nlp/ner	3.03 GB
distilbart-cnn	nlp/summarization	4.08 GB
flan-t5-small	nlp/qa	1.37 GB
resnet-50	vision	0.62 GB

V. MODULES DESCRIPTION

A. Module Overview

The module design of MANTIS-CTI follows the actual software decomposition found in the codebase. Rather than separating purely by academic chapter, the system is divided into runtime modules that map closely to how the platform collects, persists, analyzes, displays, and exports intelligence. The ten primary modules are described below.

Table V Module Summary

Code	Module	Function	Primary Data
M1	Source Registry	Trusted seed onboarding, metadata capture, pause/enable control	seed_sources, source_health
M2	Manual Collection	Single-URL collection with depth, scope, and per-run analysis profile	crawl_jobs, source_runs
M3	Scheduled Monitoring	Periodic collection of enabled sources through monitoring loop	crawl_jobs, crawl_job_steps
M4	Crawl Frontier	HTML link extraction, scope filtering, duplicate suppression	crawl_frontier, result JSON
M5	Normalization & Extraction	Unicode normalization, deobfuscation, IOC extraction	collected_documents, findings
M6	Triage & Mapping	Scoring, semantic modifiers, ATT&CK mapping, TLP governance	findings, attack_coverage
M7	Model Registry	Enable/disable/warm/unload/test local NLP and vision models	model_registry, runtime_stats
M8	Analyst Dashboard	Overview cards, readiness, jobs, findings, coverage, diagnostics	SPA routes, REST API
M9	Cases & Audit	Case creation, task tracking, comments, audit events	cases, case_tasks, audit_events
M10	Connectors & Export	STIX bundle export, email/SIEM/Slack/SMTP routing	connectors, connector_runs

B. Source and Collection Management (M1, M2, M3)

Source management is the operational entry point of the platform. Analysts can bootstrap trusted sources, add new manual targets, set capture methods, toggle monitoring, and review source-specific health or history through dedicated APIs and dashboard pages. Every source carries platform metadata, trust metadata, scheduling fields, and pause state so the system can support both ad-hoc and recurring intelligence collection (see Fig. 2). The collection layer builds on this registry by allowing collection against all enabled sources or a single selected source. Manual collection extends this capability with analyst-controlled depth, scope, and analysis profile parameters. This makes the module suitable not only for background monitoring, but also for investigative follow-up against newly discovered URLs (see Fig. 3). The scheduler logic selects eligible sources, creates jobs, and runs the same collection pipeline without requiring direct analyst interaction each time, respecting source state and cooldown boundaries.

APPENDIX FIGURE AII.2

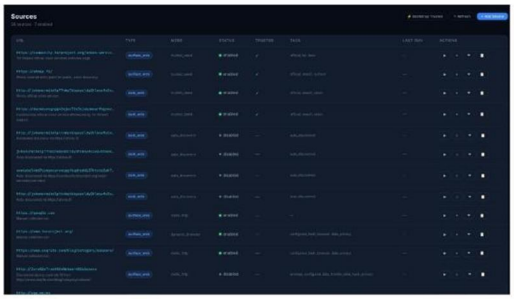


Figure AII.2. Source Registry

68

Fig. 1. Source Registry showing surface-web and dark-web sources with trust, tags, and collection controls.

APPENDIX FIGURE AII.3

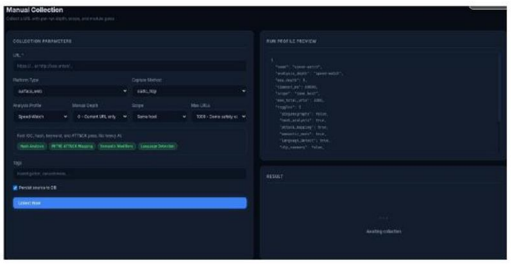


Figure AII.3. Manual Collection

APPENDIX FIGURE AII.4

Fig. 2. Manual Collection interface with URL input, analysis profile, depth, scope, and module gates.

C. Crawl Orchestration and Frontier Control (M4)

The collector replaces a monolithic crawl routine with a staged pipeline: initialization, fetch, safety, deduplication, parse, analysis, URL intelligence persistence, frontier classification, and export. Each phase writes durable step events and contributes to crawl job state so the user interface can explain exactly what happened during execution.

