

AIR Blackbox Compliance Report

CrewAI Framework Analysis

Framework	CrewAI (crewAIInc/crewAI)
Scanner Version	air-blackbox v1.2.2
Scan Date	March 13, 2026
Articles Covered	EU AI Act Articles 9, 10, 11, 12, 14, 15
Detection Method	95% automated, 5% hybrid/manual
License	MIT (CrewAI) / Apache 2.0 (AIR Blackbox)

15 Passing	11 Warnings	11 Failing	37 Total Checks
---------------	----------------	---------------	--------------------

Executive Summary

CrewAI is a leading open-source multi-agent orchestration framework with 1,015 Python files. This report evaluates CrewAI against EU AI Act requirements across 6 articles using the AIR Blackbox compliance scanner. CrewAI demonstrates notably strong security and human oversight patterns for a framework of its size.

The standout finding is CrewAI's dedicated security module with agent fingerprinting (UUID-based identity tracking with metadata validation and DoS protection), built-in guardrails (including a hallucination guardrail and LLM output guardrails), and a comprehensive Agent-to-Agent (A2A) protocol with authentication, TLS verification, and agent card discovery. These patterns are among the most mature we've seen across any framework scanned.

CrewAI also shows exceptional Pydantic adoption (384/1,015 files, 38%) and human-in-the-loop patterns (31 files), driven by its crew delegation architecture where agents can delegate tasks to other agents with explicit permission controls. Article 14 (Human Oversight) was the strongest area with 6/9 checks passing.

Key Findings (Code-Level)

Metric	Value	Assessment
Python files scanned	1,015	Focused, single-purpose codebase

Input validation (Pydantic)	384 files (38%)	PASS
Human-in-the-loop patterns	31 files	PASS
Prompt injection defense	65 files	PASS
Tracing / observability	72 files + event bus architecture	PASS
Fallback / recovery patterns	107 files	PASS
Budget / rate limiting	70 files (max_rpm, max_execution_time)	PASS
Retry / backoff logic	60 files	PASS
PII-aware patterns	7 files	PASS
Output validation	52 files (output_pydantic)	PASS
Agent delegation controls	54 files (allow_delegation)	PASS
Built-in guardrails	25 files (hallucination + LLM guardrails)	PASS
Security module (fingerprinting)	Dedicated module with UUID identity + metadata	PASS
Agent action audit trail	15 files + event bus	PASS
LLM call error handling	77/113 files (68%)	WARN
Docstrings coverage	4,316/8,558 (50%)	WARN
Type annotations	2,440/6,505 (38%)	WARN
Structured logging	100/1,015 files (10%)	WARN

Article 9 - Risk Management

Check	Status	Type	Evidence
Risk assessment document	FAIL	Human	No RISK_ASSESSMENT.md found
Risk mitigations active	FAIL	Human	0/4 mitigations active. Infrastructure-level requirement
LLM call error handling	WARN	Auto	77/113 files (68%). Missing in security module, agent adapters, and RAG components
Fallback/recovery patterns	PASS	Auto	Fallback patterns found in 107 files. Strong recovery architecture

Analysis: CrewAI's 68% error handling coverage is the highest we've seen across frameworks scanned. The built-in guardrails (hallucination detection, LLM output validation) are genuine risk mitigations that most frameworks lack entirely. The security module with DoS protection in metadata validation shows security-first thinking.

Article 10 - Data Governance

Check	Status	Type	Evidence
PII detection in prompts	FAIL	Auto	Gateway not reachable. Runtime check requires gateway
Data governance documentation	FAIL	Human	No DATA_GOVERNANCE.md found
Data vault (controlled storage)	FAIL	Auto	No vault configured
Input validation / schema enforcement	PASS	Auto	384/1,015 Python files use Pydantic validation (38%). Highest percentage of any framework scanned
PII handling in code	PASS	Auto	PII-aware patterns found in 7 files

Analysis: CrewAI's Pydantic adoption at 38% of all files is exceptional for a framework this size. The Pydantic-first architecture means nearly all data flowing through the system is validated against schemas, which is a strong foundation for data governance under Article 10.

Article 11 - Technical Documentation

Check	Status	Type	Evidence
System description (README)	PASS	Human	Comprehensive README.md found
Runtime system inventory (AI-BOM)	FAIL	Auto	No traffic data. Requires gateway for runtime model inventory

Check	Status	Type	Evidence
Model card / system card	WARN	Human	No MODEL_CARD.md found
Code documentation (docstrings)	WARN	Auto	4,316/8,558 public functions have docstrings (50%). At threshold
Type annotations	WARN	Auto	2,440/6,505 public functions have type hints (38%)

Analysis: Docstring coverage at exactly 50% is at the passing threshold. Type annotation coverage at 38% has room for improvement. The codebase is well-structured with clear module separation (security, a2a, events, agents, tasks, flows) which aids comprehension even where docstrings are missing.

Article 12 - Record-Keeping

Check	Status	Type	Evidence
Automatic event logging	FAIL	Auto	Gateway not reachable
Tamper-evident audit chain	FAIL	Auto	No TRUST_SIGNING_KEY set
Log detail and traceability	FAIL	Auto	No logged records via gateway
Application logging	WARN	Auto	Logging found in 100/1,015 files (10%)
Tracing / observability	PASS	Auto	Tracing patterns in 72 files. Event bus with typed events for agents, crews, tools, LLMs, flows, knowledge, MCP, and A2A
Agent action audit trail	PASS	Auto	Action-level audit logging found in 15 files. Fingerprint-based agent identity tracking

Analysis: CrewAI's event bus is a standout. It provides typed events across every layer: agent_events, crew_events, tool_usage_events, llm_events, llm_guardrail_events, flow_events, knowledge_events, mcp_events, a2a_events, and system_events. Combined with the Fingerprint system for agent identity tracking, this gives operators granular audit trails for every agent action.

