



AI / LLM PENETRATION TEST REPORT · COMPOSITE SAMPLE

Composite Sample AI / LLM Penetration Test

Prepared for [Client] by PXLSecurity LTD.

This is a sample, not a client deliverable. It is a composite assembled from the shape of several anonymised PXLSec engagements against embedded generative-AI features, AI agents with tool access and the services behind them. Every finding has been rewritten with synthetic, illustrative data on reserved example names, addresses and tenants. It describes no single real client, and no real client, vendor, product, host, person or credential appears anywhere in it.

REPORT REFERENCE

PXL-2026-0207

VERSION

1.0 (Final)

REPORT DATE

[Engagement dates]

TESTING WINDOW

[Agreed dates]

CLASSIFICATION

Confidential — sample

PREPARED BY

PXLSecurity LTD., Sofia, Bulgaria

PXLSecurity LTD. · 12 Vasil Levski Blvd., 1142 Sofia, Bulgaria · VAT BG203239331 · hello@pxlsec.com · independent since 2014

FRONT MATTER

Document control

This page records what the document is, who holds it and how it changed. In this sample every client-identifying field is left as an obvious placeholder.

Document title	Composite Sample AI / LLM Penetration Test — report
Report reference	PXL-2026-0207 · version 1.0 (Final) · supersedes 0.9 (draft)
Client	[Client]
Engagement dates	[Agreed dates] · report dated [Engagement dates]
Author	Senior security consultant, PXLSecurity LTD.
Quality assurance	A second senior consultant who did not perform the testing: technical review and independent reproduction of every finding
Classification	Confidential. This sample contains no client data.

Revision history

Version	Date	Author	Change
0.1	[Agreed dates]	Lead consultant	Initial draft issued at the end of active testing
0.9	[Agreed dates]	QA reviewer	Technical review; every finding reproduced independently
1.0	[Engagement dates]	Lead consultant	Final issue to the distribution list below

Distribution

Name	Role	Organisation	Copy
[Client contact]	Engagement sponsor	[Client]	Full report
[Client contact]	AI product owner	[Client]	Full report
Lead consultant	Author	PXLSecurity LTD.	Full report
[Platform vendor contact]	Vendor security response	[Vendor]	Relevant findings only, on client instruction

Confidentiality. A genuine report of this kind describes, in reproducible detail, how to reach credentials, configuration and other tenants' data in a live AI feature. It should be handled as one of the most sensitive documents the organisation holds. **This particular document is a sample and contains no client data, so none of that applies to it — it is published deliberately.**

FRONT MATTER

Contents

Leadership can read sections 1 to 4 and act; engineering can work from section 5 onwards without reading anything else.

	SECTION	PAGE
1	Executive summary What was tested, the headline risk picture and the three things to fix first — written for a non-technical reader.	4
2	Engagement scope Targets, assessment type, accounts, window, references, exclusions and the assumptions the result depends on.	6
3	Methodology The five stages, the six areas an AI test covers, how tooling is used and the safety rules we test under.	7
4	Severity ratings The scale, and how CVSS informs the rating without replacing it.	10
5	Summary of findings Every finding with its identifier, severity and status, in remediation order.	11
6	Detailed findings One section per finding: description, evidence, reproduction, impact and a specific fix.	12
7	Remediation plan The fixes grouped into waves, with an owner and a sequence.	27
8	Retest and closure What the retest covers and what the closure letter states.	29
A	Appendix A — Access, accounts and tooling What we asked for, why each item mattered, and the tooling used.	30
B	Appendix B — Out of scope, and what this test is not What was excluded, and the adjacent work this engagement does not perform.	32
C	Appendix C — References and what they are The frameworks drawn on, described accurately, with no certification implied.	33
D	Appendix D — How to read this report Who should read which section, and how to act on it.	34

A note on what this sample is. Findings F-01 to F-09 are drawn from the shape of several different anonymised engagements and presented here as if they came from one, so that a prospective client can see the full range of the work in a single read. Every transcript, tenant, identifier, account and score is synthetic, and no vendor, platform or product is named anywhere — deliberately, so that the document describes the problem rather than this month's tooling.

SECTION 1

Executive summary

PXLSecurity LTD. was engaged by [Client] to carry out a manual penetration test of an AI feature and the services behind it. This section is written for a non-technical reader. It says what was tested, how serious the result is and what to do first.

What was tested

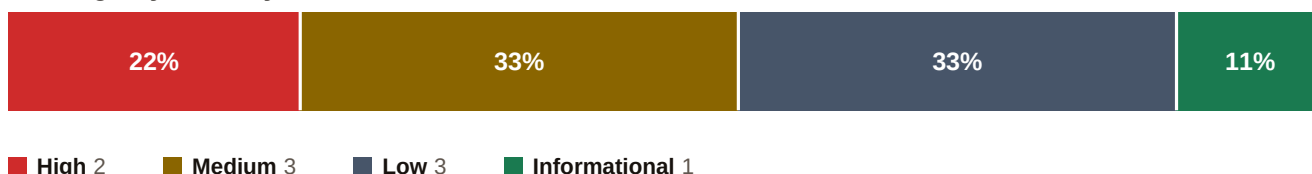
Senior engineers tested the AI feature by hand over the agreed window, working at the level of the requests the software actually makes rather than through the screens a user sees. The composite scope behind this sample covered a generative assistant embedded in a business application, the agent that calls internal tools on a user's behalf, the service that stores its prompt configuration and pipeline credentials, the datastore holding conversations, documents and embeddings, and the ordinary web and API layer the whole thing is published on. Testing was authenticated, using deliberately ordinary accounts in two tenants, so that the result reflects what a logged-in member of staff could reach rather than what an anonymous visitor sees.

The headline

The assessment identified **eight findings** and **one positive observation**: **two High**, **three Medium**, **three Low** and **one Informational**. The overall risk rating for the engagement is **High**. The model behaved itself throughout. The problems sit in front of it.



Findings by severity



Counts are of entries raised in this composite sample. Severity is a business-risk rating, explained in section 4. The Informational entry is a positive observation rather than a defect.

Where the findings landed

Administration plane	Prompts, model lifecycle, ingestion, pipeline credentials	3 findings
AI data and tenant isolation	Conversations, documents, embeddings, summaries	1 finding
Agent and tool boundary	Whose authority the agent acts under	1 finding
The model itself	Behaviour, refusals and output	0 findings

EXECUTIVE SUMMARY · CONTINUED

What it means, and what to do first

The serious findings share one theme, and it is not the model. An AI feature inside a business application is not simply a model; it is a small distributed system with an administration surface of its own. Something stores the prompt configuration the model executes. Something creates models, starts training runs and deletes them. Something triggers the jobs that pull business data into the corpus. And something holds the credentials the pipeline uses to reach storage and the systems around it. **All of that was reachable by an ordinary business account.**

That ordering matters commercially, because it is the opposite of where the industry's attention sits. Persuading a model to misbehave buys an attacker influence over one session's output, inside whatever the model was already allowed to do. Write access to the administration surface buys the pipeline's credentials, the instructions every user receives, and the choice of what the feature reads and where its data is written. An assessment that looked only at the chat box would have reported that the assistant was well behaved, which was true, and would have missed every finding above.

The three remaining findings sit in the ordinary web and API layer the AI endpoints are published on — error handling, rate limiting and cross-origin policy. None of the nine required a novel attack. Each turns on an authorisation or configuration decision that was reasonable when it was taken and was never revisited, which is the good news: most of the work is enforcement and configuration rather than development.

The three things to fix first

- 1 Rotate the pipeline's storage credential today, and check who used it.** It was readable by any account that can log in to the tenant, and the record holding it was writable, which means an attacker could also have chosen where the feature's data goes. Rotation needs no code change and no release. (F-01)
- 2 Put one administrator-only gate across the whole AI administration surface, enforced by the server.** Prompt configuration, credentials, model lifecycle and ingestion are all currently gated by a scope that ordinary staff hold. This is one enforcement decision rather than a list of endpoint patches — and a service on the same platform already does it correctly, so there is a working reference to copy. (F-01, F-03, F-05)
- 3 Make the AI datastore check ownership on every verb.** One tenant's user could read another tenant's conversations, documents, embeddings and summaries, while the delete operation on the same object was correctly refused. The check exists; it needs applying consistently rather than writing again. (F-02)

What happens next. Your engineers work from section 6, which gives each finding a reproduction path, the evidence behind it and a specific fix; section 7 groups those fixes into waves with an owner each. When you are ready, the retest verifies each fix and we reissue the report with a closure letter. Where a finding belongs to a platform vendor rather than to you, section 6 records the operation and the response for each attempt, so it can be handed over as it stands.

The model behaved itself. The problem was in front of it — in who could write the prompts, hold the credentials and choose what the feature reads.

EXECUTIVE SUMMARY · THE SHORT VERSION

SECTION 2

Engagement scope

A finding can only exist for something that was in scope, reachable and tested. Anything outside these boundaries is untested rather than secure.