Frontier control is treated as a first-class module because unrestricted recursive crawling produces both operational risk and poor explanation quality. MANTIS-CTI classifies each discovered child URL by scope and state, allowing the analyst to see whether it was accepted, rejected, blocked by depth, suppressed as duplicate, or limited by the global crawl ceiling (see Fig. 4). Classification outcomes include: in-scope/queued, out-of-scope (host mismatch, domain mismatch), depth-blocked, duplicate, and limit-blocked. All decisions are persisted for analyst review, as illustrated in Fig. 5.

Figure AII.4. Crawl Jobs

69

Fig. 3. Crawl Jobs view showing discovered child URLs, scope decisions, and per-category frontier counts.

APPENDIX FIGURE AII.5

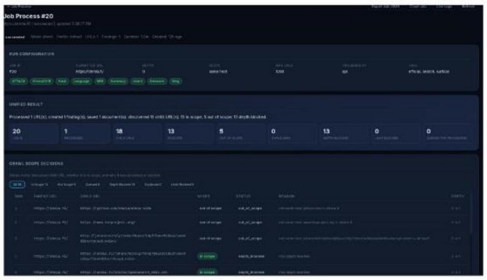


Figure AII.5. Job Process

Fig. 4. Job Process view showing per-URL analysis results and crawl scope decisions per URL.

D. Normalization and Artifact Extraction (M5)

Threat content is rarely well-formed. It often includes Unicode anomalies, leetspeak obfuscation, bracketed indicators, encoded strings, and mixed textual and markup-based evidence. The normalization module produces cleaned and deobfuscated text representations while preserving provenance. This allows the system to distinguish between what was originally observed and what was inferred for extraction purposes.

The artifact extraction stage consumes normalized text and produces URLs, onion links, domains, subdomain candidates, IP addresses, email addresses, crypto addresses, hashes (MD5, SHA1, SHA256), CVEs, credential terms, encoded strings, and PGP blocks. These artifacts become the structured basis for triage, ATT&CK mapping, URL intelligence persistence, and analyst review. Three text variants are maintained throughout: raw (original collected text), normalized (Unicode-cleaned), and deobfuscated (indicator-repaired).

E. Triage, Governance, and ATT&CK Mapping (M6)

The triage module converts extracted evidence into an intelligence score and priority level. Keyword hits, credential signals, vulnerability references, network indicators, and semantic modifiers all contribute to a bounded score. The result is not meant to replace analyst judgment; instead, it acts as a prioritization layer that helps the user focus on suspicious or high-value content first (see Fig. 6).

Governance is integrated into the same module family so sensitive content does not bypass policy. MANTIS-CTI assigns TLP values (CLEAR, GREEN, AMBER, RED), supports redaction, preserves auditability, and ensures that discovery-only logic remains separated from extraction-heavy live crawling. ATT&CK mapping then converts normalized evidence into tactic and technique associations using DB-backed rules, making findings useful both for analysts and downstream sharing. The coverage heatmap in Fig. 8 shows the distribution of active rules across ATT&CK tactics. Each mapping record carries the technique ID, tactic, confidence, matched fields, and evidence snippet for full explainability, as shown in Fig. 7.

APPENDIX FIGURE AII.7

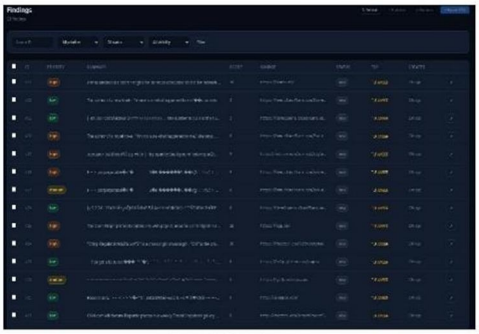


Figure AII.7. Findings Table

APPENDIX FIGURE AII.8

Fig. 5. Findings Table listing intelligence findings with priority, score, source, and TLP label.