Article 14 - Human Oversight

Check	Status	Type	Evidence
Human-in-the-loop mechanism	WARN	Auto	No traffic data for runtime analysis
Kill switch / stop mechanism	FAIL	Auto	Gateway not running
Operator documentation	WARN	Manual	No OPERATOR_GUIDE.md found
Human-in-the-loop patterns	PASS	Auto	Human oversight patterns in 31 files. Crew delegation with allow_delegation controls
Usage limits / budget controls	PASS	Auto	Rate limiting/budget controls in 70 files (max_rpm, max_execution_time, max_tokens)
Agent-to-user identity binding	PASS	Auto	User identity binding in 16 files. Fingerprint-based UUID identity system
Token scope / permission validation	PASS	Auto	Scope/permission validation in 7 files. A2A auth with API key and HTTP digest schemes
Token expiry / execution bounding	PASS	Auto	Execution boundary patterns in 32 files (max_iter, timeouts)
Agent action boundaries	PASS	Auto	Action boundary controls in 7 files. allow_delegation flag controls agent-to-agent permissions

Analysis: Article 14 is CrewAI's strongest area (6/9 passing). The crew metaphor naturally maps to human oversight: agents have explicit roles, tasks have expected outputs, and delegation requires permission. The Fingerprint system adds runtime identity tracking, and the A2A protocol adds cross-agent authentication. This is the most compliance-aligned agent architecture we've scanned.

Article 15 - Accuracy, Robustness & Cybersecurity

Check	Status	Type	Evidence
Prompt injection protection	FAIL	Auto	Gateway not running. Runtime injection filtering requires gateway
Error resilience	WARN	Auto	No traffic data to measure resilience
API access control	WARN	Human	No API keys detected in scan environment
Adversarial robustness testing	WARN	Manual	No red team / adversarial testing evidence
Retry / backoff logic	PASS	Auto	Retry/backoff patterns in 60 files
Prompt injection defense	PASS	Auto	Injection defense patterns in 65 files. Dedicated security module

Check	Status	Type	Evidence
Unsafe input handling	WARN	Auto	17 files flagged: lite_agent.py, a2a utils (content_type, task). Core agent paths need review
LLM output validation	PASS	Auto	Output validation in 52 files. output_pydantic enforces structured LLM responses

Analysis: CrewAI's 65 files with prompt injection defenses (6% of codebase) is the highest ratio we've seen. The built-in hallucination guardrail and LLM output guardrails provide defense-in-depth that most frameworks require external tools for. The 17 flagged files for unsafe input are in the lite agent and A2A utility paths, which warrant review.

Notable Compliance Patterns

Security Module with Agent Fingerprinting

CrewAI includes a dedicated security module (`crewai.security`) with a `Fingerprint` class that provides dual identifiers for every agent: a human-readable ID for debugging and a UUID fingerprint for runtime tracking and auditing. The fingerprint includes metadata validation with depth limiting and size caps (10KB max) to prevent DoS attacks. `SecurityConfig` manages authentication credentials, scoping rules, and impersonation/delegation tokens (some marked as `TODO`, indicating active development).

Built-in Guardrails System

CrewAI provides built-in guardrails at the task level, including a hallucination guardrail (`hallucination_guardrail.py`) and LLM output guardrails (`llm_guardrail.py`). These are integrated into the event bus via `llm_guardrail_events`, making them auditable. This is a genuine Article 15 compliance pattern that most frameworks lack entirely.

Agent-to-Agent (A2A) Protocol

The A2A module implements a full agent communication protocol with `AgentCard` discovery, authentication (API key and HTTP digest schemes), TLS verification, and an extension registry. This is highly relevant to Article 14 and Article 15, as it provides authenticated, verifiable inter-agent communication with proper identity management.

Event Bus Architecture

The `crewai.events` module provides typed events across every system layer: agent lifecycle, crew orchestration, tool usage, LLM calls, guardrail triggers, flow control, knowledge retrieval, MCP protocol, A2A communication, and system events. This comprehensive event taxonomy enables fine-grained audit trails for Article 12 compliance.

Questions for CrewAI Maintainers

- 1. Security module maturity** - `SecurityConfig` has `TODO` markers for authentication, scoping rules, and impersonation tokens. What's the roadmap for these features? Are they actively being developed?
- 2. Hallucination guardrail** - The scanner found `hallucination_guardrail.py`. How does this work in practice? Is it pattern-based, LLM-judge-based, or something else?
- 3. A2A authentication** - The A2A module supports API key and HTTP digest auth with TLS. Is this used in production multi-agent deployments, or is it primarily for the CrewAI Enterprise product?
- 4. `allow_delegation` semantics** - Found in 54 files. When an agent delegates a task, does the delegated agent inherit the original agent's permissions, or does it operate under its own scope?

5. Unsafe input in lite_agent and A2A utils - 17 files were flagged for potentially passing raw user input into prompts. The lite_agent.py and a2a/utils/content_type.py paths seem like they could be attack vectors for prompt injection via inter-agent messages. Can you confirm?

How to Reproduce

This scan was performed using AIR Blackbox v1.2.2. Apache 2.0, runs entirely local.

```
pip install air-blackbox
git clone https://github.com/crewAIInc/crewAI.git
air-blackbox comply --scan ./crewAI -v
```