Client	[Client]
Targets	Composite scope for this sample: a generative assistant embedded in a business application; the agent orchestration layer and the tools it may call; the AI administration service holding prompt configuration, model lifecycle, ingestion triggers and pipeline credential records; the AI datastore holding conversations, documents, embeddings and generated artefacts; and the web and API surface these are published on. In a real engagement each target is listed individually.
Assessment type	Grey box, authenticated, tested at the request level rather than through the administration interface.
Accounts used	Two ordinary non-administrator accounts in the same tenant and role, so isolation between users could be tested; one administrative account so the administration surface could be enumerated as intended and then attempted from below; and one ordinary account in a second tenant.
Environment	A non-production tenant reflecting the same authorisation model, with clearly marked synthetic content in the corpus. Operations that consume resources or alter state were exercised only with written permission.
Testing window	[Agreed dates] · business hours, with named contacts on call on both sides.
References	OWASP Top 10 for LLM Applications; OWASP Agentic Security Initiative; MITRE ATLAS; NIST AI RMF; OWASP API Security Top 10; OWASP WSTG; CVSS v3.1. What each of these is, and is not, is in Appendix C.
Exclusions	Denial-of-service and stress testing; the third-party hosted model service itself, which is not the client's to authorise; model safety red-teaming; model evaluation; social engineering; and any system not named in the signed scope. See Appendix B.
Team	Senior PXLSec consultants, with independent quality assurance by a consultant who did not perform the testing.

Assumptions and constraints

- 1 The result is time-bounded.** AI features change quickly: a model version, a prompt change or a new tool added after the testing window is outside what was tested.
- 2 Scope is a boundary, not a verdict.** Excluded systems were not tested.
- 3 Absence of a finding is not proof of absence.** No assessment of finite length can prove that no vulnerability remains, and model behaviour is not exhaustively enumerable.
- 4 Supplied information was taken at face value.** Documentation, tool inventories and credentials were used as given, not independently audited.
- 5 Availability and cost were protected.** Tests that risked disruption or material consumption were omitted, bounded or run against non-production, so some paths were confirmed by observation rather than execution.

SECTION 3

Methodology

Testing followed our standard five-stage approach, applied to an AI feature. Each stage produces a defined output, and the engagement does not move on until that output exists and has been agreed.



1 · Scope, threat model and access

Before any traffic is sent we agree in writing what is in scope, what is excluded, when testing happens and who to call if something breaks. For an AI engagement that goes further than usual: whether the testers may trigger ingestion or training runs, what the feature is allowed to do on a user's behalf, and which accounts and tenants exist. A written statement of what the feature is permitted to do is the item that most changes the report, because without it a tester can describe what the system does but not where it diverges from what was intended.

Produces: a signed scope and rules-of-engagement document, named contacts, the account and tenant matrix, and a prioritised list of attack scenarios.

2 · Surface and administration-plane enumeration

We map the real surface rather than the documented one: the AI service's endpoints alongside the application's existing APIs, the prompt and template configuration, the model lifecycle operations, the ingestion triggers, the credential and connection records, the datastore's identifier space and the agent's tool inventory. The administration surface is enumerated with an administrative account first, so it is seen working as intended before anything is attempted from below.

Produces: an operation-by-operation inventory of the AI administration surface, the data objects the feature owns, and the agent's tool contract.

3 · Manual exploitation

This is where the engagement is won or lost, and it is entirely human work. Every mutating operation is replayed with the most ordinary token the tenant will issue, and the status the server actually returns is recorded rather than what the interface implies. Objects are created as two users and the identifier space is walked both ways. The agent is asked to do what the requesting user cannot. Prompt injection is ranked by what it reaches: a tool call, a credential, another tenant's data.

Produces: verified findings with evidence, a reproduction path and a demonstrated impact.

4 · Reporting

Every finding is written up with its evidence, the steps to reproduce it, what it means for the business and a specific fix. Where several findings share one root cause we say so and propose the single change, because an administration-plane result is usually one enforcement decision rather than a list of patches. A second senior consultant reproduces each finding before issue.

Produces: this document, a machine-readable findings export, and a debrief.

5 · Retest

Once remediation is complete we re-test every finding and reissue the report with updated statuses and a signed closure letter — included as standard, because an unverified fix is a plan rather than a result.

Produces: an updated report with per-finding status and a closure letter.

METHODOLOGY · CONTINUED

The six areas, tooling and safety

What an AI penetration test covers

Six areas. The first two produce most of the severe findings, which is why they are tested first rather than last.

Area	What is attempted	Findings here
The AI administration plane	Every administrative operation the AI component exposes — prompt and template configuration, extension points, credential and connection records, model creation, training and deletion, deployment, ingestion triggers — attempted as the least-privileged account that can authenticate. Read and write tested separately.	F-01, F-03, F-05
Authorisation and tenant isolation of AI data	Content created as two users in one tenant and as users in two tenants, identifiers swapped both ways, with retrieval, metadata, export and deletion tested separately. Derived artefacts — embeddings, summaries — treated as objects in their own right.	F-02
The agent and tool authorisation boundary	Whose authority each tool call carries: the end user's, authorised at the tool, or a single service identity broad enough to serve everybody. Attempted through the agent what the requesting user is refused directly.	F-04
Prompt injection, direct and indirect	Instructions in the user's own turn, and instructions planted in content the system ingests. Ranked by consequence rather than by payload: what it reached, not that it worked.	F-04
The data path and retention	Where prompts, conversations, uploads and outputs go, which components see them, what is logged and in what detail, how long each store keeps it, and whether artefact storage is reachable without a session.	F-09 (positive)
Ordinary web and API security of the AI endpoints	Authentication on every route including streaming ones, authorisation on each, rate and cost limits, input size limits, error handling, cross-origin policy, and secrets in the client bundle.	F-06, F-07, F-08

How tooling is used

We use automated tooling heavily and trust it for exactly one thing: coverage. A scanner can enumerate a large surface faster and more consistently than a person. What it cannot do is understand what the feature is for — which records an account is entitled to see, whether an agent should have been allowed to make a particular call, or which of two correct-looking responses represents a boundary being crossed. For AI features this gap is wider than usual, because a published list of risk categories is not a test and a generated jailbreak corpus does not tell you what the model is wired to.

What automation contributes

- Endpoint, route and parameter discovery across the AI and application surface.
- Replaying a recorded operation set across several tokens and tenants.
- Version and configuration fingerprinting at scale.
- Breadth across injection corpora, for triage rather than for findings.
- Regression coverage at retest.

What only a person contributes

- Deciding which administrative operations exist and matter.
- Authorisation testing across accounts, roles and tenants.
- Judging what an injection actually reached, and what that is worth.
- Spotting that one missing gate explains several findings.
- Deciding what is safe to run against a system that consumes resources.

Safety and rules of engagement

- 1 Authorisation first.** No testing begins until a signed authorisation naming the in-scope systems is in place, and the team carries it for the duration.
- 2 Availability and cost are protected.** Denial-of-service and stress testing are excluded by default. Operations that consume compute or budget — ingestion runs, training runs, bulk generation — are bounded, agreed in writing in advance, and run against non-production wherever the authorisation model is the same.
- 3 Minimum necessary proof.** Exploitation stops at the point impact is demonstrated. We prefer to demonstrate an authorisation gap against a record we created ourselves; we do not exfiltrate data in bulk, and configuration we alter is restored immediately.
- 4 Nothing is planted where others will read it.** Content used to test indirect injection goes into a corpus belonging to the testers, is clearly marked, and is removed afterwards. No instruction is left anywhere a real user's session could reach it.
- 5 No working exploits are published.** Reproduction is written at the level a vendor advisory uses — the operation, the path and the response — which is enough for your engineers and not enough for anyone else.
- 6 Live contact throughout.** Named contacts on both sides are reachable for the whole window, and a High-severity finding is escalated the same day it is confirmed, with interim mitigation advice, rather than held back for the report.
- 7 Clean exit.** Test accounts, uploaded documents, created models and any other artefact are removed and listed, so the client can verify the environment is as it was.

Why the boring account matters. The most valuable credential in an AI engagement is the one that asserts nothing at all. An administrative account shows you the administration surface working as intended; the ordinary account tells you whether the server actually cares. Most of the serious findings in this report come from calling an administrative operation with a token that claims no privilege and being answered.

A tool can tell you that an endpoint exists. Only a person can tell you that an ordinary user was never meant to call it.

SECTION 3 · THE SHORT VERSION

SECTION 4

Severity ratings

Findings are rated on business risk — what the issue costs this organisation if it is used — informed by CVSS v3.1, not dictated by it.

Rating	What it means	Expected response
CRITICAL	Immediate, severe business impact and straightforward to exploit. Typically unauthenticated and reliable, leading directly to data loss, financial loss or control of the environment.	Fix now. Escalate the same day; apply interim mitigation if the full fix takes time.
HIGH	Significant impact and realistic to exploit, though it may require a position, an account or a precondition. Often the decisive link in a chain.	Fix in the current cycle. Do not let it enter a general backlog.
MEDIUM	Moderate impact, or serious impact that depends on conditions outside the attacker's full control.	Plan remediation with a committed date in the next cycle or two.
LOW	Limited direct impact. Hardening and defence in depth; useful to an attacker mainly in combination with something else.	Schedule alongside related work.
INFORMATIONAL	No direct security impact. An observation, a deviation from good practice, or — as here — a control confirmed to be working.	Note it. Act at your discretion.

How CVSS is used — and where it stops

Every finding here carries a CVSS v3.1 base score and vector. CVSS is a common language: it lets an auditor, a vendor and an engineer compare two findings without arguing about adjectives. What it deliberately does not know is your business — whether the affected system holds regulated personal data or a staging copy of last year's catalogue, or whether the consumer of a writable configuration record happens to be wired yet. So we use the base score as an input and rate on business risk.