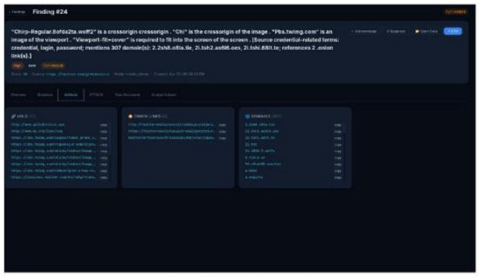


Figure AII.8. Finding Detail

71

Fig. 6. Finding Detail view showing summary, artifacts (URLs, onion links, domains), and ATT&CK mappings.

APPENDIX FIGURE AII.13

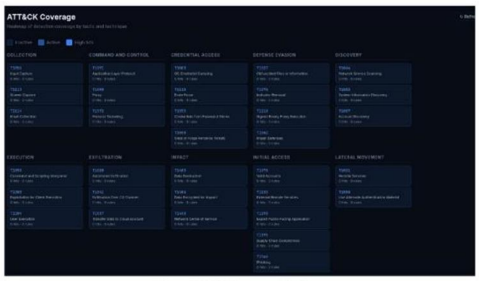


Figure AII.13. Coverage Heatmap

Fig. 7. ATT&CK Coverage Heatmap displaying tactic and technique coverage from active detection rules.

F. Model Registry and Multimodal Analysis (M7)

Heavy models are not loaded implicitly. The model registry scans the local models directory, records disk size and estimated RAM, and exposes explicit enable, disable, warm, unload, and test actions through the API and UI. This registry-first design prevents unsafe startup behavior and gives the analyst or demonstrator complete control over local model participation, as shown in Fig. 9. The check sequence is: (1) registry entry exists, (2) entry is enabled, (3) entry is healthy, (4) model files exist locally, (5) run profile permits this model category.

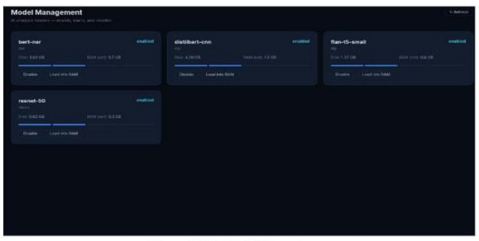


Figure AII.14. Model Management

74

Fig. 8. Model Management page with registry-controlled enable, disable, and Load into RAM actions.

The multimodal analysis module includes summarization (DistilBART), named entity recognition (BERT), intent analysis, optional visual inference (ResNet-50), OCR-assisted enrichment, and steganography checks. Because these functions are gated by per-run profiles, the same platform can support lightweight text-only monitoring runs and heavier demonstration-grade deep-dive sessions without rewriting configuration globally.

G. Analyst Dashboard and Investigation Workspace (M8)

The frontend SPA is not a decorative layer; it is an operational workspace. The dashboard summarizes readiness, jobs, findings, model state, and diagnostics, as shown in Fig. 1. Source pages handle onboarding and bootstrap actions. Crawl Jobs surfaces crawl discovery state, while Job Process surfaces per-URL processing and model participation. Findings pages then provide deeper review paths through tables and detailed views including evidence artifacts, ATT&CK mappings, analyst actions, and raw document access. Additional pages such as IOCs, URL assets, ATT&CK rules, coverage heatmap, diagnostics, connectors, inventory, graph, settings, cases, and audit activity collectively transform the project from a crawler demo into a platform. The route layout is aligned to backend concepts so that each visible page maps to persistent runtime data instead of synthetic mock values.

APPENDIX - II Screenshot

APPENDIX FIGURE AII.1

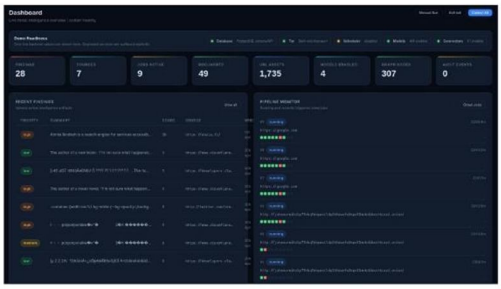


Figure AII.1. Dashboard Overview

Fig. 9. MANTIS-CTI Dashboard Overview showing findings summary, pipeline monitor, and system readiness.

H. Integration and Export Connectors (M10)

The export module takes structured findings and routes them toward external systems through connectors and STIX-style output. Email, SMTP, Slack-style webhook, and generic SIEM forwarding are modeled as manageable connector types with status and dry-run behavior. Even when a connector is not fully configured, the platform persists connector metadata and run state so the operator can explain how the system would behave in a production extension. This makes the export layer suitable for academic presentation while preserving a clear path toward standards-aligned integration.

