

AIR Blackbox — EU AI Act Compliance Report

Haystack by deepset

Date: March 12, 2026 | **Scanner:** AIR Blackbox v1.2.0 | **Author:** Jason Shotwell

Important Note on Methodology

AIR Blackbox v1.2.0 performs static pattern matching against Python source code. It detects the *presence* of compliance-relevant patterns (logging, validation, error handling, identity tracking), not whether those patterns are correctly implemented for a specific compliance purpose.

Think of it like a building inspector checking whether smoke detectors exist — not whether they're wired correctly. The scanner identifies where patterns are present and where they're absent. A human review is still needed to validate whether detected patterns serve their intended compliance purpose.

This report is a starting point for conversation, not a compliance certification.

Summary

24 Passing · 10 Warnings · 5 Failing · 39 Total Checks

95% automated detection · EU AI Act Articles 9, 10, 11, 12, 14, 15

Haystack scored highest among 6 major AI agent frameworks scanned. It is the only framework where the scanner detected patterns across all 5 OAuth delegation categories. However, some matches require human verification — see the OAuth section for details.

Rank	Framework	Pass	Warn	Fail	Total
#1	Haystack (deepset)	24	10	5	39
#2	Semantic Kernel (Microsoft)	15	4	0	19
#3	GPT Researcher	15	3	0	18
#4	OpenAI Agents SDK	14	5	0	19
#5	Mem0	13	6	0	19
#6	DSPy (Stanford)	12	6	1	19

Note: Haystack's higher total check count (39 vs 19) reflects that its gateway was running during the scan, enabling additional runtime checks. Code-level checks are comparable across all frameworks.

What Haystack Does Well

Strong input validation. 245 out of 552 Python files (44%) use Pydantic models or dataclasses for structured validation. This is well above average across the frameworks scanned.

Good logging coverage. 143 out of 552 files (26%) include structured logging via Python's logging module. This exceeds the 20% threshold and is the highest among frameworks tested.

Robust retry/backoff logic. 41 files contain retry or backoff patterns — essential for production resilience against API failures.

Human-in-the-loop patterns. 47 files contain human oversight patterns. Haystack's dedicated `human_in_the_loop` module with strategies and protocols shows deliberate architectural investment here.

Fallback patterns. 61 files contain fallback or recovery patterns.

What Needs Attention

Missing governance documents. No `RISK_ASSESSMENT.md`, `DATA_GOVERNANCE.md`, `MODEL_CARD.md`, or `OPERATOR_GUIDE.md` found. These are low-effort, high-impact additions required by Articles 9, 10, 11, and 14.

Docstrings at 29%. 1,738 out of 5,902 public functions have docstrings — just 1% below the 30% passing threshold. A focused documentation sprint would clear this check.

Type hints at 25%. 1,273 out of 4,996 public functions have type annotations. Gradual improvement recommended.

LLM call error handling at 23%. 68 out of 302 files with LLM-related calls have try/except blocks. 234 files are missing error handling around LLM calls.

OAuth Delegation Checks — Honest Assessment

The scanner detected patterns in all 5 OAuth delegation categories for Haystack. However, static pattern matching has limitations. Here's what we found and what needs human verification:

1. Agent-to-User Identity Binding — DETECTED (needs verification)

Files matched: 3 files (test_base_serialization.py, strategies.py, _telemetry.py) **What was detected:** `user_id` references in telemetry tracking and human-in-the-loop strategies **Honest assessment:** The telemetry module tracks `user_id` for analytics. The human-in-the-loop module passes `user_id` in approval flows. These are real identity patterns, but they may not constitute full OAuth delegation binding. **Partially valid — the patterns exist but may serve different purposes than delegation tracking.**

2. Token Scope Validation — DETECTED (partially valid)

Files matched: 8 files including agent.py, strategies.py, azure.py **What was detected:** `scope` references in agent configuration, human-in-the-loop protocols, and Azure auth **Honest assessment:** The agent.py scope usage is relevant — it controls what the agent can access. The Azure utility scope is OAuth-related. Some matches in PDF converter tests are incidental. **Mixed — core agent and auth files show real scope awareness, some matches are incidental.**

3. Token Expiry / Revocation — DETECTED (mostly cache TTL)

Files matched: 12 files **What was detected:** `max_age` in agent configuration, `ttl` in various components **Honest assessment:** The `max_age` in agent.py controls agent loop lifetime — this is a form of execution time-bounding. The `ttl` matches in converters and websearch are cache TTL settings, not token expiry. **Partially valid — agent max_age is real time-bounding, but most matches are cache-related, not OAuth token expiry.**

4. Agent Action Audit Trail — DETECTED (test coverage)

Files matched: 2 files (test_tool_invoker.py) **What was detected:** `execution_log` in tool invocation tests **Honest assessment:** This shows that tool execution is logged during testing. It does not confirm action-level audit logging in production flows. **Weak signal — test coverage exists but production audit trails need verification.**

5. Agent Action Boundaries — DETECTED (serialization, not actions)

Files matched: 1 file (core/serialization.py) **What was detected:** `is_allowed` function for serialization class validation **Honest assessment:** This controls which classes can be deserialized — a security boundary, but not an agent action boundary. **False positive for the intended check — this is deserialization safety, not agent tool restrictions.**

Summary of OAuth Checks

Check	Scanner Result	Honest Assessment
Identity binding	✅ PASS	Partially valid — telemetry + HITL <code>user_id</code>
Scope validation	✅ PASS	Mixed — real scope in agent.py, incidental in tests
Token expiry	✅ PASS	Partially valid — agent <code>max_age</code> is real, cache TTL is not
Action audit trail	✅ PASS	Weak signal — test coverage, not production logging

Action boundaries	 PASS	False positive — serialization safety, not action restriction
-------------------	--	---

Bottom line: Haystack shows stronger compliance-adjacent patterns than other frameworks, particularly in its human-in-the-loop module and agent configuration. But the scanner's pattern matching cannot distinguish between a `scope` variable controlling agent permissions and a `scope` variable in a PDF test. Human review is required to validate these findings.

Article-by-Article Results

Article 9 — Risk Management

Check	Status	Evidence
Risk assessment document	✖ FAIL	No RISK_ASSESSMENT.md found
Risk mitigations active	✖ FAIL	0/4 runtime mitigations active
LLM call error handling	⚠ WARN	68/302 files (23%) have try/except
Fallback/recovery patterns	✔ PASS	61 files with fallback patterns

Article 10 — Data Governance

Check	Status	Evidence
PII detection in prompts	✔ PASS	Gateway active, no PII detected
Data governance documentation	✖ FAIL	No DATA_GOVERNANCE.md found
Data vault	✖ FAIL	No vault configured
Input validation	✔ PASS	245/552 files (44%) — Pydantic, dataclass
PII handling in code	✔ PASS	25 files with PII-aware patterns

Article 11 — Technical Documentation

Check	Status	Evidence
README	✔ PASS	Found
Runtime inventory	✔ PASS	Gateway observed
Model card	⚠ WARN	Not found
Docstrings	✖ FAIL	29% coverage (threshold: 30%)
Type hints	⚠ WARN	25% coverage

Article 12 — Record-Keeping

Check	Status	Evidence
Event logging	✔ PASS	Gateway active
Tamper-evident audit chain	⚠ WARN	No signing key set
Log traceability	✔ PASS	Records include run_id, model, timestamp
Application logging	✔ PASS	26% of files — highest among frameworks
Tracing/observability	✔ PASS	28 files with tracing patterns

Action audit trail	✅ PASS	See OAuth section — needs verification
--------------------	--------	--

Article 14 — Human Oversight

Check	Status	Evidence
Human-in-the-loop (gateway)	⚠️ WARN	No approval gates in gateway traffic
Kill switch	⚠️ WARN	Gateway running, no guardrails
Operator documentation	⚠️ WARN	No OPERATOR_GUIDE.md
Human-in-the-loop (code)	✅ PASS	47 files — dedicated HITL module
Usage limits	✅ PASS	38 files with rate/budget controls
Identity binding	✅ PASS	See OAuth section
Scope validation	✅ PASS	See OAuth section
Token expiry	✅ PASS	See OAuth section
Action boundaries	✅ PASS	See OAuth section — false positive

Article 15 — Robustness & Cybersecurity

Check	Status	Evidence
Injection protection	✅ PASS	Gateway scanning
Error resilience	✅ PASS	0% error rate
API access control	⚠️ WARN	No API keys in environment
Adversarial testing	⚠️ WARN	No evidence found
Retry/backoff	✅ PASS	41 files
Injection defense	✅ PASS	17 files
Unsafe input handling	⚠️ WARN	13 files with potential issues
Output validation	✅ PASS	27 files

Quick Wins

1. **Docstrings: 29% to 30%** — ~50 more public functions documented clears this check
2. **RISK_ASSESSMENT.md** — Generate template: `pip install air-blackbox && air-blackbox init`
3. **DATA_GOVERNANCE.md** — Same: `air-blackbox init`
4. **OPERATOR_GUIDE.md** — Document capabilities, limitations, intervention criteria

How This Report Was Generated

```
pip install air-blackbox
git clone https://github.com/deepset-ai/haystack.git
air-blackbox comply --scan ./haystack -v
```

18 code-level checks. 5 OAuth delegation pattern checks. Runs entirely on your machine. Apache 2.0.

Important: This scanner detects pattern presence, not implementation correctness. Results should be validated by a human reviewer familiar with the codebase.

GitHub: github.com/airblackbox/gateway **PyPI:** pypi.org/project/air-blackbox/1.2.0

Report generated by AIR Blackbox v1.2.0 — The flight recorder for AI agents. Contact:
jason.j.shotwell@gmail.com | github.com/airblackbox