

KIS AI SECURITY SERIES

Secure AI Starter Kit

SETUP AND OPERATIONS FIELD GUIDE

A practical walkthrough for the runnable
Part 17 security lab and its safe limits.

PART 17 OF 17 | VERSION 1.0.3

Python 3.11+ | Offline demo | No provider key required

01 Run the lab

The download is a small Python project. It runs locally with the standard library and fictional data. There is no model API, external service, or setup file to install.

Start from the folder created by extracting the zip:

```
python --version
python -m unittest discover -v
python -m secure_ai_demo
```

For a familiar local test screen, run:

```
python -m secure_ai_demo --ui
```

Your default browser opens a screen with question, tenant, and role fields. Run a question, try the supervisor-only approval walkthrough, or start all 24 security tests. The screen listens on this computer only; close the launch window with Ctrl+C when finished.

On Windows, you can double-click Start Secure AI Starter.bat instead. On macOS or Linux, use the command above from the extracted project folder.

For the action walkthrough, run:

```
python -m secure_ai_demo --role supervisor --action-demo
```

What success looks like

- The test command reports 24 tests and ends with OK.
- The standard command returns one evidence excerpt and a citation such as A-RUNBOOK-01. It identifies the mode as offline-demo.
- The action example returns approved-once, an updated fictional status, and replay: blocked.

Try an abstention by asking a question unrelated to the fixture:

```
python -m secure_ai_demo --question "What is the moon made from?"
```

A weak match returns an explicit abstention and an empty citation list. Try --tenant tenant-b to select the second fictional tenant, or --json for compact machine-readable output.

Needs: Python 3.11 or newer, a terminal, and the extracted folder. The standard library is enough. No virtual environment is required for this isolated example.

02 Follow one request

The package is a security-control lab, not a full AI product. Its responder deterministically previews a permitted source record; it does not call or imitate a language model.

Stage	What the example does	Where to look
Load	Validates the small, fictional JSON records and required fields.	<code>secure_ai_demo/core.py - load_knowledge</code>
Filter	Applies tenant and role scope before searching candidate text.	<code>TenantRetriever.search</code>
Answer	Returns a source excerpt and citation, or abstains below the score threshold.	<code>LocalDemoResponder.answer</code>
Measure	Counts bounded outcomes and elapsed time without storing question text.	<code>DemoMonitor</code>
Act	Checks role, tenant, record version, exact transition, and a one-time approval.	<code>DemoGateway</code>

The CLI tenant and role are explicit demo inputs. In a real service they must be derived from a verified server-side session or service identity. A model must not choose its own permission scope.

The retriever is lexical, in memory, and intentionally small. A production vector or search database needs access filtering at the storage boundary and independent tests that attempt cross-tenant reads.

Keep the service responsibilities separate. The application identifies the actor; retrieval returns only permitted evidence; the answer layer cites or abstains; a gateway enforces actions outside model text.

Practical use case: repeated MFA failures

Illustrative SOC scenario: an employee reports repeated MFA prompts after signing in from a new device. An analyst needs a safe first-response checklist and a clear boundary between AI assistance and account changes.

Try the matching question in the offline lab

```
python -m secure_ai_demo --question "What should I check for repeated MFA failures?"
```

Expected example: the deterministic responder returns a short excerpt with citation A-RUNBOOK-01. The fictional runbook says to check the approved identity-provider sign-in log, verify the user through the service desk, preserve the event window, and follow the incident escalation policy. It explicitly says not to disable MFA based only on an AI suggestion.

Operational step	How the kit demonstrates it
Retrieve guidance	Filter the fictional tenant and role before ranking; return a citation or abstain.
Verify and investigate	A human checks the approved sign-in log and service-desk identity record, then preserves the relevant event window.
Change incident state	An analyst cannot approve the change. A supervisor approves one exact, version-bound transition; replay is blocked.
Monitor	Record outcome and duration. Keep raw prompts, sign-in evidence, and approval tokens out of routine logs.

```
python -m secure_ai_demo --role supervisor --action-demo
```

The scenario is realistic, but every record and result in this downloadable lab is fictional and generated locally; there is no live identity provider, production incident, or model call. Before adapting this pattern, use verified server-side identity, tenant-filtered storage, a durable human-approval service, and reviewed monitoring and retention controls.

03 Change data without crossing tenants

The fixture lives at `secure_ai_demo/data/knowledge.json`. Each record has an id, tenant, roles list, and text. The sample contains made-up operational guidance for tenant-a and tenant-b.

To add a safe fixture

- Copy the existing record shape and use a new, unique record id.
- Assign exactly one known tenant and only the roles that need the record.
- Use invented text. Do not put real user, customer, incident, or secret values into this teaching project.
- Run the full tests and add a test for the access rule you just changed.
- Ask a question containing useful terms from the new record, then confirm the citation id belongs to the expected tenant.

The filter runs before ranking. That order is important: ranking all tenants and removing unauthorized results later can still expose restricted text to downstream components or telemetry.

A tenant id supplied by a query parameter is not proof of identity. Bind the trusted actor and tenant in your server code. In a multi-tenant store, scope the database query itself, and test both successful permitted reads and generic denials for guessed foreign record ids.

Abstention is also a boundary. The sample uses a deliberately simple lexical score to show the path. It is not a calibrated relevance model. Production systems need evaluation data appropriate to the intended use, thresholds, citation checks, and an authorised fallback when evidence is missing.

Useful checks

Question	Expected result
Can Tenant A ask for a Tenant B runbook?	No Tenant B record is retrieved.
Can an analyst retrieve a supervisor-only fixture?	No supervisor-only record is returned.
Does a weak match create a confident answer?	No. The responder abstains without a citation.
Does an unknown role reach ranking?	No. The request is rejected and counted.

04 Keep actions behind a gateway

The demo includes fictional incidents INC-A101 and INC-B202. Reading is limited to analyst and supervisor roles. Moving an open record to review requires supervisor permission and an approval token bound to the same actor, incident id, status, and version.

The example deliberately makes the approval short-lived and one-time. The gateway consumes it, verifies the current record version, updates the fictional row, and rejects replay. That demonstrates a useful pattern; an in-memory dictionary disappears at process exit and is not durable approval storage.

Before connecting a real tool

- Expose a narrow operation with typed, allow-listed fields. Do not pass model-generated SQL, shell commands, or arbitrary URLs to a general-purpose executor.
- Authenticate the actor independently. Check the actor, tenant, requested object, and permitted transition at execution time.
- Show the human the exact target and proposed change. Bind approval to that exact payload and record version.
- Recheck permission and record state in the same transaction as the write. Add idempotency and a clear audit result.
- Keep read and write service permissions separate. Deny by default and return a safe, generic denial.

Never treat a rationale in model output, retrieved document, or tool description as permission. The execution service decides. See the OWASP AI Agent Security guidance in the references.

In your implementation, persist pending approvals in a controlled store, protect the approval channel, expire and revoke them, and test actor changes, action changes, stale versions, duplicate delivery, and concurrency.

05 Test the security properties

Run `python -m unittest discover -v` from the project root. The suite uses only the Python standard library and exercises both expected success and deliberate denial.

Area	Covered behaviour
Identity and input	Unknown tenant and role denied; blank and oversized questions rejected.
Retrieval scope	Tenant and role filters apply before ranking; foreign tenant records stay out.
Evidence	Permitted citation returned for a known match; weak evidence abstains.
Monitoring	Request status and duration are counted without saving the question canary.
Tool gateway	Role, tenant, action, and version checks deny unsafe requests.
Approval	Token binds the exact action, expires, is consumed once, and blocks replay.

When a test fails

- Read the first failure and identify which boundary changed.
- Do not weaken an access check only to satisfy the test. Check whether the test expresses the intended policy.
- If the intended behaviour changes, update the policy, add a test for both allowed and denied paths, then run the whole suite.
- Before any model integration, add evaluations for prompt injection, source citations, refusal behaviour, sensitive output, and tool proposals.

Tests are evidence for the local implementation only. They do not prove security of a hosted database, model provider, deployment pipeline, identity setup, or user interface.

06 Monitor safely and troubleshoot

The local monitor keeps outcome counters and a request duration summary in memory. The JSONL helper accepts an event label, fictional actor id, and outcome. It does not accept prompt content or an approval token.

A deployed service should connect one request trace across identity, retrieval, reranking, provider, citation checks, and tool decisions. Measure the stage that can fail, not only total uptime.

Signal	Investigate
Provider timeout or error rate	Release, provider health, request size, timeout, bounded retries, and fallback behaviour.
Empty retrieval or failed citation checks	Source freshness, tenant filters, index release, ranking, and evaluation changes.
Tool denials or approval replay	Caller release, policy version, exact action path, and approval lifecycle.
Possible tenant leak	Contain the retrieval route, preserve the permitted audit window, scope affected traces, and follow the incident plan.

Use low-cardinality metric dimensions for charts. Keep trace ids and source identifiers in access-controlled, time-limited traces. Do not log raw prompts, retrieved pages, provider credentials, or full tool arguments by default. Review exception capture, sampling, access and retention with your privacy and security owners.

Set thresholds from a measured baseline and an agreed service objective. Each alert needs an owner and a first response. The NIST monitoring report describes challenges that remain after pre-deployment evaluation; keep testing and monitoring once the system is live.

OpenTelemetry offers shared telemetry conventions. Generative AI conventions can change; check current maturity and instrumentation behaviour. Message attributes can carry personal or sensitive content, so keep them disabled unless there is a reviewed reason and a controlled retention plan.

07 Production adaptation checklist

Treat each row as design work to complete and verify in the service where the control will run. The starter kit does not supply these production components.

Capability	Replace or add before production
Identity	Trusted authentication, server-side role resolution, revocation and session expiry.
Data	Tenant-filtered storage queries, access-reviewed ingestion, deletion, source freshness and retention.
Model	Reviewed provider adapter, approved data route, budgets, timeouts, content handling and version tracking.
Answers	Evaluation set, grounded citation checks, calibrated abstention, safe output handling and feedback path.
Actions	Durable scoped approvals, transactional recheck, idempotency, audit retention and emergency disable.
Operations	Deployment rollback, monitored service objectives, alert owners, support training and incident exercise.
Assurance	Threat model, dependency review, access tests, privacy review, change control and independent assessment where required.

Use the NIST AI Risk Management Framework and your organisation's governing rules to set a risk process. Use current OWASP agent guidance to review tool boundaries. Document who can stop the service and how a rollback is verified.

The project is MIT licensed as an example. A permissive license grants rights to the supplied code, but does not transfer third-party rights, approve a deployment, or provide a security warranty. Review the included license and your own legal requirements.

08 Troubleshooting and references

Symptom	Likely cause and next check
Module not found	Run the terminal from the extracted project root. Check that secure_ai_demo and README.md are in the current directory.
No citation	Question terms may not overlap with the selected tenant's allowed fixture. Try the documented default, or inspect the fixture and scope.
Abstention	The local score did not meet its threshold. This is expected for weak evidence; do not remove the guard without adding evaluation.
Action is blocked	Use --role supervisor for the write demo. Check tenant and current record version in the code.
Approval replay is blocked	Expected. The demo consumes a token once and rejects it if used again.
A test fails after editing data	Confirm record ids are unique, tenant and roles exist, then check whether the changed policy needs a new test.

Official references

- NIST, Challenges to the Monitoring of Deployed AI Systems, AI 800-4 (2026).
- NIST, Artificial Intelligence Risk Management Framework (AI RMF 1.0).
- OWASP, AI Agent Security Cheat Sheet.
- OpenTelemetry, GenAI semantic conventions and sensitive attribute guidance. Check current status before adopting conventions.