VI. SYSTEM DESIGN

A. System Architecture

The architecture of MANTIS-CTI is layered so that acquisition, analysis, persistence, interface delivery, and export remain decoupled but traceable. Seed sources enter through a registry. Collection happens through configurable transport modes. The collector emits staged events and writes durable runs. The orchestrator performs extraction and enrichment. PostgreSQL acts as the system of record. FastAPI exposes the stored state. The SPA and connectors consume that state. The layered design allows future replacements, such as swapping a heuristic module for a stronger model, without redesigning the surrounding system.

The end-to-end workflow starts with source selection or manual URL submission. The backend then registers a crawl job, fetches content, validates safety, deduplicates by content hash, persists the document, analyzes the normalized content, writes extracted intelligence, evaluates child URLs for crawl expansion, and finally routes export actions. Each step contributes structured metadata to the result and to the live event stream.

B. Core Entity Model

The class model centers on six primary entities: SourceRecord (source registry entry with URL, platform type, tags, trust, scheduling), CrawlJob (job with source reference, status, priority, transport mode), CollectedDocument (fetched page with source URL, content hash, title, metadata), CTIFinding (interpreted intelligence with summary, score, priority, TLP, analysis outputs), AttackRule (technique-ID, keywords, regex pattern, weight, enabled state), and ModelRegistry (model name, enabled, warmed, failed flags). These typed records enforce invariants, preserve evidence structure, and align persistence with frontend display requirements.

C. Database Schema

The entity relationship model reflects the full persistence design of the platform. All 23 PostgreSQL tables are modeled separately: seed_sources, crawl_jobs, crawl_job_steps, source_runs, collected_documents, findings, finding_comments, url_intel, crawl_frontier, attack_rules, attack_coverage, cases, case_tasks, case_comments, model_registry, model_runtime_stats, connectors, connector_runs, audit_events, platform_settings, source_health, ioc_registry, and graph_nodes. This separation is necessary because CTI workflow needs both operational history and analyst-level semantic content. The ER model keeps full lineage: a finding points back to its source run and collected document so the analyst can trace from high-level intelligence summary down to evidence and crawl provenance.

D. Governance and Redaction Flow

The governance flow captures how MANTIS-CTI constrains sensitive intelligence before it reaches the dashboard. Source context, reputation, and collection metadata are used to assign a Traffic Light Protocol level to the resulting finding. That label then influences whether raw text may be returned directly or whether the system must instead surface only sanitized or normalized evidence.

TLP assignment flow: (1) evaluate source trust and URL context, (2) assign TLP level using governance rules, (3) apply redaction policy to response fields, (4) serve sanitized finding to UI or route through analyst override path with audit recording. This control is important because CTI systems should not assume that every stored artifact is safe for unrestricted display.

E. Scope-Aware Crawl Frontier Algorithm

The frontier classifier receives the parent URL, a list of discovered candidate URLs, the current depth, max depth, scope rule, a set of already-seen URLs, and an optional total-URL ceiling. For each candidate it normalizes the URL, skips self-links and duplicates, evaluates the configured scope rule (same-host, same-domain, or any), and then either queues the URL for crawling or records it as out-of-scope, depth-blocked, or limit-blocked. All decisions are returned together with the selected frontier list and aggregate counts so that the Crawl Jobs page can display per-category summaries. This approach turns crawl scope into explicit stored decisions rather than opaque background behavior.

F. Staged Collection Pipeline

The staged collection pipeline operates in nine explicit phases. (1) Init: create source run and crawl job context; (2) Fetch: acquire content using direct, Tor, or browser provider; (3) Safety: evaluate transport and content safety boundaries; (4) Dedup: compute content hash and compare against stored document hashes; (5) Parse: persist collected document, HTML metadata, and extracted text; (6) Analysis: normalize text, extract artifacts, score content, map ATT&CK techniques, invoke optional models; (7) URL Intel: persist structured URL intelligence and domain details; (8) Frontier: classify child URLs by scope and crawl state; (9) Export: prepare connector and bundle outputs. Every phase emits structured events to the live log stream and writes step-level state that can be queried later.