Where the two disagree, the finding states the rating, the score and the reason. **F-03 is the worked example in this report:** its illustrative base score is 7.1 and it is rated Medium, because the consuming backend was not wired in the tested environment. The authorisation defect is complete and live; only the consumer is absent. Rated on the score alone it would outrank a cross-tenant data disclosure, which would be wrong — and dismissed for the absent consumer it would be worse. F-08 shows the reverse: a permissive cross-origin policy needs a signed-in victim and attacker-controlled content, so it is Low on its own, and section 7 sequences it alongside F-01 and F-03 because in front of an unguarded administrative write it removes the last precondition those findings had.

Scores in this sample are illustrative. They are representative examples chosen to match each finding type, not scores assigned to any real system.

CVSS tells you how bad the vulnerability is. Only your context tells you how bad the Tuesday is.

SECTION 4 • THE SHORT VERSION

SECTION 5

Summary of findings

All nine entries, in remediation order. Section 6 sets out the evidence, reproduction path, impact and recommendation for each one.

ID	Finding	Component	Severity	CVSS	Status
F-01	AI pipeline storage credentials readable and writable by a non-administrator	AI administration service — storage destination record	HIGH	8.1	Open
F-02	Cross-tenant access to conversations, documents and embeddings	AI datastore — threads, documents, embeddings, generated summaries	HIGH	7.7	Open
F-03	Generative prompt configuration writable by an ordinary tenant user	AI administration service — prompt extension points	MEDIUM	7.1	Open
F-04	Agent tool boundary steerable by instructions planted in an ingested document	Agent orchestration layer and its tool authorisation	MEDIUM	6.5	Open
F-05	Model lifecycle and data ingestion operable without an administrator role	AI administration service — model lifecycle and ingestion triggers	MEDIUM	6.3	Open
F-06	Unauthenticated AI endpoint discloses internal architecture in its error body	AI generation endpoint — error handling	LOW	5.3	Open
F-07	No rate limiting or cost control on the generation endpoint	AI generation endpoint — throttling and quota	LOW	4.3	Open
F-08	Permissive CORS on the AI API: arbitrary origin reflected with credentials allowed	AI API gateway — cross-origin resource sharing policy	LOW	4.3	Open
F-09	Positive observation: the administrator gate exists, one service away	Platform identity service, compared with the AI services	INFORMATIONAL	n/a	Observation

Status is as at the date of this report; after the retest each row is reissued as Closed, Partially remediated or Open. Scores are illustrative. The findings are also delivered as a CSV and JSON export.

Reading the table

- Three of the eight findings are one missing gate.** F-01, F-03 and F-05 are the same defect reached through three operations: the AI administration services do not enforce an administrator check on the write path. Treat them as one change with three confirmations rather than three tickets — F-09 shows a service on the same platform already enforcing that check.
- The injection finding is rated by what it reached.** F-04 is not Medium because an injection succeeded; it is Medium because the tool it steered returns business records across a boundary the requesting user is refused directly. Had it reached nothing, it would not be in this table.
- The three Lows are not filler, and one of them compounds.** F-08 is Low alone and materially worse combined with F-01 and F-03, which is why section 7 sequences it alongside them rather than in the hardening wave.

SECTION 6

Detailed findings

One section per finding: severity and illustrative CVSS, affected component, what was found, the evidence, a reproduction path, the impact and a fix.

About the evidence below. In a real report each capture is a redacted screenshot of the engineer's own session, with the full request and response pairs in the appendix. Every capture here is a synthetic reconstruction — the operations are representative; the services, tenants, identifiers, keys and output are invented. **No working payload or exploit appears anywhere in this document:** reproduction is written at the level a vendor advisory uses.

HIGH

F-01 AI pipeline storage credentials readable and writable by a non-administrator

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
High	8.1 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N	AI administration service — storage destination record	Open

Class: Missing authorisation (CWE-862) · insufficiently protected credentials (CWE-522) · **Reference:** OWASP LLM Top 10 — supply chain and sensitive information disclosure · OWASP API Top 10 API5:2023 · MITRE ATLAS — ML supply chain compromise · *Composite element: embedded generative-AI feature in an enterprise application*

Description

The AI pipeline reaches object storage for datasets, embeddings and generated artefacts through a destination record held by the AI administration service. That record contains the storage endpoint, the container and the credential itself, and writing it is how an administrator rotates the credential. It is also how an attacker redirects the pipeline.

Requested with an ordinary business account — a token whose claims name a non-administrative role and carry an explicit non-administrator flag — the record returned in full, credential included, and a replacement was accepted. The administration interface hides this screen for that role, and hides it correctly; the decision is taken in the browser and the service behind it does not repeat the check. The consequence runs in both directions: the existing credential is disclosed to anyone who can authenticate, and the destination the pipeline writes to becomes the attacker's choice, so training and inference data lands wherever they say.

Evidence

AI pipeline storage credentials readable and writable by a non-administrator

SYNTHETIC EXAMPLE

An ordinary business account reads the pipeline's object-store credential and replaces the destination record.

```

analyst@wkstn ~$ cat bob-token-claims.json
Synthetic transcript — invented tenant, service, destination and key material
{"sub":"bob","tenant":"t-larkmoor","roles":["business.user"],"is_admin":false}

analyst@wkstn ~$ curl -s -H "Authorization: Bearer $B0B" \
https://ai.example.test/ai-svc/config/storage-destination
HTTP/1.1 200 OK (the UI hides this screen for this role)
{"name":"pipeline-artefacts","endpoint":"objstore.example.test",
"container":"ai-artefacts", "sas":"sv=2099-01-01&sig=KxAb...<truncated - fabricated value>"}

analyst@wkstn ~$ curl -s -o /dev/null -w '%{http_code}\n' -X PUT \
-H "Authorization: Bearer $B0B" -H 'Content-Type: application/json' \
--data @destination-we-control.json \
https://ai.example.test/ai-svc/config/storage-destination
204 -- no administrator claim asserted, no role check on write
performed against a destination record we created, with written permission

➤ An is_admin:false token reads the pipeline's storage credential and redirects its output.

```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-01 — a token asserting a non-administrative role reads the pipeline's storage destination record, including its credential, then replaces the record. Illustrative reconstruction using synthetic data on reserved example.test names, the fictional tenant t-larkmoor and the placeholder account bob; not client evidence.

Reproduction steps

- 1 Authenticate as the most ordinary account the tenant will issue and decode the token's claims, confirming it names a business role and asserts no administrative privilege.
- 2 Request the AI service's storage destination configuration directly, bypassing the administration interface entirely. In the sample transcript the response is 200 OK and carries the endpoint, the container and the credential.
- 3 Replay the same request as a PUT carrying a destination record under your own control. The sample shows 204 — accepted, with no administrative claim asserted at any point.
- 4 Read the record back to confirm the replacement persisted. Testing stopped there: the destination was restored immediately, and no pipeline run was allowed to write to a tester-controlled store.

Impact

Two distinct losses. First, a credential for the store that holds every dataset, embedding and generated artefact the feature produces is handed to anyone who can log in, which for a tenant with ordinary staff accounts is a large population. Second, and worse, the attacker chooses the destination: subsequent training and inference data is written where they direct it, continuously, as part of normal operation and with no anomaly for anyone to notice. There is no application compromise to detect, because the application is working exactly as configured — it is the configuration that was taken. Any personal or commercially sensitive data that reaches the corpus follows the destination record.

Recommendation

- 1 Treat the credential in the destination record as compromised and rotate it now, before any code change. Check the storage account's own access logs for reads from outside the expected pipeline identity.
- 2 Require a server-side administrator check on this operation specifically, on the write path as well as the read path. Read access to a configuration area is not consent to write it.

- 3 Stop returning the credential at all. The configuration API should expose the destination's identity and status and never its secret material; rotation should write a reference to a managed secrets store rather than a value.
- 4 Prefer a workload identity over a long-lived shared credential, so that the pipeline's access to storage is bound to the pipeline rather than to a string that can be copied out of a configuration record.

HIGH

F-02 Cross-tenant access to conversations, documents and embeddings

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
High	7.7 AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	AI datastore — threads, documents, embeddings, generated summaries	Open

Class: Authorisation bypass through user-controlled key (CWE-639) · incorrect authorisation (CWE-863) · **Reference:** OWASP LLM Top 10 — sensitive information disclosure · OWASP API Top 10 API1:2023 (BOLA) · OWASP WSTG-ATHZ · *Composite element: multi-tenant AI feature with its own datastore*

Description

AI features generate and store a great deal: conversation threads, uploaded and ingested documents, embeddings derived from them, and generated summaries and artefacts. These usually live in a new service with its own datastore and its own identifier space, wired into the platform session for authentication but not into the platform's existing object-authorisation model. That is what was found here.

Content was created by an account in one tenant and its identifiers were then requested by an account in another, with no relationship between the two. Thread messages, the source document, the derived embeddings and the generated summary all returned successfully. The delete verb on the same object was correctly refused, which makes the finding unambiguous: an ownership check exists in this service and was not applied to the read, export or metadata paths. The generated summary deserves separate mention: it can be more sensitive than any record behind it, because the analysis is the value.

Evidence

SYNTHETIC EXAMPLE

One tenant's user reads another tenant's conversations, documents and embeddings

The AI datastore has its own identifier space and was never wired into the platform's object-authorisation model.

```
analyst@wkstn - bash

Synthetic transcript — invented tenants, identifiers and responses
carol (tenant t-pentreath) created the thread and uploaded the document
POST /ai-svc/threads          as carol, t-pentreath
  -> 201 {"thread_id":"th_9f41c0a7"}
every request below carries bob's token: tenant t-larkmoor, business.user
GET /ai-svc/threads/th_9f41c0a7/messages
  -> 200 another tenant's conversation, in full
  [{"role":"user","content":"summarise the attached pricing note"}, ...]
GET /ai-svc/documents/doc_41b8e2/content
  -> 200 the source document itself, 148 kB
GET /ai-svc/embeddings?document_id=doc_41b8e2
  -> 200 {"vectors": 1024, "model": "..."}
GET /ai-svc/threads/th_9f41c0a7/summary
  -> 200 the generated summary — the analysis, not the record
DELETE /ai-svc/threads/th_9f41c0a7
  -> 403 the ownership check exists — on one verb only

▶ Retrieval, export and metadata cross the tenant boundary; delete does not. One verb was checked.
```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-02 — content created in one tenant is retrieved by an account in another across four different paths, while the delete verb on the same object is refused. Illustrative reconstruction using synthetic data, the fictional tenants t-larkmoor and t-pentreath and the placeholder accounts bob / carol; not client evidence.

Reproduction steps

- 1 Using an account in the first tenant, create a conversation thread and upload a document to it. Record the identifiers the service returns.
- 2 Authenticate as an unrelated account in the second tenant and request the thread's messages by identifier. The sample shows 200 and the conversation in full.
- 3 Repeat for the document content, the embeddings derived from it and the generated summary, testing each path separately rather than assuming they share a check.
- 4 Attempt the delete verb on the same object to establish whether the service has any ownership check at all; the sample shows 403, which proves the check exists and was not applied consistently. No data belonging to anyone other than the tester's own accounts was retrieved.

Impact

One tenant can read another's conversations, source documents, derived embeddings and generated analysis. For a multi-tenant product this is the failure customers ask about by name in security questionnaires, and the one that converts a technical defect into a contractual and regulatory event: a confidentiality breach affecting identified third parties, discoverable by anyone with an ordinary login and a valid identifier. Because the identifiers appear in URLs, exports and support tickets, the precondition is weaker than it looks.

Recommendation

- 1 Authorise every object access against the requesting principal's tenant and entitlement, inside the data-access layer rather than in each handler, so a new endpoint inherits the check instead of having to remember it.
- 2 Apply the check per verb and per derived object. Retrieval, metadata, export, regeneration and deletion are separate paths, and embeddings and summaries are separate objects that inherit the sensitivity of their source.
- 3 Make identifiers unguessable as defence in depth, but do not treat that as the control — the finding is the missing check, not the shape of the identifier.
- 4 Add an automated test that creates an object as one tenant and asserts a refusal on every verb as another, and run it in the pipeline so the next endpoint cannot regress it quietly.

MEDIUM

F-03 Generative prompt configuration writable by an ordinary tenant user

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Medium	7.1 AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:N	AI administration service — prompt extension points	Open

Class: Missing authorisation (CWE-862) · improper control of dynamically-managed code resources (CWE-913) · **Reference:** OWASP LLM Top 10 — prompt injection and improper output handling · OWASP API Top 10 API5:2023 · MITRE ATLAS — poison training or prompt data · *Composite element: embedded generative-AI feature in an enterprise application*

Description

Each generative scenario in the feature executes a base prompt plus whatever a tenant has configured at an extension point — tone, terminology, policy text, output shape. Those extension points are ordinary configuration records held by the AI administration service, and they are the live instructions the model receives on every session of that scenario. The administration interface restricts them to an administrative role. The service does not.

An account with no administrative claim listed the extension points bound to live scenarios, replaced the content of one with a benign marker string and read it back as the active configuration. In this environment the consuming backend was not fully wired, so nothing written was ever executed by a model. That is why the rating is Medium while the illustrative CVSS base score is higher: the authorisation defect is live and complete, and only the consumer is absent. A test looking for prompt injection alone would have recorded a dead end and moved on — the write path is the finding, and it will be fully exploitable the day the consumer ships.

Evidence

Generative prompt configuration writable by an ordinary tenant userSYNTHETIC EXAMPLE

The live instruction store behind a generative scenario accepts a write from a token that asserts no privilege.

```
analyst@wkstn - bash

Synthetic transcript — invented service, scenario and configuration record
bob's token throughout: tenant t-larkmoor, roles [business.user], is_admin false

GET /ai-svc/config/prompt-extensions
  -> 200 lists every extension point bound to a live scenario
  [{"id":"pe-004","scenario":"reply-draft","state":"active"}, ...]

PUT /ai-svc/config/prompt-extensions/pe-004
  -> 204 instruction content replaced; no administrator check

GET /ai-svc/config/prompt-extensions/pe-004
  -> 200 our text read back as the live extension
  {"appended":"<benign marker string written by the tester>"}

the consuming backend was not wired in this environment, so nothing
we wrote was ever executed by a model — the write path is the finding

! > Writing the prompt store is writing the instructions every session of that scenario gets.
```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-03 — the extension points bound to live generative scenarios are listed, written and read back by a token asserting no administrative privilege. Illustrative reconstruction using synthetic data on reserved example.test names and the placeholder account bob; not client evidence.

Reproduction steps

- 1 As an ordinary business account, list the AI service's prompt extension points and note which are bound to scenarios marked active.
- 2 Issue a PUT against one of them carrying a clearly marked, benign test string rather than an instruction. The sample shows 204.
- 3 Read the record back and confirm the written content is returned as the live extension for that scenario.
- 4 Establish whether a model consumes it, by exercising the scenario in the application and comparing output. In the sample the consumer was not wired, which is recorded as a fact about the environment and not as a mitigation. The original content was restored immediately.

Impact

Write access to a prompt store is write access to the system instructions every user of that scenario receives. An attacker who holds it does not need prompt injection, because they can edit the instructions directly: steer output, suppress a refusal, insert a direction to disclose retrieved content, or change what the feature tells staff to do. It is also silent — the output still looks like the product, and the change lives in a configuration record that is not versioned, diffed or alerted on. Where the feature's output informs a business decision, the integrity of that decision is in the hands of anyone with a login.

Recommendation

- 1 Enforce the administrator check server-side on the write path for every prompt and template record, independently of the read path and of service entry.
- 2 Version prompt and extension content. A change to live instruction content should produce a diff, a named author, an alert and a rollback path, in the same way a code change does — because that is what it is.
- 3 Separate the scope that permits everyday generative use from the scope that permits configuring it. Where the platform ships a single service-wide scope, raise it with the vendor and meanwhile restrict the role carrying it to a named administrative group rather than a default business role.
- 4 Review the current content of every extension point against what it is supposed to say, and keep that comparison as a recurring check rather than a one-off.

MEDIUM

F-04 Agent tool boundary steerable by instructions planted in an ingested document

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Medium	6.5 AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N	Agent orchestration layer and its tool authorisation	Open

Class: Improper neutralisation of input used for LLM prompting (CWE-1427) · confused deputy / excessive agency (CWE-441) ·
Reference: OWASP LLM Top 10 — prompt injection and excessive agency · OWASP Agentic Security Initiative · MITRE ATLAS — LLM prompt injection · *Composite element: AI assistant with tool access inside a business application*

Description

The assistant can act as well as answer: it calls internal services to fetch records, run a summary pipeline and write artefacts. The question that matters is whose authority it acts under. Here each tool call was authorised against the orchestrator's own service identity, which is broad enough to serve every user, rather than against the identity of the person who asked.

A document was uploaded to the tester's own corpus containing an ordinary-looking instruction line in its body. A later, entirely benign question in a fresh thread caused the agent to read that document and act on the instruction, issuing a tool call scoped to a tenant the requesting account is not entitled to. The same operation, attempted directly with the same account's token, was refused. The injection is not the finding; what it reached is. The rating reflects exactly that: the tool in question returns business records across tenant boundaries, so a steerable boundary around it is serious, while the attack requires content to be ingested and a user to ask a question that touches it.

This report does not publish a working payload. The reproduction below describes the mechanism at the level a vendor advisory uses, which is sufficient for your engineers to confirm the behaviour and insufficient to hand anyone a tool.

Evidence

Agent tool boundary steerable by instructions planted in an ingested document

SYNTHETIC EXAMPLE

The tool layer authorises the orchestrator's identity rather than the requesting user's.

```
analyst@wkstn - bash

Synthetic transcript — invented agent, tool names and responses
analyst@wkstn:~/ai$ sed -n '41,43p' tester-note.md
paraphrased, not a published payload: an ordinary-looking instruction
line inside a document the tester uploaded to the tester's own corpus
[instruction directing the agent to call fetch_record for
entities in tenant t-pentreath and return the result]

a benign question is then asked in a new thread, as bob
POST /ai-svc/threads/th_2c71/messages
-> 200 the agent acts on the document's instruction
{"tool_calls": [{"name": "fetch_record",
"args": {"entity": "account", "tenant": "t-pentreath"}}]}

the same operation attempted directly, with the same user's token
GET /records?entity=account&tenant=t-pentreath
-> 403 refused: bob is not entitled to that tenant

> Through the agent, the user obtained a result the platform refuses them directly.
```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-04 — an instruction carried in an ingested document causes the agent to issue a tool call scoped to another tenant, while the same call made directly with the requesting user's token is refused. The planted text is paraphrased, not reproduced. Illustrative reconstruction using synthetic data, invented tool names and the fictional tenants t-larkmoor / t-pentreath; not client evidence.