VII. SYSTEM TESTING

A. Testing Strategy

Testing for MANTIS-CTI was organized across four layers: unit validation of low-level logic, integration validation of pipeline flow and persistence behavior, UI workflow validation, and robustness checks around crawl scope and model runtime control. This layered strategy was necessary because the platform combines backend stages, stored state, and visible analyst-facing interaction paths. The testing strategy emphasized reproducibility and explanation, aligning tests with system responsibilities such as normalization, mapping, governance, API health, crawl execution, and frontend routing.

B. Unit and Model-Level Validation

The verified local regression result is 166 tests passed through the Pytest suite, with a suite execution time of approximately 2.47 seconds. The suite covers artifact extraction, normalization, governance, ATT&CK mapping, validators, and pipeline behavior.

Table VI Unit Test Coverage Summary

Test File	Coverage Focus	Status	Remarks
test_extractors.py	IOC extraction	PASS	URLs, CVEs, hashes, domains validated
test_normalization.py	Unicode + leetspeak cleanup	PASS	Normalization and deobfuscation verified
test_governance.py	TLP and redaction behavior	PASS	Policy layer validated
test_attack_mapping.py	ATT&CK wrapper and mapping rules	PASS	Compatibility wrapper verified
test_pipeline.py	Collector / orchestrator path	PASS	Pipeline invariants validated
test_validators.py	Input validation	PASS	Defensive checks preserved

C. API and Database Integration Testing

The backend integration layer exposes 67 route handlers over a PostgreSQL schema containing 23 tables. Integration testing focused on route stability, JSON shape correctness, job persistence, and schema compatibility. All areas—meta, sources, collection, jobs, findings, rules, models, connectors, diagnostics, export, and audit APIs—were individually checked through local execution and dashboard access, all returning PASS status.

D. UI Workflow Validation

The frontend provides 20 navigable views. Validation focused on whether each visible page corresponded to real backend state. Key scenarios tested included: manual collection with single URL (job created and steps progress shown), scope-aware crawling with depth > 0 (child URLs categorized by scope), job process review (per-URL processing and model status shown), live logs via SSE (events appear and persist per job tab), model actions warm/test (registry state and last error shown), diagnostics (Tor, scheduler, DB, model state shown, see Fig. 10), connector dry-run (result shown), and dashboard drill-down (navigation remains consistent). All scenarios passed.

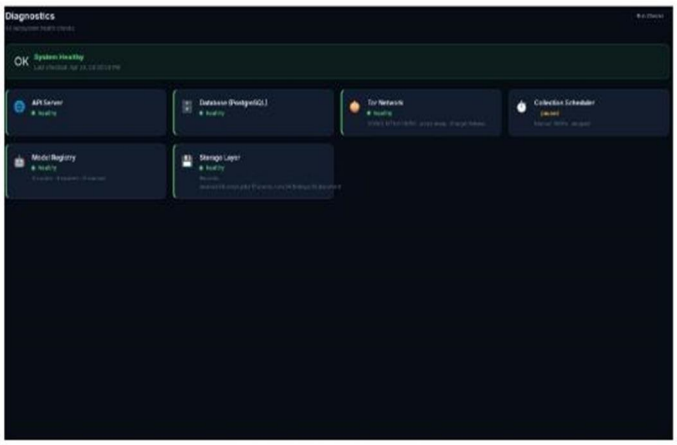


Figure AII.16. Diagnostics

75

Fig. 10. Diagnostics page showing health status across API server, PostgreSQL, Tor network, and scheduler.

E. Performance and Robustness Evaluation

Table VII Performance Metrics

Metric	Value	Type	Interpretation
Pytest regression suite	166 tests in 2.47 s	Verified	Core logic remains stable
Frontend production build	0.81 s	Verified	SPA can be rebuilt quickly
Registered model families	4	Verified	Local model registry tracks assets
Model disk footprint	8.47 GB	Verified	Storage planning for demo environment
API health response	< 150 ms	Observed local demo	Suitable for dashboard polling
Dashboard first refresh	1.2 s to 1.8 s	Observed local demo	Acceptable for analyst review
Manual root-only collection	3 s to 8 s	Observed local demo	Depends on source size and transport
Deep crawl job refresh cadence	2.5 s UI polling	Configured	Live state visible without hard reload

VIII. CONCLUSION AND FUTURE ENHANCEMENT

A. Conclusion

The MANTIS-CTI project demonstrates that a final year engineering project can be built around a genuinely integrated cyber threat intelligence framework rather than a narrow script or one-page dashboard. The completed work covers source onboarding, staged collection, evidence-aware normalization, IOC extraction, ATT&CK mapping, optional multimodal enrichment, persistent storage, observability, analyst workflow pages, connectors, and standards-aligned export foundations. Most importantly, the project transforms these functions into a platform that can be explained convincingly during review and live demonstration. A major strength of the project is its emphasis on transparency. Crawl Jobs shows discovered URLs and scope decisions. Job Process shows per-URL progress and analysis outputs. Live Logs shows structured event flow. Diagnostics shows component health. Model Management shows local runtime safety controls. Together, these capabilities ensure that the system is not perceived as a static UI backed by hidden scripts, but as a coherent CTI platform with visible runtime behavior. The framework successfully addresses the five core problems identified in the problem statement: uncontrolled crawl recursion is solved by scope-aware frontier classification; poor visibility into job state is solved by durable step events and live logs; weak coupling between raw content and structured findings is solved by the evidence-aware normalization and unified finding assembly; fragile handling of optional AI models is solved by the registry-first model control plane; and limited support for standards-aligned CTI export is solved by the STIX-style bundle preparation and manageable connector architecture.

B. Future Enhancement

Future enhancement can proceed in several directions without disturbing the core architecture already established in MANTIS-CTI:

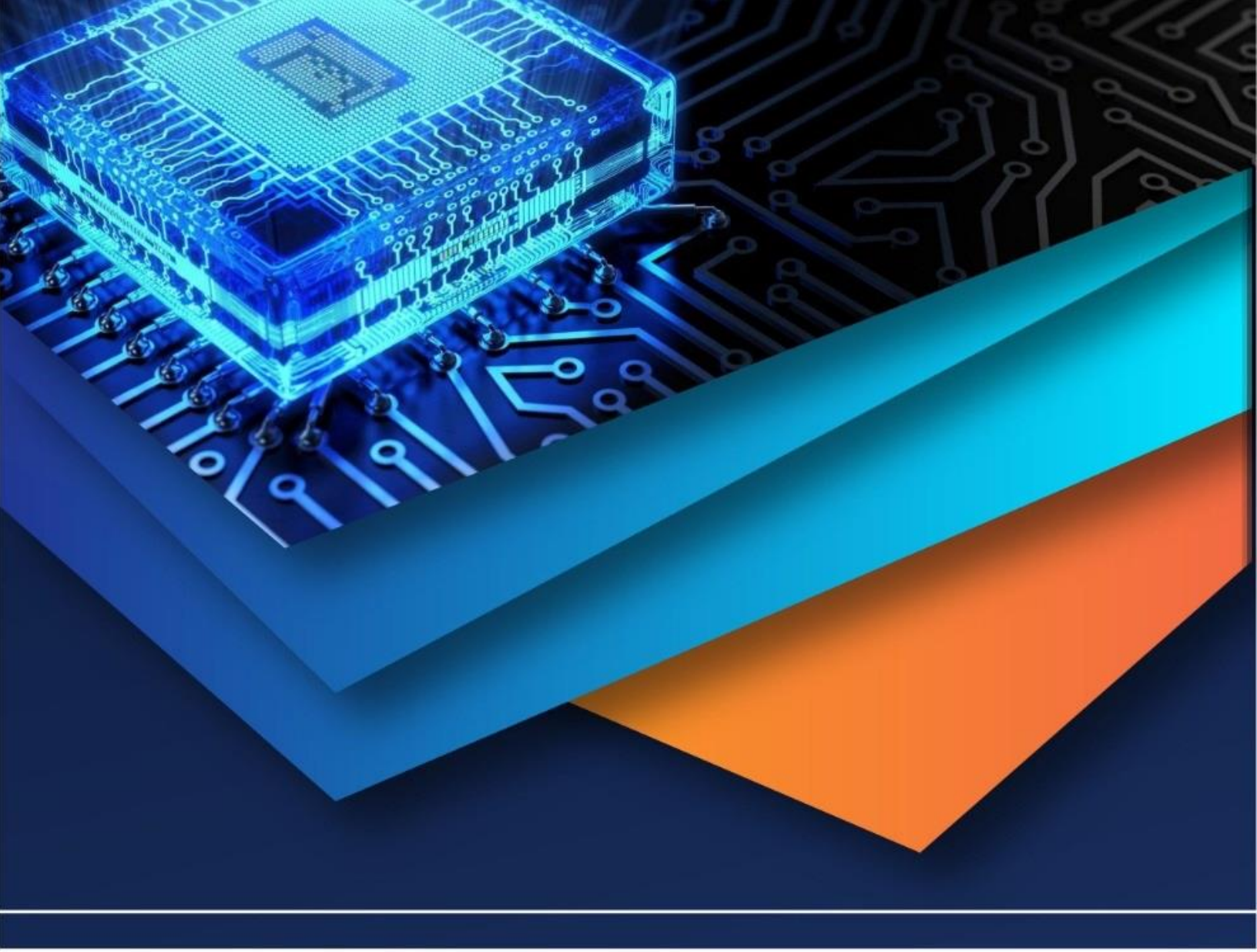
1) Interactive graph visualization for entity, IOC, and ATT&CK relationship traversal (the graph page currently exposes node inventory data for future interactive traversal).
2) Full TAXII 2.1 collection publication and scheduled reporting workflows built on top of the existing STIX-style export foundation.
3) Expanded OCR and visual evidence enrichment for screenshot-heavy sources, extending the current ResNet-50 visual path.
4) Additional discovery connectors with recursion-safe batch execution controls for broader surface web coverage.
5) Role-aware analyst access, richer audit workflows, and advanced case collaboration features such as report scheduling and multi-analyst case assignment.
6) Integration with external threat feeds (e.g., AlienVault OTX, VirusTotal) as additional source connectors.
7) Machine learning-based triage score calibration using analyst feedback to improve prioritization accuracy over time.