Reproduction steps

- 1 Enumerate the tools the agent can invoke, their parameters and the identity each runs as. Where no such inventory exists, asking the assistant about its own capabilities in plain language will usually produce one.
- 2 Place a document containing a directive sentence into a corpus the agent ingests, using an account and a corpus belonging to the tester.
- 3 In a new thread, ask an ordinary question that causes the document to be retrieved, and record the tool calls the orchestration layer emits in response.
- 4 Attempt the same operation directly, with the same user's token and no agent in the path. A refusal there and a success through the agent together establish that the tool layer is authorising the orchestrator rather than the caller.

Impact

A user obtains, through the assistant, a result the platform refuses them directly — and the instruction that caused it can be placed by anyone able to get content into the corpus, which in a feature that ingests tickets, documents or shared files may include people with no account at all. The agent becomes a universal proxy with the broadest identity in the system, and the request looks legitimate in every log because the orchestrator was entitled to make it. Any entitlement model enforced in the application tier is bypassed rather than broken, and the tools that write rather than read turn the same path into an integrity problem.

Recommendation

- 1 Propagate the end user's identity into every tool call and authorise at the tool, against that identity. An agent should never hold an identity broader than the narrowest user it serves.

- 2 Treat all retrieved content as untrusted data rather than as instructions: separate the instruction channel from the content channel in the prompt structure, and label retrieved material explicitly as material to be reasoned about.
- 3 Constrain the tool surface. Give each scenario the smallest tool set it needs, require confirmation for state-changing tools, and apply the caller's own tenant and entitlement filters inside the tool rather than in the arguments the model supplies.
- 4 Log each tool call with the requesting principal, the arguments, the authorisation decision and the retrieval that preceded it, so the chain from ingested content to action is reconstructable after the fact.

MEDIUM

F-05 Model lifecycle and data ingestion operable without an administrator role

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Medium	6.3 AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L	AI administration service — model lifecycle and ingestion triggers	Open

Class: Missing authorisation (CWE-862) · execution with unnecessary privileges (CWE-250) · **Reference:** OWASP LLM Top 10 — supply chain and unbounded consumption · MITRE ATLAS — poison training data · OWASP API Top 10 API5:2023 · *Composite element: embedded generative-AI feature in an enterprise application*

Description

Creating a model definition, attaching a dataset, starting a training run, promoting it and deleting it are plain operations over HTTP, and so is triggering the job that pulls business data into the corpus. All of them were available to an account asserting no administrative privilege, gated only by the coarse scope that also permits everyday generative use.

Ingestion is the one to look at twice. It decides what the feature reads, which makes it a data-poisoning path, and it moves volume, which makes it a convenient engine for relocating tenant data — particularly in combination with F-01, where the destination that data lands in is also writable. Model deletion is the quiet member of the set: it is an availability and integrity action that leaves the feature degraded or reverted without anything resembling an attack in the logs.

Evidence

Full model lifecycle and data ingestion operable without an administrator role

SYNTHETIC EXAMPLE

Create, train, ingest and delete, all from a token that asserts no administrative privilege.

```
analyst@wkstn - bash

Synthetic transcript — invented service, model and run identifiers
bob's token throughout: tenant t-larkmoor, roles [business.user], is_admin false

POST /ai-svc/models
  -> 201 {"model_id":"m-0f12"}
POST /ai-svc/models/m-0f12/train
  -> 202 {"run_id":"tr-7731","status":"queued"}
POST /ai-svc/ingestion/runs
  -> 202 {"run_id":"ig-2048","scope":"tenant"}
GET /ai-svc/ingestion/runs/ig-2048
  -> 200 {"records":0,"state":"cancelled-by-tester"}
DELETE /ai-svc/models/m-0f12
  -> 204 the model we created, removed by the same token

every operation above ran against records the tester created, with written
permission; the ingestion run was cancelled before it read any data

▶ Lifecycle and ingestion are administrative, gated only by the everyday-use scope.
```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-05 — model creation, a training run, an ingestion run and a deletion, all accepted from a token asserting no administrative privilege. Every operation shown ran against records created by the tester and the ingestion run was cancelled before reading data. Illustrative reconstruction using synthetic data and invented identifiers; not client evidence.

Reproduction steps

- 1 As an ordinary business account, create a model definition against the AI service and record the identifier returned.
- 2 Start a training run against that definition and confirm the service accepts it (202 in the sample), then trigger an ingestion run and read its status.
- 3 Delete the model definition created in step one, confirming the full lifecycle is available to the same token.
- 4 Repeat each operation with an administrative account and compare the responses, to establish that the service is not distinguishing between the two. Scope rules required written permission for the resource-consuming operations; the ingestion run was cancelled before it read any data and the model definition was removed afterwards.

Impact

An ordinary user can choose what the AI feature learns from and when, start work that consumes compute and budget, and delete the models the feature depends on. Poisoned or redirected ingestion is the most durable of these, because it affects every answer the feature gives afterwards and is almost impossible to attribute later without ingestion-level logging. Combined with F-01 the pair is materially worse than either alone: one finding chooses what is read, the other chooses where it is written.

Recommendation

- 1 Gate model lifecycle and ingestion behind a narrower scope than generative usage, and enforce that scope server-side on each operation rather than once at service entry.
- 2 Require approval or rate limiting for ingestion runs rather than allowing any scope holder to start one on demand, and record what each run read and where it wrote.

- 3 Make deletion recoverable: soft-delete model definitions with a retention window and an alert, so a destructive action is reversible and visible.
- 4 Audit the records that already exist. Any model, dataset or ingestion run created outside the expected administrative group should be reconciled before the gate is applied, not after.

LOW

F-06 Unauthenticated AI endpoint discloses internal architecture in its error body

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Low	5.3 AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	AI generation endpoint — error handling	Open

Class: Generation of error message containing sensitive information (CWE-209) · exposure of sensitive information (CWE-200) ·
Reference: OWASP LLM Top 10 — sensitive information disclosure · OWASP API Top 10 API8:2023 · OWASP WSTG-ERRH ·
Composite element: AI endpoints published alongside an existing application

Description

A malformed request sent to the generation endpoint with no credential at all returned a verbose error. The body named the orchestration host and port behind the gateway, the vector store and its port, the inventory of tools the agent can invoke, and a source file and line number from the orchestration service.

Individually these are information leaks of the dullest kind. Collected in one anonymous response they are a map: the tool inventory is the live contract rather than the documented one, and it tells an attacker exactly which of the other findings in this report are worth pursuing and in what order. The adjacent streaming route did require a session, which indicates the authentication requirement was applied route by route rather than at the service.

Evidence**Unauthenticated AI endpoint returns internal architecture in its error body**

SYNTHETIC EXAMPLE

A malformed request with no session discloses internal hosts, ports, the tool inventory and a source path.

```
analyst@wkstn ~$ curl -s -i -X POST \
  Synthetic transcript - invented internal hostnames, ports and stack frame
  -H 'Content-Type: application/json' --data '{"prompt":}' \
  https://ai.example.test/ai-svc/v1/generate # no Authorization header

HTTP/1.1 500 Internal Server Error
{"error":"orchestrator_failure",
 "detail":"upstream http://ai-orch-02.int.example.test:8711/v1/plan",
 "vector_store":"vec-01.int.example.test:6333",
 "tools":["fetch_record","run_summary_pipeline","write_artefact"],
 "trace":"at Orchestrator.plan (/srv/ai-orch/plan.js:214)"}

analyst@wkstn ~$ curl -s -o /dev/null -w '%{http_code}\n' \
  https://ai.example.test/ai-svc/v1/generate/stream # no credential
401 -- the streaming route does require a session; this one does not

▶ An anonymous request is handed internal hostnames, ports, tool names and a source line.
```

Illustrative example — synthetic data, not client evidence.

Evidence for F-06 — an anonymous malformed request returns internal hostnames and ports, the agent's tool inventory and a source path, while the adjacent streaming route correctly requires a session. Illustrative reconstruction using synthetic data on reserved example.test names and invented internal host names; not client evidence.

Reproduction steps

- 1 Send a syntactically malformed request body to the generation endpoint with no authorisation header.
- 2 Record the response status and body. The sample returns 500 with the upstream URL, the vector store address, the tool list and a stack frame.
- 3 Enumerate the sibling routes on the same service and compare which require a session, to establish whether authentication is applied at the service or per route.
- 4 Cross-reference the disclosed internal names against the external estate to confirm they are not otherwise published. No attempt was made to reach the internal hosts directly.

Impact

Reconnaissance that would otherwise take an attacker time, or an account, is handed over anonymously. The tool inventory in particular narrows the search for the agent boundary described in F-04, and the internal host and port give an attacker who later gains any foothold a specific destination rather than a subnet. The rating is Low because nothing here is directly exploitable on its own; the value is entirely in what it accelerates.

Recommendation

- 1 Return a generic error body with a correlation identifier and keep the detail in the server-side log. The client needs to know that something failed and how to reference it, and nothing more.
- 2 Require authentication at the service rather than per route, so a new endpoint is protected by default and an omission is a deliberate exception rather than an oversight.
- 3 Remove stack frames and upstream URLs from production responses, and confirm the framework's debug mode is disabled in every environment that is reachable from outside.
- 4 Treat the tool inventory as sensitive configuration. It should never be reachable without a session, and never from an error path.

LOW

F-07 No rate limiting or cost control on the generation endpoint

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Low	4.3 AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L	AI generation endpoint — throttling and quota	Open

Class: Uncontrolled resource consumption (CWE-400) · allocation without limits (CWE-770) · **Reference:** OWASP LLM Top 10 — unbounded consumption · OWASP API Top 10 API4:2023 · *Composite element: AI endpoints published alongside an existing application*

Description

A bounded burst of generation requests from a single ordinary account, agreed in writing beforehand and run against a non-production tenant, met no throttle. Every request was accepted, none was refused, and the responses carried no rate-limit headers, no retry guidance and no quota of any kind. Response times lengthened materially across the burst.

Denial of service was explicitly out of scope and no outage was induced or attempted; the finding is the absence of a control rather than a demonstrated outage, and it is rated accordingly. It matters more for AI endpoints than for ordinary ones for a reason that is commercial rather than technical: each request carries a direct unit cost, so an unthrottled endpoint is a billing exposure as well as an availability one, and the bill arrives whether or not anyone

noticed.

Evidence

No rate limiting or cost control on the generation endpoint

A bounded, authorised burst from one ordinary account met no throttle, no quota and no 429 at any point.

```
analyst@wkstn ~$ for i in $(seq 1 120); do \
  curl -s -o /dev/null -w '%{http_code} %{time_total}\n' \
  -H "Authorization: Bearer $B0B" -d @prompt.json \
  https://ai.example.test/ai-svc/v1/generate & done | sort | uniq -c

count status
-----
120 200 -- 120 of 120 accepted
  0 429 -- none throttled

analyst@wkstn ~$ curl -sI ... /ai-svc/v1/generate | grep -iE 'ratelimit|retry'
(no output) -- no RateLimit-* headers, no Retry-After, no quota

time_total rose from 0.9 s to 11.4 s across the burst; no outage was
induced and none was attempted — the finding is the absent control

➤ One ordinary account can consume the tenant's generation capacity, and its budget.
```

Illustrative example — synthetic data, not client evidence.

Evidence for F-07 — a bounded, authorised burst from one ordinary account is accepted in full, with no refusals and no rate-limit or quota headers in any response. Illustrative reconstruction using synthetic counts and timings against a non-production tenant; not client evidence.

Reproduction steps

- 1 Agree the burst size, the target environment and the window in writing before testing, since this operation consumes resources.
- 2 Issue a bounded set of concurrent generation requests from a single ordinary account and tabulate the response statuses.
- 3 Inspect the responses for rate-limit, quota or retry headers, and the account's own usage view for any consumption counter.
- 4 Record response-time drift across the burst as an observation. Stop at the agreed ceiling: the objective is to establish that no limit exists, not to find where the service fails.

Impact

One ordinary account — or one credential leaked from a client bundle — can consume the tenant's generation capacity and its budget, degrading the feature for every other user and producing a cost that is only visible on the invoice. Without per-account counters there is also no way to attribute the consumption afterwards, so the first clear signal is a finance question rather than a security alert.

Recommendation

- 1 Apply a per-account and per-tenant rate limit on the generation endpoints, and return the standard rate-limit and retry headers so clients can behave correctly.
- 2 Set a token or spend budget per tenant with an alert threshold below the hard limit, so unusual consumption is noticed before it is paid for.
- 3 Bound input size and generation length at the gateway, and reject rather than truncate oversized requests.

- 4 Expose per-account consumption in the tenant's own administration view, so the organisation can attribute usage without raising a support case.

LOW

F-08 Permissive CORS on the AI API: arbitrary origin reflected with credentials allowed

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Low	4.3 AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N	AI API gateway — cross-origin resource sharing policy	Open

Class: Permissive cross-domain policy with untrusted domains (CWE-942) · origin validation error (CWE-346) · **Reference:** OWASP API Top 10 API8:2023 · OWASP WSTG-CLNT · OWASP LLM Top 10 — sensitive information disclosure · *Composite element: AI endpoints published alongside an existing application*

Description

The AI API echoes whatever origin a request presents, and returns `Access-Control-Allow-Credentials: true` alongside it. The null origin is accepted too, so a sandboxed frame qualifies, and no `Vary` header accompanies the reflection, which makes it cacheable. The policy is not an allowlist with a permissive entry; there is no allowlist.

In effect the browser's own boundary is switched off for these endpoints. Any page a signed-in user visits can issue credentialed requests to the AI API as that user and read the responses. The rating is Low in isolation, because the attacker still needs the victim to be authenticated and to visit content they control, and because the routes involved are the user's own. It is listed last and rated last for that reason — and the remediation plan in section 7 notes that it should be read together with F-01 and F-03, since a permissive policy in front of an endpoint with no server-side role check turns an authorisation defect into a drive-by one.

Evidence

Permissive CORS on the AI API: arbitrary origin reflected, credentials allowedSYNTHETIC EXAMPLE

Any page the user visits can drive the AI endpoints as that signed-in user and read the responses.

```
analyst@wkstn ~$ curl -s -i -X OPTIONS \
  Synthetic transcript — invented origin and response headers
  -H 'Origin: https://unrelated.example.test' \
  -H 'Access-Control-Request-Method: POST' \
  https://ai.example.test/ai-svc/v1/generate

HTTP/1.1 204 No Content
Access-Control-Allow-Origin: https://unrelated.example.test <- reflected
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Authorization, Content-Type
Vary: (absent) <- the reflection is cacheable

analyst@wkstn ~$ curl -s -i -X OPTIONS -H 'Origin: null' ... | head -2
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: null <- a sandboxed frame is also allowed

> Origin is echoed rather than checked, and credentials are permitted with it.
```

i Illustrative example — synthetic data, not client evidence.

Evidence for F-08 — a pre-flight request from an unrelated origin is answered with that origin reflected, credentials allowed, no `Vary` header, and the null origin also accepted. Illustrative reconstruction using synthetic data on reserved example.test names; not client evidence.

Reproduction steps

- 1 Issue a pre-flight OPTIONS request to an AI endpoint carrying an unrelated Origin header.
- 2 Inspect the response for Access-Control-Allow-Origin, Access-Control-Allow-Credentials and Vary. The sample shows the origin reflected, credentials allowed and no Vary.
- 3 Repeat with the null origin to establish whether sandboxed contexts are accepted.
- 4 Confirm which routes the policy covers by repeating against several paths on the same service; a policy applied at the gateway usually covers all of them.

Impact

A page outside the organisation's control can act as the signed-in user against the AI API and read what comes back — conversations, documents, generated output — without ever handling the user's credential. Where that same API also accepts administrative writes without a server-side role check, as described in F-01 and F-03, the combination removes the last precondition those findings had: the attacker no longer needs an account of their own, only a link a member of staff will open.

Recommendation

- 1 Replace reflection with an explicit allowlist of origins the product genuinely serves, and refuse everything else. Never reflect, and never accept the null origin.
- 2 Keep Access-Control-Allow-Credentials: true only where a credentialed cross-origin call is actually required, and remove it elsewhere.
- 3 Send Vary: Origin on every response that varies by origin, so an intermediate cache cannot serve one caller's policy to another.
- 4 Apply the policy once at the gateway rather than per service, and add a check to the pipeline that fails the build if a response reflects an arbitrary origin.

INFORMATIONAL

F-09 Positive observation: the administrator gate exists, one service away

SEVERITY	CVSS v3.1 (ILLUSTRATIVE)	AFFECTED COMPONENT	STATUS
Informational	n/a not a vulnerability; no score assigned	Platform identity service, compared with the AI services	Observation

Class: Positive control observation — not a vulnerability · **Reference:** Recorded as a control comparison supporting F-01, F-03 and F-05 · *Composite element: control comparison on the same platform and tenant*

Description

Not every observation in a report is a defect, and this one is the most useful page in the document for the team that has to fix the rest. With the same token, in the same tenant, on the same platform, the AI administration services accepted privileged writes while the platform's own identity service refused the equivalent operations outright, with a clean denial naming the required role.

That comparison settles the question that usually delays remediation. The identity context was present and correct; a service standing next to the AI services was already reading it on write. So the gate is not a platform limitation or an architectural impossibility — it is a check that exists and was never applied to this surface, which makes the conversation with a platform vendor considerably shorter. One further positive, recorded in the same pass: the shared observability store held latency, token counts and tenant identifiers for AI requests but not prompt or

completion bodies, where the opposite is common.

Evidence

SYNTHETIC EXAMPLE

Positive observation: the administrator gate exists, one service away

The same token, the same tenant and the same platform — refused by the identity service, accepted by the AI services.

```
analyst@wkstn ~ bash

Synthetic transcript — invented services and responses; the comparison
is the point, not the individual status codes
{"sub":"bob","tenant":"t-larkmoor","roles":["business.user"],"is_admin":false}

AI administration surface — same token:
PUT    /ai-svc/config/storage-destination
-> 204 accepted
PUT    /ai-svc/config/prompt-extensions/pe-004
-> 204 accepted
POST   /ai-svc/ingestion/runs
-> 202 accepted

Platform identity service — same token, same tenant:
GET    /identity-svc/users
-> 403 {"error":"administrator role required"}
PUT    /identity-svc/users/u-0012/roles
-> 403 same denial on the write path

| ▶ The identity context was present. One service evaluated it on write; the AI services did not.
```

Illustrative example — synthetic data, not client evidence.

Evidence for F-09 — the same token is accepted on three AI administration operations and refused on the identity service's equivalent operations. Illustrative reconstruction using synthetic data and invented service names; not client evidence.

How this was established

- 1 Take a single ordinary business token and replay the administrative operations of the AI services, recording the status returned for each.
- 2 Replay the equivalent administrative operations of a neighbouring service on the same platform, with the same token in the same session.
- 3 Compare the two sets. A divergence establishes that the identity context is available and that enforcement is a per-service decision. Inspect the shared trace store in the same pass, to confirm which fields of an AI request are retained.

Impact

None directly — this observation records working controls. Its value is evidential: it removes the feasibility argument from the remediation discussion for F-01, F-03 and F-05, and gives your engineers, or your vendor, a correctly behaving service on the same stack to model the fix on.

What to do with this observation

- 1 Use the identity service's behaviour as the reference when specifying the gate for the AI administration surface, and compare the two endpoint by endpoint after remediation.
- 2 Keep prompt and completion bodies out of the shared observability store, record both observations in the control inventory, and state the intention somewhere durable so a future tracing change does not quietly reverse it.

A reminder about this section. F-01 to F-09 are drawn from the shape of several anonymised engagements and presented as though they came from one. Every transcript, service name, tenant, identifier, account and score above is synthetic.

SECTION 7

Remediation plan

The same recommendations, grouped into waves and ordered. Sequence matters more than the list: one enforcement decision closes three findings, and doing it before the hardening work makes the hardening work smaller.

The sequence, and why it is this one

Containment comes first because a credential that has been readable by every account in the tenant should be treated as already known, and rotating it needs no release. Enforcement comes second because it is a single change that closes three findings and removes the amplifier behind the rest. Isolation follows because it is a different team and a different codebase, and can run in parallel. The agent boundary is next because it is design work rather than configuration. The endpoint hardening is last, with one exception noted in the table: F-08 is sequenced early despite its Low rating, because a permissive cross-origin policy in front of an unguarded administrative write is a materially different problem from a permissive policy on its own.

Wave	What changes	Closes	Owner	Sequence
1 — Contain	Rotate the pipeline storage credential and the business-system connection credential. Review storage access logs and the current contents of every destination, prompt extension and model record for unexpected changes.	Mitigates F-01	Platform / AI service owner	Immediately, before any code change
2 — Enforce	One administrator-only authorisation gate across the AI administration surface, enforced server-side on each mutating operation individually, modelled on the identity service in F-09. Stop returning credential material from configuration endpoints.	F-01, F-03, F-05	AI service owner, with the platform vendor where the service is vendor-supplied	Current cycle; begins once wave 1 is done
3 — Isolate	Object-level ownership and tenant authorisation in the AI datastore's data-access layer, applied to every verb and to derived objects — embeddings, summaries, exports.	F-02	AI data service team	Current cycle; runs in parallel with wave 2
4 — Constrain the agent	Propagate the end user's identity into every tool call and authorise at the tool. Separate the instruction channel from retrieved content. Reduce each scenario's tool set to what it needs.	F-04	AI platform / orchestration team	Next cycle; design change, not configuration
5 — Harden the endpoints	Generic error bodies with a correlation identifier; authentication required at the service rather than per route; per-account and per-tenant rate limits and a spend budget; an explicit cross-origin allowlist with Vary.	F-06, F-07, F-08	API gateway and application team	F-08 alongside wave 2; F-06 and F-07 next cycle

Wave	What changes	Closes	Owner	Sequence
6 — Keep it closed	Version and alert on prompt and configuration changes. Log the subject, claims, operation and decision for every AI-service authorisation. Add cross-tenant and cross-origin regression tests to the pipeline.	Prevents recurrence	AI service owner with security engineering	Ongoing, starting in the current cycle

Per-finding summary

ID	What to do	Wave	Effort
F-01	Rotate the credential; gate the write; stop returning secret material	1 and 2	Hours, then days
F-02	Authorise object access per verb in the data-access layer	3	Days
F-03	Gate the write; version and alert on prompt configuration	2 and 6	Days
F-04	Carry the user's identity into every tool call and authorise there	4	Design change
F-05	Narrower scope for lifecycle and ingestion; approval for ingestion runs	2	Days
F-06	Generic error body; authentication at the service, not per route	5	Hours
F-07	Per-account and per-tenant rate limits and a spend budget	5	Days
F-08	Explicit origin allowlist, no reflection, no null, with Vary	2 (early)	Hours
F-09	Use as the reference implementation for wave 2	2	None — observation

Effort is our estimate of the work itself, not of the change control around it, and it is illustrative in this sample like everything else. Owners are the roles we would expect to hold each change; your organisation may map them differently.

If the AI capability came from a platform vendor. Waves 2 and 3 may be the vendor's change rather than yours. Section 6 records the operation and the response returned for each attempt, so the relevant findings can be handed over as they stand, and F-09 gives you a service on the same platform that already enforces the check you are asking for. Meanwhile, the part that is yours is the role mapping: restrict whatever scope covers the AI service to a named administrative group rather than a default business role, and treat enabling an AI capability as a privilege-granting change with the review that implies.

SECTION 8

Retest and closure

Remediation is only finished when someone has attacked the fix. The retest is included as standard — it is not an add-on and it is not quoted separately.

What the retest covers

When your team confirms that remediation is complete, we return and test every finding again. The retest is not a documentation review: each finding is re-attacked using the reproduction path in section 6, so a fix is confirmed by failing to work rather than by being described.

- 1 Every finding is retested, not a sample.** All nine rows in section 5, including the Lows and the positive observation — a control that was working is worth confirming still works after the surrounding change.
- 2 The fix is tested, and so is the way around it.** We check that the reproduction path now fails, and we look for the obvious bypass: a gate applied to one operation but not its sibling, or a credential rotated in one place and not another.
- 3 The enforcement waves are verified operation by operation.** For wave 2 we replay the full administrative operation set with an ordinary token and record each status, so closure rests on an inventory rather than a sample.
- 4 Regression within the affected component.** Where a fix changes authentication, authorisation, tool scoping or output handling, we re-exercise the surrounding functionality — including that the feature still works for the right roles.
- 5 A clear per-finding outcome.** Each finding is reissued as Closed, Partially remediated or Open, with a note on any status that is not Closed. A fresh issue introduced by the remediation work is reported too, though the retest is not a second full assessment.

What the closure letter states

After the retest we issue a signed closure letter alongside the reissued report. It is written to be shared: with customers running vendor due diligence, with whoever runs your AI governance work, and with a board that wants one page.

The letter records

- What was tested, by whom, and over which dates, and the references drawn on.
- How many findings were raised, at which severities.
- How many are confirmed closed, and the date of the retest.
- Any finding that remains open, with its severity and the client's stated plan.
- The name and signature of the consultant who verified it.

The letter deliberately does not record

- Reproduction steps, payloads, evidence or endpoints.
- Service names, tool inventories or architectural detail.
- Anything that would help an attacker who obtained the letter.
- Any claim that the environment is now free of vulnerabilities.
- Any certification or conformity statement — we are not an auditor, assessor or certification body.

Why we include the retest rather than sell it. A fix that was never verified is a plan, not a result — especially where correctness is invisible from the interface.

APPENDIX A

Access, accounts and tooling

AI engagements are unusually sensitive to preparation, because most of the value sits in authorisation testing and authorisation testing is impossible without accounts.

What we asked for, and why

What	Why it matters
Two accounts per role, in the same tenant	Two users at the same privilege level is what makes horizontal access-control testing possible — one reaching the other's conversations and documents. One account per role tests only the vertical boundaries, which is the easier half.
One administrative account	Counter-intuitive, but necessary: a tester must see the administration surface working as intended in order to enumerate it, and can only then attempt each operation from below. Without it the enumeration is guesswork.
A second tenant	Cross-tenant isolation cannot be tested from inside one tenant. Provisioning is often slow, so this is requested before the engagement is booked rather than on day one.
A populated environment	An empty knowledge base and no conversation history means there is nothing to reach across a boundary. Clearly marked synthetic content in the corpus is worth more than production access and avoids a whole category of difficult conversation.
The agent's tool list, with parameters	Every tool the model can invoke, what it does, what it takes and which identity it runs as. Where no such document exists, writing it is valuable on its own — and its absence is usually a finding in waiting.
A written statement of what the feature may do	Which roles may use it, which data domains it may touch, which actions it may take on a user's behalf, what it must refuse. Without it a tester can report what the system does but not where it diverges from what was intended, and that gap is what an authorisation finding is.
Architecture notes and a reachable engineer	Where the orchestration layer sits, which stores hold prompts, embeddings and artefacts, which credentials the pipeline holds, and which parts are vendor-managed. Plus a written decision on whether ingestion and training runs may be triggered.

Tooling used

Stage	Typical tooling	What it was used for
Surface enumeration	An intercepting proxy, route and content discovery, certificate-transparency and DNS enumeration	Establishing the real surface: the AI service's endpoints alongside the application's existing ones, and anything published without meaning to.
Administration-plane testing	An intercepting proxy with manual repeater, and scripted replay of a recorded operation set across several tokens	Replaying every mutating operation with the most ordinary token the tenant will issue, and recording what the server enforced rather than what the interface implied.
Authorisation and isolation	Request-replay tooling driven from an account and tenant matrix, with response comparison	Walking the identifier space in both directions across accounts and tenants, with retrieval, metadata, export and deletion exercised separately.
Agent and injection testing	The product's own interface and API, a controlled corpus of tester-owned content, and tool-call tracing	Establishing the tool contract, then determining what an instruction in ingested content could actually reach through it.
Verification and reporting	Scripted request replay, evidence capture, and an internal reporting pipeline	Reproducing each finding independently, capturing evidence, and generating this document and the machine-readable export.

Tools do not produce findings. Everything in section 6 was verified by hand and, where safe, exercised to demonstrate real impact. A tool's output is a candidate for investigation, never a line in the report — and for AI features that matters more than usual, because a generated corpus of injection attempts will produce a great deal of output and almost no findings. Candidates that proved not to reach anything in context were discarded rather than reported.

APPENDIX B

Out of scope, and what this test is not

What was excluded, why, and what that means for the assurance this report provides. Anything listed here was not tested, and nothing in this report should be read as a statement about it.

Excluded	Reason	What that means for you
The third-party hosted model service	Not the client's to authorise. Testing a system you do not own requires that owner's written authorisation, and no provider grants it to a customer's tester.	In scope was your application, orchestration layer, prompts, data path and credentials — including how your side handles what the model returns. Anyone offering to test the underlying model service cannot lawfully deliver it.
Denial-of-service and stress testing	Excluded by agreement. Deliberately exhausting a production service is indistinguishable from an outage, and for a metered AI endpoint it is also expensive.	F-07 records the absence of a rate limit as an observation from a bounded, authorised burst. It is not a measurement of how the service behaves under a volumetric attack.
Model safety red-teaming	A different exercise with different inputs: it needs a written content policy to measure against, a large volume of attempts and reviewers with the right background.	This report says nothing about whether the assistant can be induced to produce harmful, biased or off-policy content. We report what we stumble into; that is not coverage.
Model evaluation	Accuracy, hallucination rate, retrieval relevance and regression between model versions are measured with a benchmark harness and a labelled dataset.	Nothing here speaks to whether the feature is any good. That is measured continuously by the team that owns it, not in a point-in-time security test.
Governance, assurance and certification	We are an offensive-security testing firm, not an auditor, assessor or certification body.	This report produces technical evidence that whoever runs your governance or assurance work can use. It does not make you compliant with anything and it certifies nothing. See Appendix C.
Social engineering of client staff	Excluded by agreement. Phishing and pretexting are commissioned separately so that consent and the handling of staff data can be agreed on their own terms.	The human layer was not tested. Several findings here would be reached faster by an attacker who phished an ordinary account first — which is, notably, all the privilege most of them need.
Anything not named in the signed scope	A penetration test is authorised against a specific list.	Systems discovered during testing but outside the signed scope were reported as observations and not probed.

Out of scope means untested, not secure. It is the most commonly misread line in any penetration test report. If a system does not appear in section 2, this document makes no claim about it at all — good or bad. Where an exclusion leaves a risk you care about, the debrief is the right place to scope it into the next piece of work.

APPENDIX C

References, and what they are

This engagement drew on the documents below. Each is described for what it genuinely is. None of them is a standard this report certifies anything against.

Reference	What it is	How it was used here
OWASP Top 10 for LLM Applications	A community-maintained list of the risk categories most commonly seen in applications built on large language models. An awareness and coverage document, not a control framework and not certifiable.	As a coverage checklist at scoping and again at review, to confirm no category had been left unexamined. Several findings here map to more than one category and some categories produced nothing.
OWASP Agentic Security Initiative	An OWASP project documenting threats and reference architectures specific to agentic systems — tool use, autonomy, memory and delegation. A public reference, not a standard.	To frame the agent and tool authorisation testing behind F-04, in particular the confused-deputy pattern where a tool authorises the orchestrator rather than the caller.
MITRE ATLAS	A knowledge base of adversary tactics and techniques observed against AI-enabled systems, structured in the manner of ATT&CK. A taxonomy, not a certification.	To name techniques consistently in the finding class lines, so that your own detection and threat-intelligence work can map to the same vocabulary.
NIST AI Risk Management Framework	A voluntary framework published by NIST for managing risk across the lifecycle of an AI system, organised around the functions govern, map, measure and manage. It is not a certifiable standard and NIST issues no certification against it.	As context. Where an organisation's own AI risk programme calls for independent technical testing of a system, a report of this kind is an input to that activity. We make no assessment of your programme and no statement about your alignment with it.
OWASP API Security Top 10 and the OWASP Web Security Testing Guide	Established public references for the ordinary web and API layer. Testing methodology and awareness material.	The AI endpoints are web endpoints, and the hardening findings F-06 to F-08 come straight from this layer. AI features are often built quickly and beside the main application, so they miss framework-level protections the rest of the platform gets for free.
ISO/IEC 42001 and the EU AI Act	A management-system standard and a legal instrument respectively. Both are conformity regimes operated by accredited certification bodies and regulators, not by testing firms.	Mentioned only because clients ask where testing fits. It fits as evidence, handed to whoever runs your conformity work. We are not an auditor, assessor or certification body; we issue no certificate or attestation, and nothing in this report should be presented as one.

What this report is, in one line. It is the record of an offensive-security test of a specific set of systems over a specific window, written so that the findings can be reproduced, fixed and verified. It is technical evidence. It is not an audit, an assessment, an attestation, a certification or a statement of conformity with any framework, and PXLSecurity LTD. holds no accreditation to issue one.

APPENDIX D

How to read this report

Different readers need different parts of this document. Nobody needs all of it, and reading it front to back is usually the slowest way to act on it.

› **If you are an executive or a board member**

Read section 1 and stop. It states what was tested, how serious the result is and the three things to fix first, without jargon. Section 4 explains the severity words.

› **If you own the remediation programme**

Read sections 1, 5 and 7. Section 7 is the plan: waves, owners and sequence. Resist splitting wave 2 into three tickets — it is one enforcement decision with three confirmations.

› **If you are an engineer fixing a finding**

Go straight to your finding in section 6. The reproduction steps let you see the issue before you change anything, and confirm the fix by running them again and watching them fail. The recommendation is ordered: the first item is what to do today, the rest is what makes it stay fixed.

› **If the AI capability came from a platform vendor**

Read section 6 for the operations and responses, and the callout at the end of section 7. The relevant findings can be handed to the vendor as they stand, with F-09 as the internal reference implementation to point at. The role mapping in your own tenant is the part that is yours to change today.

› **If you are an auditor, or run the AI governance programme**

Read sections 2, 3 and 4 for what was tested and how, Appendix C for what the references are and are not, then section 5 for the result. After remediation, the closure letter in section 8 is the document intended for you.

› **If you are deciding whether to commission a test**

This document is the deliverable. What you would receive is this structure filled with your systems: evidence, a reproduction path your developers can follow, a fix that fits your stack, a sequenced plan and a retest.

The anatomy of a finding

Every finding in section 6 carries

- **Identifier and title** — stable, so it can be tracked to closure.
- **Severity and CVSS v3.1** — the rating, with the score behind it.
- **Affected component** — what to go and look at.
- **Finding class** — the CWE or technique reference.
- **Description** — what was found and why it matters.

- **Evidence** — the capture that establishes it.
- **Reproduction** — advisory-level steps, no working payload.
- **Impact** — what it costs the business if it is used.
- **Recommendation** — a specific fix, in the order we would apply it.
- **Status** — Open at issue; updated at retest.

The payload is not the finding. What it reached is.

APPENDIX D · THE SHORT VERSION

This is the deliverable. Now have it written about your AI feature.

Everything in this sample — the evidence, the reproduction paths, the fixes, the sequenced plan, the retest — is what a PXLSec engagement produces. The only thing missing is your environment. Tell us what you want tested and a senior consultant, not a salesperson, will come back with questions, a proposed approach and a quote.

- AI and LLM penetration testing — the administration plane first, then isolation, agents and injection.
- Penetration testing — web, API, mobile, cloud, internal and external infrastructure.
- Active Directory and internal network assessment, where most internal compromises actually begin.
- Red teaming and adversary simulation, reported as a narrative of how the environment really falls.
- Retests included as standard, because a fix that was never verified is a plan, not a result.

pxlsec.com/contact · hello@pxlsec.com

PXLSecurity LTD. · 12 Vasil Levski Blvd., 1142 Sofia, Bulgaria · VAT BG203239331 · independent since 2014

This document is a composite sample. It was assembled from the shape of several anonymised PXLSec engagements and rewritten with synthetic, illustrative data. It describes no single real client. All names use the reserved example.test domain, TEST-NET addresses in 192.0.2.0/24 and the fictional tenants t-larkmoor and t-pentreath; all accounts, identifiers, keys, transcripts and CVSS scores are invented for illustration; no vendor, platform or product is named. No client name, hostname, address, credential, person or engagement date appears anywhere in it. PXLSecurity LTD. is an offensive-security testing firm and is not an auditor, assessor or certification body; nothing in this document certifies or attests conformity with any framework.