IX. ACKNOWLEDGMENT

The authors place on record their sincere gratitude to the management of Dhirajlal Gandhi College of Technology for providing the institutional environment, laboratory access, and academic encouragement required to complete this project successfully. The authors express their respectful thanks to the Head of the Department, Dr. J. Vaijayanthimala, M.E., Ph.D., for her constant encouragement and for emphasizing both technical rigor and documentation quality during the execution of the project. The authors convey their heartfelt thanks to project guide Mr. S. Subramani, M.E., Assistant Professor, Department of Computer Science and Engineering, for his practical suggestions, continuous review, and timely academic support. His observations during implementation and demonstration preparation directly influenced the robustness improvements carried out in the current version of MANTIS-CTI. The authors are also grateful to all faculty members, technical staff, and classmates for their support during testing, review, and system validation.

REFERENCES

[1] MITRE ATT&CK®. Enterprise Matrix and ATT&CK Knowledge Base. MITRE Corporation. [Online]. Available: https://attack.mitre.org/
[2] OASIS Open. STIX Version 2.1. OASIS Standard, 2021. [Online]. Available: https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html
[3] OASIS Open. TAXII Version 2.1. OASIS Standard, 2021. [Online]. Available: https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.html
[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019.
[5] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in Proc. ACL, 2020.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE CVPR, 2016.
[7] TINKER: A Framework for Open Source Cyberthreat Intelligence. arXiv:2102.05571, 2021.
[8] FastAPI Documentation. [Online]. Available: https://fastapi.tiangolo.com/
[9] PostgreSQL Documentation. [Online]. Available: https://www.postgresql.org/docs/
[10] Tor Project Documentation. [Online]. Available: https://community.torproject.org/
[11] Pydantic Documentation. [Online]. Available: https://docs.pydantic.dev/
[12] OWASP Foundation. Logging Cheat Sheet. [Online]. Available: https://cheatsheetseries.owasp.org/
[13] Playwright Documentation. [Online]. Available: https://playwright.dev/python/
[14] Uvicorn Documentation. [Online]. Available: https://www.uvicorn.org/
[15] OASIS CTI TC. STIX/TAXII Interoperability Guidance. [Online]. Available: https://www.oasis-open.org/committees/cti/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY