SPECIMEN — FICTIONAL ENGAGEMENT

Production-Readiness Audit

Project Basalt and every company, person, finding and line of evidence in it are invented; the method, structure and evidence discipline are exactly those of a real engagement.

This is a complete sample report from our production-readiness audit — the document a client receives, from the executive summary and scorecard to the evidence-backed risk register, remediation roadmap and behavioral AI test. It is written for the people who commission such audits: CTOs, boards, investors and enterprise buyers running vendor assurance. A commissioned audit delivers this exact structure, applied to your systems.

This is a specimen report. The subject, Project Basalt, with its company, people, findings, numbers and code references, is entirely fictional and synthetic, composed to show what a real engagement delivers. The methodology, the evidence rule, the register discipline and the report structure are the real deliverable.

In this specimen the engagement runs three weeks with a lead auditor and a security/AI specialist — parameters a real engagement scopes to the subject's size and stakes; the method is the constant. Access is read-only throughout: repositories, CI/CD, cloud configuration, dashboards and documentation, complemented by interviews with nine roles across engineering, product and operations. Every claim in the final report is tied to evidence the auditors inspected themselves, and the draft is shared with the teams for factual correction before ratings are finalized. The one artifact the audit leaves behind — the behavioral test harness — is delivered as a pull request reviewed and merged by the client's own engineers.

// ENGAGEMENT

Duration: 3 weeks (W1-W3), re-audit delta at +6 weeks

Team: Lead auditor + security/AI specialist

Access: Read-only: repos, CI/CD, cloud config, dashboards, docs

Interviews: 9 roles across engineering, product and operations

Subject profile: European insurtech scale-up (fictional): ~340 staff, ~55 engineers, 7 squads, 24 services

AI features in scope: intake-extractor, adjuster-copilot, fraud-screener

Report scope: 7 blocks, 32 areas, 26 register findings (4 P0 / 8 P1 / 10 P2 / 4 P3)

// DELIVERABLES

✓ Full audit report: 7 blocks, 32 rated areas, every finding backed by file:line or configuration evidence

✓ Risk register: the single source of truth, 26 findings with owners, priorities and status

✓ Remediation roadmap: phased fixes sequenced by priority and dependency

✓ Go-live gate checklist: the explicit conditions attached to each launch decision

✓ Behavioral AI test harness and report: generated probes run in safe mode, left in the repo and re-run before each release

✓ Re-audit delta: a follow-up pass that verifies closures against commit evidence and re-scores the areas

// SAMPLE REPORT · PROJECT BASALT (FICTIONAL)

01 Executive summary

2.9/5 Conditionally ready 4 × P0 · 8 × P1 · 10 × P2 · 4 × P3

Project Basalt arrives at this audit with more engineering discipline than most scale-ups at its stage. Delivery is trunk-based with protected mainlines and roughly daily deploys; all four DORA delivery metrics are computed automatically and trended per squad; the claims core shows clean, mechanically enforced service boundaries backed by 118 reviewed ADRs; and the incident culture, with named roles, a severity matrix and blameless postmortems, is practiced rather than aspirational. The overall rating is 2.9 out of 5 — the mean across the 32 rated areas: conditionally ready.

The verdict splits in two. For the core claims platform, including the two pending enterprise onboardings, the audit returns a GO, conditional on closing three launch blockers: the cross-tenant analytics export (R-01), the unexercised disaster-recovery path (R-02) and the prompt-injection route past the reserve-approval gate (R-03). For general availability of adjuster-copilot to all tenants, the audit returns a NO-GO until all four P0 findings are closed and verified and an evaluation gate stands in front of every prompt and model change (R-04). The existing copilot pilot may continue once the approval gate is enforced in the tool executor rather than only in the UI.

What drives the conditions is a consistent pattern rather than four isolated defects: controls that exist in one layer are assumed, not verified, in the next. The tenant guard holds on the transactional path, but the analytics export trusts a caller-supplied tenant list; backups run on schedule, but no restore drill is on record, so the stated RTO of 4 hours and RPO of 15 minutes are claims, not capabilities; the approval gate is checked in the UI but not in the tool executor; model changes ship behind a floating alias with no golden set, which is how a vendor version bump degraded extraction for 9 days unnoticed. Each P0 carries a concrete, phased fix: re-derive tenant scope from the auth context, run a full DR exercise and reconcile the standby region, enforce approval in the executor and tag retrieved content with its provenance, and pin model versions behind a CI evaluation gate.

None of this is a rebuild. All four P0s are closable in weeks, not quarters, and the strengths that make this plausible (uniform quality gates on all 24 services, a versioned requirements baseline with typed trace links, prompts already under code review in a registry) are the same foundations the fixes will stand on. The eight P1 findings set the agenda for the following quarter; the P2 and P3 items are planned improvements and housekeeping, sequenced in the remediation roadmap. That roadmap totals roughly 55 engineer-weeks across three phases (15, 18 and 22) — comfortably within a 55-engineer organization's quarter — and the commercial stakes are concrete: both pending enterprise onboardings carry vendor-assurance clauses that today's P0 list would fail.

// WHAT HOLDS

- ✓ Trunk-based delivery with protected mainlines, mandatory CODEOWNERS review and roughly daily deploys; all four DORA delivery metrics computed automatically and trended per squad.
- ✓ Clean, mechanically enforced service boundaries in the claims core, backed by a current context map and 118 reviewed ADRs.
- ✓ Uniform, blocking quality gates (lint, type-check and coverage) across all 24 services, with mutation testing on the payout engine.
- ✓ A versioned requirements baseline under change control with typed trace links: a 20-requirement sampled walk ran clean from story to PR to test to release.
- ✓ A practiced, blameless incident culture with named roles, a severity matrix and tracked postmortem actions.
- ✓ AI compliance groundwork further along than typical at this stage: GDPR Art. 30 records, pre-log redaction and immutable decision logs verified in place.

// LAUNCH BLOCKERS (P0)

- ◆ A partner token scoped to one tenant can export another tenant's claims, because the analytics export trusts a caller-supplied tenant list instead of re-deriving it from the authenticated context (R-01).
- ◆ Disaster recovery has never been exercised end to end, so the recovery targets promised to insurers are currently claims the platform cannot demonstrate (R-02).
- ◆ A poisoned claim PDF can steer adjuster-copilot's tools past the human-approval gate, which is enforced in the UI but not in the tool executor (R-03).
- ◆ Prompt and model changes ship with no evaluation gate, so a routine vendor model bump silently degraded document extraction for nine days before a complaint surfaced it (R-04).

BLOCK	RATING	SUMMARY
A · Product & requirements	4/5 (3.5)	Requirements management is the strongest part of this block: a versioned baseline under change control, typed trace links that walk cleanly from requirement to release, and NFRs recorded as measurable, owned targets. Goal-setting is also in place, with OKRs decomposed to owned KPIs and a single merge-checked Definition of Done. The dominant gap is commitments and acceptance criteria that live outside the tooling: two contracted partner features never entered the baseline, and release acceptance rests on declared sign-off rather than executable gates, with no error-budget policy or CI eval gate behind it.
B · Delivery & organization	4/5 (3.7)	Delivery is the strongest block of this audit: trunk-based development with protected mainlines, mandatory review under CODEOWNERS, roughly daily deploys, and all four DORA delivery metrics computed automatically and trended per squad. The residual risk is organizational rather than procedural: the payout engine depends on two maintainers with no succession plan, and onboarding averages ~10 working days to first merged PR against README-level documentation.
C · Architecture, code & data	3/5 (3.3)	The claims core is well architected: service boundaries are clean and mechanically enforced, quality gates are uniform and blocking across all 24 services, and performance targets are tested on a cadence rather than asserted. The block is pulled down by tenant isolation (a proven cross-tenant export path in analytics, R-01) and by untested recovery paths: migration rollbacks, restores and event-contract coverage lag an otherwise disciplined delivery machine.

BLOCK	RATING	SUMMARY
D · Quality & testing	3/5 (2.7)	Quality engineering around the claims core is solid: merge-blocking coverage gates on all 24 services, mutation testing on the payout engine, and a traceability matrix generated nightly rather than rebuilt for audits. The weak edges are an end-to-end quarantine that has grown to 41 specs (14%) with no budget or owner, and an AI layer that entered the audit with no behavioral test coverage at all. The harness delivered during the engagement gives a first measured baseline and is re-run before each release, but it is not yet a blocking gate on prompt or model changes; that gate is a condition for adjuster-copilot GA.
E · Security & compliance	3/5 (2.7)	Secrets live in a managed platform with audited access, SAST gates most services, and a culture of blameless postmortems and commissioned penetration tests gives security work a solid base. The dominant gap is verifiable assurance: no SBOMs or build provenance, unpinned base images, a CI workflow printing runtime secrets into readable logs, and claim-attachment retention that contradicts signed DPAs. The controls that exist are real; the evidence trail that enterprise insurers and regulators will ask for is largely missing.
F · Operations & reliability	3/5 (2.8)	Operations rests on a genuinely strong delivery foundation: reviewed pipelines as the only path to production, disciplined infrastructure-as-code in the primary region, and a practiced, blameless incident culture. Observability coverage is broad but uneven, and SLO and cost governance lag behind the tooling. The dominant gap is resilience: disaster recovery has never been exercised end-to-end, so the stated RTO of 4 hours and RPO of 15 minutes are currently claims, not capabilities.
G · AI systems	2/5 (2.2)	Real engineering intent is visible in this block: prompts and model configs live in a versioned prompt-registry, the behavioral harness found no cross-tenant retrieval leakage in 402/402 relevant probes, and compliance groundwork (Art. 30 records, immutable decision logs) is further along than typical at this stage. It is still the weakest of the seven blocks: the approval gate on state-changing tools is enforced only in the UI and was bypassed by a poisoned claim PDF in staging, and model and prompt changes ship with no evaluation gate — a vendor version bump degraded extraction for 9 days unnoticed shortly before the audit. Both P0s gate adjuster-copilot GA and are closable in weeks, not quarters.

The engagement covered the production platform of 24 services, its delivery and operations toolchain, and the three AI features, over three weeks of read-only access to repositories, CI/CD, cloud configuration, dashboards and documentation, plus interviews with nine roles. One rule governs everything in this report: no finding enters it without evidence — a repo-relative file and line reference, a configuration excerpt or a dashboard reading the auditors inspected themselves. Interview statements set direction; only verified evidence sets ratings.

// HOW THE AUDIT WORKS

02 Method

- Evidence first: every finding carries a repo-relative file:line, configuration or dashboard reference; nothing enters the report on interview testimony alone.
- Stable IDs: findings keep their register IDs (R-01 onwards) across the report, roadmap and re-audit, so remediation can be tracked without re-mapping.

- Single source of truth: the risk register is the one canonical list of findings; every narrative section references it rather than keeping its own.
- Closure only on evidence: a finding becomes closed-verified only when the auditor has re-read the fixing commit or PR, never on a status update.
- Strengths are findings too: what the team does well is recorded with the same specificity, because it tells the roadmap what to build on.
- Factual correction, not negotiation: the draft is shared with the teams to correct facts before ratings are finalized; the ratings themselves are not negotiated.
- Two DORAs appear in this report: the DORA delivery metrics (DevOps Research and Assessment) and EU DORA, the EU's Digital Operational Resilience Act for financial entities. They are unrelated; the regulation is always written as EU DORA.

// AREA RATING, 0-5

0 **Absent** — No practice in place; the capability is unrecognized and unowned.

1 **Initial** — Recognized and owned, but ad hoc or unverified; results depend on individual effort.

2 **Emerging** — A deliberate practice is taking shape, but coverage is partial and consistency depends on individuals.

3 **Defined** — The practice is documented, owned and consistently followed; gaps are known and bounded.

4 **Managed** — The practice is enforced and measured; residual gaps are named, tracked exceptions.

5 **Optimized** — The practice is continuously measured and improved, and demonstrably prevents whole classes of failure.

// FINDING PRIORITY

P0 **Launch blocker** — Must be closed and verified before the gated launch or onboarding proceeds.

P1 **Fix within a quarter** — Material risk to operations, security or compliance; scheduled into the next quarter's plan.

P2 **Planned improvement** — Worth fixing on a planned cadence; enters the normal backlog with a named owner.

P3 **Housekeeping** — Low-risk hygiene; batched into regular maintenance work.

Each entry carries a stable ID, a title, the affected area(s), a likelihood-times-impact rating that sets its priority, an owning squad and a status.

Status is one of open, in progress or closed-verified; closed-verified requires commit or PR evidence re-read by the auditor.

Every finding names exactly one owning squad, so no entry can sit unowned between teams.

Priority follows likelihood times impact against the launch decisions at stake, not how uncomfortable the finding is.

> The 0-5 area ratings in this report are a production-readiness instrument, deliberately distinct from dev-ai.dev's Agentic Engineering Maturity ladder (Stages S0-S4). A maturity assessment on that ladder is the Phase 0 service, a different engagement with a different question. The overall score here is the plain mean of the 32 area ratings; the maturity ladder's weakest-link rule does not apply to it.

// 32 AREAS IN 7 BLOCKS — WHAT WE CHECK, AND HOW BASALT SCORED

03 Scorecard & framework

A · Product & requirements

4/5

A1 Requirements, traceability & scope

4/5

Versioned baseline with typed trace links; a 20-requirement sampled walk ran clean from story to PR to test to release; two signed partner commitments never entered it.

- A versioned requirements baseline with stable requirement IDs is kept under change control in the requirements-management/ALM platform, with an approval workflow, a cross-team change-control board, and recorded impact analysis for every scope change.
- Each release has an explicitly agreed scope boundary — what is in and out of the go-live or release-train commitment — signed off by the accountable business owner, so production readiness is decidable against a fixed baseline rather than a vision document.
- Bidirectional traceability (requirement ↔ story ↔ code/PR ↔ test ↔ release) is generated automatically from the ALM platform and CI, and the full chain can be walked end to end for any sampled requirement.
- Every requirement declares a verification method (test, analysis, inspection or demonstration) and carries a live verification status fed from CI, having passed a definition-of-ready testability review (e.g. INVEST) before entering a sprint.
- Non-functional requirements are recorded in the baseline as measurable targets — latency percentiles, concurrency, availability, capacity, data residency, accessibility, internationalization — each with a named owner, never as qualitative mentions.
- Regulatory obligations are mapped row-by-row to requirements with named control owners: GDPR Art. 30 records of processing and DPIA triggers; EU AI Act (as amended by the 2026 AI Omnibus, Reg. 2026/1744) Art. 50 transparency obligations in force since 2 Aug 2026 and Annex III high-risk obligations applying from 2 Dec 2027; and EU DORA where in scope.
- Cross-team ownership is explicit: every requirement domain has an owning team and RACI recorded in the service or requirements catalog, and inter-team API contracts (OpenAPI/AsyncAPI) are treated as versioned requirements in their own right with a published deprecation policy.
- Delivered scope is continuously reconciled against the baseline from tracker, feature-flag and deployment state — with drift alerts for silently unimplemented scope and symmetric detection of shipped features that appear in no agreed requirement.

2 Some teams keep structured backlogs with acceptance criteria, but the organization as a whole still relies on vision-level documents and hand-maintained matrices: traceability is reconstructable only through one-off manual effort, NFRs remain qualitative, and scope-versus-delivery drift surfaces only when someone audits it. 4 A tool-maintained baseline under change control spans every contributing team, bidirectional traceability and per-requirement verification status are generated automatically from the ALM platform and CI, NFR and regulatory scope sit in the baseline as first-class rows with named owners, and drift between agreed and delivered scope raises alerts instead of being discovered at audit time.

A2 Business goals, success metrics & acceptance

3/5

OKRs map to owned KPIs and merge checks enforce one Definition of Done, but there is no error-budget policy, CI eval gate or copilot cost tracking.

- Business objectives decompose into a documented metrics tree — OKR-to-KPI with a north-star metric and unit economics such as cost per transaction or per tenant — where each metric carries a numeric target and a named owner, shown on live dashboards and reviewed on a fixed executive cadence.
- A single organization-wide Definition of Done (functionality, tests, security review, documentation, observability hooks) is enforced by merge and release gates in the delivery tooling, not negotiated per team or module.
- Release acceptance criteria are formalized as executable specifications wired into CI, so acceptance of each release is mechanically checkable rather than declared.
- Reliability targets are adopted as declared success metrics: SLOs with error budgets and a written error-budget policy (Google SRE practice) stand as acceptance constraints that demonstrably gate release pace when budgets burn.
- The DORA four key delivery metrics (deployment frequency, lead time, change failure rate, failed-deployment recovery time) are trended per team as organizational KPIs on delivery dashboards.
- Teams using AI-assisted development pair throughput gains with declared stability safeguards — review discipline, rollback capability, change-failure monitoring — reflecting the 2025 DORA State of AI-assisted Software Development finding that AI lifts throughput but amplifies delivery instability without them.
- For AI features, success governance aligned to ISO/IEC 42001:2023 and NIST AI RMF 1.0 — model quality thresholds, drift-monitoring KPIs, human-in-the-loop correction-rate targets and an eval regression gate — is recorded as release acceptance criteria in CI and release records.
- Launch acceptance has named authority: staged gates (internal, beta, GA) each carry sign-off criteria, an accountable accepting role, auditable sign-off records per release, and segregation of duties between builders and acceptors.

2 A clear qualitative product vision exists and some teams track metrics or demo against informal criteria, but KPI targets are unquantified or inconsistent across the organization, Definition of Done varies by team, and acceptance is declared in meetings rather than recorded against mechanically checked criteria. 4 Numeric KPI targets, SLOs with an enforced error-budget policy, per-team DORA delivery metrics and executable acceptance gates — including AI eval regression gates — govern every release, each accepted through an auditable sign-off by a named accepting authority.

B · Delivery & organization

4/5

B1 Ownership, topology & knowledge continuity

3/5

All 24 services carry a cataloged owner and CODEOWNERS approval, but the payout engine rests on two maintainers and onboarding runs ~10 working days to first PR.

- Every production service, repository and data store has exactly one accountable owning team recorded in a service catalog with criticality tier and on-call rotation, and a periodic scan confirms no orphaned assets.
- Code-owner rules enforce owning-team approval on critical paths — security-sensitive modules, shared libraries, infrastructure and pipeline code — so no change merges without it.
- Team topology and inter-team dependencies are mapped against service boundaries and reviewed on a fixed cadence.
- Knowledge concentration is measured from commit and review history, with at least two active maintainers on every tier-1 component and a documented succession or rotation plan for single-maintainer hotspots.
- Cross-team design decisions are captured in a searchable ADR/RFC log with a defined review gate for high-risk or cross-boundary changes.
- Onboarding is a managed, per-role path measured by time to first merged change, with guides kept current under a named owner.
- A joiner/mover/leaver process governs repository and environment access, contractors and vendors included, with periodic access reviews evidenced per ISO/IEC 27001 and SOC 2 Type II expectations.
- Every AI system or model in production has a named accountable owner in a maintained AI inventory, as ISO/IEC 42001:2023 and NIST AI RMF 1.0 expect.

2 Ownership is written down for some services and a few teams keep ADRs and code-owner rules, but the catalog is incomplete or stale, several tier-1 components depend on a single maintainer, and access reviews happen ad hoc. 4 Every asset has a cataloged owner with tier and on-call cover, code-owner approval is tool-enforced on critical paths, knowledge-concentration and onboarding metrics are reviewed on a fixed cadence, and access reviews run on schedule with exceptions logged.

B2 Change process & code review

4/5

Trunk-based development, protected mainlines, mandatory CODEOWNERS review and ~daily deploys; the floating model alias in prompt-registry sidesteps that discipline.

- Mainline branches are protected in every production repository — no direct commits, required status checks, at least one independent reviewer — and any bypass is logged and reported, supporting SLSA v1.2 Source-track expectations.
- High-risk paths additionally require code-owner approval, and segregation of duties between author, approver and deployer is evidenced in system records, as SOC 2 Type II and ISO/IEC 27001 change-management controls expect.
- Every production change traces from work item to merge request to pipeline run to deployment, and the chain holds up under sampling.
- Changes are risk-tiered, with an emergency-change path that requires retrospective review and approval rather than a silent bypass.
- Model, prompt, configuration and infrastructure changes pass through the same versioned, reviewed change process as application code, in line with ISO/IEC 42001:2023.
- A written policy governs AI-assisted and agent-authored changes — disclosure, review depth, merge guardrails — reflecting the 2025 DORA finding that AI lifts throughput but amplifies delivery instability without safeguards.
- Review health is measured and acted on: change size, review turnaround and rubber-stamp approval rates are tracked per team.
- Releases are versioned with release notes and an auditable approval trail, so any deployed artifact maps to a reviewed change set.

2 Merge requests and protected branches exist in most repositories, but enforcement varies by team, bypasses go unlogged, emergency changes skip review, and model or configuration changes travel outside the standard process. 4 Branch protection, independent review and author/approver/deployer segregation are tool-enforced across all repositories, every change traces from ticket to deployment, AI-assisted and emergency changes follow written policies, and review-health metrics are reviewed with exceptions logged.

B3 Planning & delivery metrics

4/5

All four DORA delivery metrics auto-computed from CI/CD and incident data, trended per squad; the error-budget policy is still unwritten (tracked under Operations).

- The four DORA delivery metrics — deployment frequency, lead time, change failure rate and failed-deployment recovery time — are computed automatically from delivery systems and trended per team.
- Delivery dashboards and planning-review minutes present throughput and instability metrics together, segmented where AI-assisted development is in use, per the 2025 DORA State of AI-assisted Software Development finding that AI lifts throughput but amplifies instability without safeguards.
- Work is managed in a tracker with objective-to-epic-to-work-item traceability, so any shipped change links back to a planned outcome.
- Flow metrics — work in progress, cycle time, ageing work — are visible per team and demonstrably used in planning reviews.
- Capacity allocation across feature work, technical debt and compliance work is explicit in each planning cycle and compared against actuals.
- Cross-team dependencies and delivery risks are tracked in a shared register with named owners and dates, reviewed on a fixed cadence.
- A written error-budget policy, per Google SRE practice, demonstrably alters release pace or planning priorities when budgets burn.
- Regulatory milestones are dated, owned roadmap items — for example EU AI Act Art. 50 transparency obligations in force since 2 Aug 2026 and Annex III high-risk obligations applying from 2 Dec 2027.

2 Some teams hand-collect velocity or a subset of delivery metrics in spreadsheets and keep a backlog, but trends and cross-team dependencies are not reviewed and no written policy connects stability data to planning decisions. 4 All four delivery metrics are auto-computed and trended per team, planning reviews work from flow and stability data with explicit capacity allocation, and owned regulatory and contractual milestones visibly shape release planning.

C · Architecture, code & data

3/5

C1 Architecture & service boundaries

4/5

A current context map and 118 reviewed ADRs back clean claims-core boundaries enforced by dependency-cruiser CI gates; 9 of 24 services carry no ADR trail.

- A versioned target-state architecture and a current context map of services and bounded contexts exist and have been reviewed within the last twelve months.
- Every production service has exactly one owning team recorded in the service catalog, with cross-team dependencies mapped and aligned to team structure.
- Architecture decision records capture significant choices with status, consequences and a defined review cadence, and new services reference the decisions they build on.
- Dependency and layering rules are enforced mechanically as CI fitness functions, so boundary violations fail the build rather than accumulate as drift.
- Shared mutable state (sessions, workflow state, cross-request buffers) is externalized to declared stores, with no process-local state that blocks horizontal scaling, verified in code and deployment configuration.
- AI components respect an authority boundary — models propose (classify, extract, draft) while state-changing actions execute through deterministic, authorized code behind risk-tiered human confirmation — limiting Excessive Agency as described in the OWASP Top 10 for LLM Applications (2025).
- Technology choices follow a sanctioned-stack or golden-path policy with a documented, time-bounded exception process.

2 Diagrams and a few decision records exist in pockets but are stale and unenforced; boundaries hold by convention, ownership of several services is ambiguous, and shared state still sits inside individual processes. 4 A current context map with single-team ownership of every service is backed by CI-enforced boundary rules, decision records reviewed on a cadence, and a tracked exception list, with violations measured and trending down.

C2 API & contract governance

3/5

All 6 Kafka event schemas sit in a registry, but only 4 carry consumer contracts and the partner-facing API has no versioning or deprecation policy.

- Every exposed API (request/response, RPC and event-based) has a machine-readable schema in version control as the single source of truth, never hand-maintained strings scattered through code.
- Breaking-change detection (schema diff) runs in CI on every provider change, and consumer-driven contract tests run in both provider and consumer pipelines across team boundaries.
- A published versioning and deprecation policy defines sunset timelines, and every deprecated surface carries an announced date communicated to consumer teams.
- Consumed third-party APIs are pinned to a schema snapshot in the repository and validated in CI by code generation or schema diff, so upstream drift fails the build instead of production.
- An internal API catalog records every API with owner, consumers and lifecycle status, and new surfaces pass automated style-guide linting plus design review.
- Event and message schemas are governed through a schema registry with declared compatibility rules, so producers cannot publish changes that break downstream consumers.

2 Schemas exist for some APIs but drift from the implementation, breaking changes are discovered by consumers in production, and third-party contracts are exercised only at runtime. 4 All APIs have registered, owned schemas guarded by CI breaking-change and contract-test gates on both provider and consumer sides, deprecations follow tracked sunset dates, and upstream drift is caught before deployment.

C3 Code quality & maintainability

4/5

Blocking lint, type-check and CODEOWNERS review gates cover all 24 services; DORA delivery metrics are reviewed fortnightly and the debt register names owners.

- Linting, formatting, static analysis and type checking run as blocking CI gates on every merge request in every repository, with a ratchet policy that stops legacy code getting worse.
- Code review is mandatory and enforced through branch protection and code-ownership rules, producing an approval audit trail that satisfies segregation-of-duties requirements in regulated change management.
- Complexity, size and nesting budgets are enforced or tracked, and churn-versus-complexity hotspot analysis feeds a managed technical-debt register with owners and allocated remediation capacity.
- Error-handling standards forbid broad silent catches: exceptions are typed, logged with context and surfaced to error tracking, with the rule enforced by static analysis rather than convention.
- Data crossing module or service boundaries uses typed models or schemas instead of untyped maps, enforced by type checking in CI.
- The four DORA delivery metrics (deployment frequency, lead time, change failure rate, failed-deployment recovery time) are measured per team and reviewed on a cadence as quality signals.
- AI-assisted changes are identifiable and pass the same review and test gates as human-written code, with their change failure rate and rework tracked against throughput, reflecting the 2025 DORA State of AI-assisted Software Development instability findings.
- Bus factor is assessed for load-bearing modules, with ownership and knowledge-transfer actions tracked for any component below the agreed threshold.

2 Quality tooling is configured in some repositories but gates are advisory or bypassed, review depth varies by team, and technical debt is known anecdotally rather than tracked or budgeted. 4 Uniform blocking quality gates and enforced review policy cover every repository, delivery metrics and hotspot trends are reviewed on a cadence, and the debt register shows owners, budgets and measurable burn-down.

C4 Scalability & performance

4/5

Per-journey p95 targets are enforced by a recurring k6 gate ramping to planned peak, inference in scope; happy-path only, with no 10x catastrophe FNOL model.

- Documented capacity targets (p95/p99 latency, throughput, concurrency) exist per critical journey and service, with current headroom reported against them.
- Load, stress, soak and spike tests run as recurring release gates in the delivery pipeline with baselines and regression thresholds, not as one-off exercises.
- Autoscaling behavior — scale-out, scale-in, warm-pool and cold-start budgets — is validated under test for every tier that claims horizontal scalability.
- Admission control is configured at every public and shared internal entry point: rate limiting, load shedding and backpressure, with queue depth and shed rates observable.
- AI inference workloads have dedicated capacity management: request queueing, concurrency caps, batching and utilization targets, with model-level latency and throughput telemetry such as time-to-first-token and tokens per second.
- Fan-out paths carry explicit tail-latency budgets per hop, caching tiers have a declared invalidation policy, and third-party quota limits are documented with degradation plans for when they are hit.
- Capacity planning runs on a cadence, with growth forecasts and unit economics (cost per request, per user, per tenant) feeding scaling decisions.

2 Some load tests have been run and the worst bottlenecks fixed, but targets are informal, known scaling blockers remain in code, and capacity grows reactively after incidents. 4 Published capacity targets per journey are enforced by recurring load-test gates, autoscaling is test-validated with headroom reported on a cadence, and inference capacity is queued, capped and measured.

C5 Resilience & fault tolerance

3/5

Timeouts, circuit breakers, DLQs and a payout outbox are standard in the NestJS service template, but DR has never been exercised end-to-end (R-02), capping the rating.

- Every outbound dependency call carries explicit timeouts, bounded retries with backoff and circuit breaking — verified in code and configuration, with retry budgets and deadline propagation across the call graph.
- Services operate to SLOs with error budgets, a written error-budget policy and burn-rate alerting, per Google SRE practice.
- A graceful-degradation matrix documents, for each dependency outage, which features degrade, in what order and with what user messaging, and silent failure modes (errors swallowed into empty results) are treated as defects.
- State-changing operations are idempotent as a platform standard (idempotency keys enforced at the gateway or platform level), and asynchronous pipelines use dead-letter queues, transactional outbox or saga compensation.
- Disaster recovery posture is defined and exercised: service tiering with RTO/RPO per tier, a single-point-of-failure register, declared multi-AZ or multi-region stance, and failover drills with measured recovery times.
- Fault-injection or chaos experiments (dependency kill, latency injection, zone evacuation) run on a cadence against pre-production or production, with findings tracked to closure.
- AI dependencies have tested failure plans: fallback model or provider routing, per-capability kill switches behind feature flags, and rapid model-version rollback with retained artifacts.
- For financial-sector workloads, resilience controls map to EU DORA — covering ICT risk management, digital operational resilience testing and ICT third-party risk for critical or important functions.

2 Timeouts and retries exist on some calls, but failures still surface as silent empty results or raw user-facing errors, recovery depends on manual intervention, and disaster recovery is a document that has never been exercised. 4 Resilience patterns are enforced platform-wide with SLO burn-rate alerting, degradation paths are documented and rehearsed through regular chaos and failover drills, and every critical dependency — AI providers included — has a tested fallback.

C6 Multi-tenancy & isolation

2/5

The row-level tenant guard holds on the transactional path, but analytics export trusts caller-supplied tenant_ids[]; cross-tenant export proven in staging (R-01).

- A documented tenancy model states, per service and data store, the isolation mechanism (separate database, schema or row-level) and a blast-radius analysis for tenant-affecting failures.
- Tenant context propagates end-to-end through requests, jobs, events and logs, and every query path is tenant-scoped by construction — enforced through middleware or database policy and verified by automated cross-tenant access tests in CI.
- Per-tenant quotas, rate limits and fairness controls prevent any single tenant from exhausting shared compute, storage or AI inference capacity.
- Caches, queues, search and retrieval indexes, and AI context (prompts, embeddings, conversation history) are tenant-partitioned so one tenant's data cannot surface in another's results, addressing LLM02 Sensitive Information Disclosure from the OWASP Top 10 for LLM Applications (2025).
- Tenant-scoped operations work at scale and leave evidence: per-tenant backup and restore, export, and verifiable deletion satisfying the GDPR right to erasure.
- Per-tenant observability exists: tenant-level SLIs, usage metering and cost attribution are live in dashboards, so noisy-neighbour effects are visible before customers report them.
- Tenant lifecycle — onboarding, offboarding and migration between tiers — is automated and audited rather than performed by hand.

2 Tenant separation relies on query filters developers must remember, shared caches and indexes are unpartitioned, and nobody can state the blast radius of a tenant-level failure. 4 The tenancy model is documented and mechanically enforced with cross-tenant tests in CI, per-tenant quotas and metering are live, and tenant-scoped restore and erasure have been demonstrated on demand.

C7 Data architecture, migrations & integrity

3/5

Every per-service PostgreSQL store has an owner, classification and monitored encrypted backups, but restores are unexercised and one payout migration is irreversible.

- Every data store has a named owner, a data classification and documented RPO/RTO targets.
- Backups are encrypted, access-controlled and monitored for success, with restores exercised on a schedule and the measured results recorded against RPO/RTO.
- Schema migrations follow a zero-downtime expand/contract policy, are tested in CI against production-like data volumes, and ship with rollback plans and mandatory review.
- Durability-critical state (scheduled jobs, user-facing commitments, workflow state) lives only in durable stores — never in non-persistent caches or brokers — with reconciliation scanners that recover delayed or scheduled work after infrastructure loss.
- Retention schedules are defined per dataset, mapped to legal obligations (GDPR storage limitation and Art. 30 records of processing, plus sector record-keeping), and executed by automated purge or archival jobs with legal-hold support.
- Cross-system integrity is managed: a system of record is declared per entity, reconciliation jobs compare stores on a cadence, and propagation uses outbox or change-data-capture patterns with atomic state updates.
- Growth is governed for every expanding dataset through partitioning, TTL or archival tiering backed by forecasting, with no unbounded append-only tables.
- Load-bearing datasets carry data-quality SLOs (freshness, completeness, duplication rate) with monitoring and alerting.

2 Migrations and backups exist for the main database, but restores are untested, retention is undefined, and some durable commitments sit in non-persistent infrastructure where a restart silently loses them. 4 Every store has an owner, classification and restore-tested RPO/RTO, migrations follow an enforced expand/contract policy rehearsed in CI, and retention, reconciliation and data-quality monitors run automatically with reviewed results.

D · Quality & testing

3/5

D1 Test strategy & pyramid

3/5

Coverage gates block merges for all 24 services and mutation tests guard the payout engine; the e2e quarantine, 41 specs (14%) and growing, has no budget or owner.

- A written test strategy defines the pyramid layers (unit, component, contract, integration, end-to-end), names an owning team per service, and the actual layer mix is measured against it.
- Every repository blocks merges below agreed coverage thresholds, with ratchets on legacy code and mutation testing on business-critical paths.
- Consumer-driven contract tests and schema-compatibility checks gate every inter-service and third-party interface change.
- Integration and end-to-end suites run hermetically in ephemeral per-change environments, with no dependence on shared mutable staging or live credentials.
- Performance and resilience test layers (load, soak, stress, fault injection) exist with pass criteria derived from published SLOs and run before major releases.
- Flaky tests are detected and quarantined under an owned flake budget, with suite health visible on a dashboard.
- Test data for lower environments is synthetic or PII-masked under a documented, GDPR-lawful provisioning and retention rule.
- Escaped defects and change failure rate — one of the four DORA delivery metrics — are tracked per team and reviewed to steer test investment.

2 Real automated tests run in CI in some teams, but the layer mix is unplanned and uneven, gates are advisory or easily bypassed, integration suites depend on a shared mutable staging environment, and flaky tests are simply re-run. 4 The codified pyramid is enforced by merge-blocking gates in every repository, non-functional layers run against SLO-derived criteria, and flake budgets, test-data rules and delivery metrics are measured against thresholds with exceptions tracked.

D2 Requirement traceability & coverage

4/5

The traceability matrix is generated nightly from tracker, PR and test links; 94% of in-scope items walk end to end, with regulatory rows the tracked exception.

- A living traceability matrix is generated from the delivery toolchain with typed links requirement–story–change–test–release, not rebuilt by hand for audits.
- The chain requirement → implementation → test → acceptance can be walked end to end for every in-scope item, with orphaned requirements and untraced tests surfaced automatically.
- Acceptance criteria are executable specifications wired into CI, so the requirement-to-test link is enforced mechanically.
- Coverage is risk-weighted by business criticality, and a requirement counts as implemented only when verified live via feature-flag state or production telemetry.
- Non-functional requirements (SLOs, latency, capacity, availability) have their own coverage rows, each with a named verification method.
- Regulatory requirements are traced to specific clauses and controls — GDPR articles, ISO/IEC 27001 Annex A, SOC 2 criteria, EU AI Act Art. 50 transparency duties where AI features face users — with evidence collected automatically.
- Scope is baselined under change control, with change-impact analysis across traced items and variance tracking that flags out-of-scope delivery as well as gaps.
- Acceptance evidence is audit-grade: immutable sign-off records with segregation of duties between those who build and those who accept.

2 Requirements live in a tracker and some teams link stories to tests, but the matrix is rebuilt by hand when someone asks, whole areas of scope lack linked tests or acceptance criteria, and gaps surface only on inspection. 4 Traceability is generated continuously across every team, each requirement resolves to passing tests and recorded acceptance, non-functional and regulatory rows are covered, and exceptions and drift are tracked against thresholds.

D3 Behavioral AI testing

1/5

Before the audit neither adjuster-copilot nor intake-extractor had any behavioral test; the audit's 412-probe harness now sits in the repo and is re-run each release.

- Evaluation is two-layer: deterministic gates assert tool invocations, arguments and resulting state with binary pass/fail, while stochastic outputs are measured statistically against explicit thresholds.
- AI-claimed outcomes are verified against the system of record (a dual oracle), and any LLM judge is confined to separately reported text-quality opinion, never the headline verdict or numeric correctness.
- An adversarial suite is mapped to the OWASP Top 10 for LLM Applications (2025 edition), with block rates for LLM01 Prompt Injection and LLM02 Sensitive Information Disclosure enforced as a release gate.
- Probe coverage is derived from the product's own capability surface crossed with a misuse-vector taxonomy, so it scales with the codebase instead of a hand-curated script list.
- Models, prompts and auxiliary components are pinned by exact version or digest, and every change runs a regression eval against versioned golden datasets, with degradation versus baseline blocking release.
- Safety-critical behavior — confirmation before irreversible actions, honest refusal of out-of-scope requests, no fabricated values — is enforced by blocking checks under a non-regression guardrail.
- Behavioral stability is measured by re-running identical probes, with agreement rates reported per release.
- Production failures are converted into new eval cases under a documented process, with the eval suite versioned as a release artifact in line with ISO/IEC 42001:2023 and NIST AI RMF 1.0.

2 A demo eval set and some prompt tests exist in a few teams, but hard checks and LLM-judge opinion blend into one score, models float on latest tags, adversarial cases are ad hoc, and nobody can say whether behavior changed since the last release. 4 Every model or prompt change passes a versioned eval gate in every AI-bearing service — deterministic hard checks separated from judge scores, OWASP-mapped attack suites at enforced block rates, stability reported per release — with exceptions tracked and thresholds reviewed.

E · Security & compliance

3/5

E1 Application security

3/5

Gateway auth and blocking SAST on 19 of 24 services, ASVS 5.0 mapping across two squads; authorization depth varies, with cross-tenant export (R-01) tracked under C6.

- Security requirements are baselined against OWASP ASVS 5.0 at a declared verification level, with each applicable control mapped per service to implementation evidence rather than asserted.
- AI-facing features are additionally assessed against OWASP AISVS and the OWASP Top 10 for LLM Applications (2025), with LLM01 Prompt Injection defences designed in at every model-input boundary.
- Every service endpoint enforces authentication and authorization through a centrally governed mechanism (gateway, mTLS or token-based), and no internal API trusts caller-supplied identity.
- SAST and DAST run as blocking merge gates across all repositories, with severity-based remediation SLAs tracked in the organization's work-management system.
- Inputs and uploaded content are validated against actual content and declared schemas (content-type verification, size caps), and internal error detail never reaches end users.
- Model-mediated actions follow least privilege — tool and permission scoping per capability — with human confirmation gates on destructive or irreversible operations.
- A documented secure-development lifecycle defines security requirements, review triggers and sign-off roles per team, and is demonstrably followed in recent changes.

2 Scanners run in some teams but are advisory, authentication patterns vary service by service, and security requirements exist as informal knowledge held by individual engineers rather than an enforced organization-wide baseline. 4 Blocking SAST/DAST gates and centrally enforced authentication cover every repository and service, ASVS/AISVS mappings are current with linked evidence, and remediation SLA compliance is measured and reviewed by security leadership.

E2 Secrets & configuration

3/5

All 24 services read secrets from the managed platform with audited access; one debug step in a Partner Integrations workflow prints runtime-fetched secrets to CI logs.

- All production secrets live in a managed secrets platform with per-access audit logging, and history-wide scanning confirms none exist in repositories, images or plain configuration files.
- Secret scanning with push protection runs organization-wide in CI and pre-commit, backed by a documented revocation-and-rotation runbook for leaked credentials.
- Service-to-service authentication uses workload identity (cloud IAM roles or OIDC federation) rather than long-lived static tokens, with any remaining static credentials inventoried and rotation-scheduled.
- Applications fail fast at startup on missing or invalid required configuration, and no fallback values for cryptographic keys or credentials are committed in code.
- Configuration is typed and schema-validated per environment, with insecure defaults blocked by policy-as-code gates in the pipeline and drift detected across environments.
- Encryption keys are KMS- or HSM-backed with envelope encryption and a written rotation policy defining crypto-periods, producing evidence usable in certification audits.
- Access to production secrets is role-governed with approval workflow, just-in-time elevation and periodic access reviews whose records are retained.
- Third-party credentials are inventoried with named owners, expiry dates and contractual rotation obligations.

2 Secrets are kept out of version control and read from the environment in most teams, but storage, rotation and access control remain manual and inconsistent across the organization, with no audit trail of who accessed what. 4 A centrally managed secrets platform with audited access and automated rotation covers all teams, workload identity has displaced long-lived tokens, and policy-as-code gates plus fail-fast validation are enforced in every pipeline with compliance measured.

E3 Supply chain security

2/5

No build emits an SBOM or SLSA provenance, and 11 of 24 service Dockerfiles pull base images by floating tag; lock files exist but update cadence varies widely.

- An SBOM in CycloneDX or SPDX format is generated and retained for every build artifact, and is queryable organization-wide when a new vulnerability is disclosed.
- Builds meet a declared SLSA v1.2 level on the Build track: signed provenance attestations are produced and verified at deploy time before anything reaches production.
- Source-track controls match a declared SLSA v1.2 level: protected branches, enforced two-party review and code-owner rules on every production repository.
- All dependencies are pinned through lock files with hashes and container images by immutable digest, and a clean-environment build is verified in CI.
- AI components — models, weights, datasets and third-party model APIs — are inventoried, version-pinned and license-cleared, addressing LLM03 Supply Chain from the OWASP Top 10 for LLM Applications (2025).
- Software composition analysis covers direct, transitive and OS/base-image packages, with severity-based remediation SLAs and a documented exception workflow.
- Packages resolve only through a private registry or proxy with quarantine and allow-listing, defending against dependency confusion and typosquatting.
- Automated dependency-update pipelines run continuously behind regression gates, making vulnerability patching routine rather than episodic.

2 Some teams pin dependencies and run occasional scans, but lock-file discipline, SBOMs and artifact signing are absent or team-local, and updates land only when an individual engineer takes the initiative. 4 Every build emits an SBOM and signed provenance verified at deploy, SCA with SLA-tracked remediation spans all repositories and images, and update automation plus registry controls are enforced organization-wide with compliance reported to security leadership.

E4 Threat modeling & security operations

3/5

Blameless postmortems are routine and annual pen tests are commissioned, but two medium findings sit open past 180 days and threat models omit prompt injection.

- A current, signed-off threat model exists for every externally exposed or data-critical service and is refreshed on major architectural change, with AI threats such as prompt injection and data exfiltration through model outputs explicitly in scope.
- Security-relevant events — authentication failures, privilege changes, secret access, administrative actions — feed a centralized, tamper-evident log pipeline with defined retention.
- Detections map to modeled threat scenarios with alerts routed to a staffed escalation path, and alert quality (noise and actionability rates) is reviewed on a schedule.
- Adversarial testing runs at a defined cadence: independent penetration tests for the platform and prompt-injection/jailbreak red-teaming for AI features, with findings remediated within SLA and retested.
- Incident response is rehearsed through tabletop exercises and followed by blameless postmortems with tracked actions, in line with Google SRE practice.
- AI incidents form an operational class of their own, with misbehavior classifications, per-capability kill switches and rapid model rollback with retained artifacts.
- Administrative and debug surfaces are inventoried and either removed or locked down behind network restrictions and multi-factor authentication.
- A vulnerability intake and triage process operates with severity-based SLAs, and residual or accepted risks are recorded in a register with expiry dates and accountable owners.

2 Threat models exist for one or two flagship services but are stale, security logging is partial and inconsistently retained, and incidents are resolved by individual heroics without rehearsed playbooks or AI-specific detection. 4 Threat models are living documents tied to architecture change across all critical services, detections trace back to them with reviewed alert quality, response is rehearsed with postmortem actions tracked to closure, and AI features receive scheduled red-teaming with working kill switches.

E5 Data protection & privacy

2/5

Claim photos and medical attachments are retained indefinitely against a 24-month DPA commitment; no lifecycle deletion rule exists on the claim document store.

- A maintained personal-data inventory and classification spans all stores, automated where feasible, with named data owners and coverage of AI-pipeline intermediates such as prompts, completions, extracted fields and embeddings.
- Logs and telemetry are provably free of unredacted personal data: masking is enforced in the logging layer and verified by sampling, including LLM prompt and completion traces (LLM02 Sensitive Information Disclosure).
- Every store holding personal data enforces authentication, TLS and encryption at rest, with no default-open datastores reachable from the network.
- GDPR operational artifacts are current: Art. 30 records of processing, DPIAs for high-risk processing including AI features, and a documented legal basis per processing activity.
- Retention schedules are enforced by automated deletion, and data subject rights (access, erasure) run as SLA-tracked workflows reaching replicas, backups, analytics stores and any model-training data.
- Personal data shared with subprocessors and third-party model providers is governed by a DPA inventory, transfer assessments (SCCs/TIAs), retention limits and no-training clauses, with minimization at the integration boundary.
- Non-production environments and test fixtures contain no production personal data; masked or synthetic data is the verified default.
- Breach response is ready: a 72-hour notification playbook, a breach register, and defined regulator and data-subject communication paths.

2 Teams can broadly say what personal data they process and some controls exist, but redaction, retention and erasure are partial and vary by service, and artifacts such as the records of processing and DPIAs are incomplete or stale. 4 Automated discovery keeps the inventory current with named owners, redaction and encryption are enforced and sampled, rights requests and retention run as measured workflows covering backups and training data, and DPIAs plus breach playbooks stand ready as audit evidence.

E6 Regulatory & certification readiness

3/5

A regulatory register names owners for GDPR and EU DORA; EU AI Act classification lives in a one-off exposure memo rather than a maintained capability-level register, and the copilot DPIA is unsigned.

- A regulatory applicability register maps each market and feature to its regimes — GDPR, the EU AI Act, and sector rules such as EU DORA — with a named accountable owner.
- Each AI feature carries a documented role and risk classification under the EU AI Act as amended by the 2026 AI Omnibus (Reg. 2026/1744), with Art. 50 transparency obligations, in force since 2 Aug 2026, demonstrably met in the product.
- Where Annex III high-risk uses apply — risk assessment and pricing in life and health insurance among them — a dated readiness plan tracks the obligations deferred to 2 Dec 2027, and Annex I embedded-product AI is tracked for 2 Aug 2028.
- An AI management system aligned to ISO/IEC 42001:2023 (certifiable) or NIST AI RMF 1.0 (voluntary) governs the model inventory, risk tiering and approval workflow.
- Implemented controls are traced in a control matrix to ISO/IEC 27001 Annex A and the SOC 2 Trust Services Criteria, with evidence collection automated where possible to sustain a Type II reporting period.
- Certification posture is deliberate: current certificates or attestation reports exist, or a dated gap assessment and roadmap is owned at executive level.
- Regulated or AI-influenced decisions leave immutable audit trails linking input, model version, prompt version and resulting action, retained per regulatory schedule.
- Incident- and breach-reporting obligations are mapped across regimes with notification clocks and templates: GDPR duties, EU DORA ICT-incident reporting where in scope, and prepared readiness for EU AI Act serious-incident duties as Annex III obligations take effect.

2 People can name the regimes that probably apply and controls exist in places, but there is no maintained applicability register or control mapping, evidence is assembled by hand when someone asks, and certification remains an intention without a roadmap. 4 A maintained register and control matrix with largely automated evidence spans the organization, certification or attestation is current or on a funded dated roadmap, EU AI Act obligations are tracked per feature against statutory deadlines, and reporting duties are mapped with templates and named owners.

F · Operations & reliability

3/5

F1 Environments & deployment

4/5

24 services ship via reviewed GitHub Actions pipelines with canary rollout and daily trunk-based deploys; drift detection covers the primary region, not the standby.

- Production, staging and development environments are defined entirely as version-controlled infrastructure-as-code, with automated drift detection and no out-of-band changes to production.
- Every service ships through a reviewed CI/CD pipeline that is the only path to production, deploying signed, digest-pinned artifacts from a governed registry (SLSA v1.2 Build track).
- Customer-facing services use progressive delivery — canary or blue/green with feature flags — with automated rollback criteria wired to health metrics and rollback rehearsed, not assumed.
- Database and schema changes follow a zero-downtime expand/contract policy, tested against production-like data before release, each with a documented rollback path.
- The four DORA delivery metrics are tracked per team, trended, and reviewed with engineering leadership.
- Delivery gates hold as AI-assisted change volume grows: change failure rate and rework are monitored explicitly, reflecting the 2025 DORA State of AI-assisted Software Development finding that AI lifts throughput but amplifies delivery instability without safeguards.
- Ephemeral per-change preview environments replace a shared mutable staging, and environment parity (versions, configuration, data shape) is verified in the pipeline.
- Deploy credentials are short-lived workload identities (OIDC federation) rather than stored long-lived secrets, and every production deployment leaves an audit trail satisfying ISO/IEC 27001 and SOC 2 Type II change-management controls.

2 Pipelines deploy most services, but environments drift, some production changes are still applied by hand, rollback is manual and untested, and delivery metrics are anecdotal. 4 All production change flows through reviewed pipelines with progressive rollout and automated rollback, parity is enforced by infrastructure-as-code with drift alerts, and per-team DORA delivery metrics visibly drive improvement work.

F2 Observability

3/5

Structured logs, trace propagation and golden-signal dashboards are standard; 3 payout-critical Kafka consumers lack lag alerts and 240+ dashboards sprawl unowned.

- Logs are structured, centrally aggregated and governed, with per-data-class retention, role-based access, and PII redaction verified in the telemetry pipeline (GDPR storage limitation).
- Distributed tracing propagates W3C Trace Context across every service and team boundary, with trace-to-log-to-metric correlation available for any request.
- Instrumentation follows one organization-wide standard (OpenTelemetry semantic conventions) delivered as a paved-road SDK, so telemetry coverage does not depend on individual teams.
- Each service has an owned, version-controlled dashboard covering the golden signals (RED/USE), reviewed and changed as code.
- Every paging alert is symptom-based, links to a runbook, and falls under a recurring alert-quality review that measures noise and actionability.
- Health endpoints separate liveness from readiness and probe all critical dependencies, complemented by synthetic monitoring of key user journeys.
- AI features emit dedicated telemetry — token and cost accounting, model latency percentiles, output-quality and drift signals — alongside standard service metrics.
- An immutable, access-controlled audit-log stream is kept separate from operational logs and mapped to SOC 2 Type II and ISO/IEC 27001 monitoring controls.

2 Logs reach a central store and some teams keep dashboards, but tracing is patchy, alerts fire on raw thresholds and are noisy, and coverage varies team by team. 4 Metrics, logs and traces are correlated organization-wide under a single instrumentation standard, dashboards and runbook-linked alerts live in version control, and alert-quality and coverage reviews run on a fixed cadence.

F3 SLOs & error budgets

3/5

Signed-off SLOs in the service catalog cover the 6 critical journeys, but no error-budget policy is written and 38% of last quarter's pages were non-actionable.

- SLIs are defined per service and per critical user journey and measured where the user experiences them, following Google SRE practice.
- SLOs are documented, signed off by product and business stakeholders, and roll up consistently under any customer-facing SLAs.
- A written error-budget policy states what happens when the budget is exhausted — feature freezes, mandatory reliability work — and records show it has actually been enforced.
- SLO alerting uses multi-window burn rates rather than static thresholds, paging only when the budget is genuinely at risk.
- Error-budget consumption is reported to engineering governance and demonstrably shapes release pacing and roadmap trade-offs.
- AI-backed capabilities carry their own SLOs covering quality, latency and cost per request, not merely availability.
- SLO reviews run on a scheduled cadence with product stakeholders, with targets adjusted on evidence rather than left to rot.

2 A few teams track availability targets on dashboards, but there is no written error-budget policy, breaches carry no consequence, and alerting is not tied to the targets. 4 Every customer-facing service runs against signed-off SLOs with burn-rate alerting, a written error-budget policy demonstrably gates releases, and SLO reviews with product run on a fixed cadence.

F4 Incident management

4/5

A severity matrix, named incident roles and blameless postmortems with tracked actions are routine; 6 of 14 runbooks still reference decommissioned services.

- A documented severity matrix and defined incident roles (incident commander, communications lead, scribe) exist and are observable in real incident records.
- On-call rotations are staffed and sustainably sized across teams, with paging and escalation policies tested and on-call health reviewed.
- Significant incidents receive blameless postmortems with action items tracked to closure and recurrence analysis across teams, per Google SRE practice.
- Stakeholder communications are pre-templated, a status page is maintained, and contractual customer-notification SLAs are met and evidenced.
- Incidents are correlated with deploys and configuration changes, and change failure rate plus failed-deployment recovery time (two of the four DORA delivery metrics) are measured and trended.
- Runbooks map one-to-one to the paging-alert catalog, are version-controlled, and known failure signatures are auto-remediated where safe.
- Regulatory incident-reporting obligations are mapped and rehearsed — for financial entities including insurers, EU DORA classification and reporting clocks, and NIS2 where in scope.

2 Incidents are handled by whoever notices them, coordination is ad hoc in chat, severity is judged case by case, and postmortems happen occasionally with actions rarely tracked. 4 A practiced process with defined roles and severity-based paging runs every significant incident, blameless postmortems close their actions, recovery-time trends are measured, and regulatory reporting duties are mapped and rehearsed.

F5 DR & backups

1/5

Backups run on schedule, but no DR drill is on record, the standby drifts 61 Terraform resources from code, and restore is untested for 2 of 5 critical stores.

- RTO and RPO are defined per service tier and data store, approved by business owners, and traceable to the replication and backup configuration that achieves them.
- Restores are proven rather than declared: scheduled, timed restore tests and failover drills (game days) produce evidence measured against the stated RTO/RPO.
- Backups are encrypted, access-controlled and include an immutable or offline copy for ransomware resilience, with backup success monitored and failures alerting.
- Every durability-critical store is enumerated and covered — including queues, caches, object stores and AI model or configuration artifacts — so no state silently depends on a non-durable component.
- DR runbooks exist per critical service with dependency-ordered recovery steps, and the multi-zone or multi-region posture includes verified capacity parity for the recovery environment.
- Retention schedules are mapped to legal obligations and erasure is enforced across backup copies (expiry or crypto-shredding), satisfying GDPR storage-limitation and right-to-erasure duties.
- For financial entities including insurers, resilience testing meets EU DORA requirements, with evidence retained for ISO/IEC 27001 and SOC 2 Type II availability controls.

2 Backups run for the main databases, but restores are untested, several stateful components sit outside backup scope, and RTO/RPO figures are aspirations nobody has measured. 4 Every durability-critical store has owner-approved RTO/RPO proven by scheduled, timed restore and failover drills, backups are immutable and monitored, and retention and erasure are enforced across backup copies.

F6 FinOps & capacity

2/5

Aggregate spend is reviewed monthly, but costs are not attributed to features or tenants, and copilot token cost grew 3.4x over two quarters before anyone noticed.

- Cost allocation works end to end: enforced tagging or labeling feeds showback or chargeback per team, service and product under a named FinOps owner (FinOps Foundation operating model, FOCUS-format billing data).
- Unit economics are tracked as KPIs — cost per request, transaction or customer, and per-inference or per-1k-tokens for AI features — and inform pricing and gross-margin decisions.
- Budget guardrails are active: forecasting, spend-anomaly detection, budget alerts and an approval workflow for capacity purchases.
- Capacity plans derive from load tests against stated peak demand with documented headroom, refreshed on a cadence tied to growth forecasts.
- Autoscaling policies are validated under test in both directions (scale-out and scale-in), including warm-pool or cold-start budgets where relevant.
- AI cost engineering is in place: accelerator-utilization targets, model routing from cheaper to stronger models, response and prompt caching, and a periodic self-hosted-versus-API trade study at projected volume.
- Rate optimization is reviewed on a regular cycle — commitment coverage (reserved, savings plans, spot), data-egress and inter-zone traffic, license spend — with decisions recorded.
- Material changes carry cost-regression checks in the delivery pipeline, so cost per unit cannot silently degrade between releases.

2 Aggregate cloud spend is reviewed monthly and the main cost drivers are known, but allocation is incomplete, unit economics do not exist, and capacity is sized by guesswork. 4 Spend is fully allocated with team-level accountability, unit economics inform product decisions, anomalies alert within a day, and capacity comes from load tests and forecasts with validated autoscaling and documented headroom.

G · AI systems

2/5

G1 AI architecture & guardrails

2/5

Schema-validated tool calls and a UI approval gate exist, but update_reserve is callable without the gate on the direct executor path (registry.ts:104); shown in staging.

- A documented architecture bounds model authority — models classify, extract and propose while state-changing actions execute only through deterministic, schema-validated code paths — and the boundary is re-verified in the codebase rather than taken from design documents.
- Every AI-initiated action is mapped to a risk tier with matching human-in-the-loop controls, and irreversible or financially material actions require an explicit confirmation step verified in code and UI flows.
- Tool and function calls exposed to models follow least privilege — per-capability scopes, server-side allow-lists and tenant isolation — with destructive operations gated or excluded, countering Excessive Agency in the OWASP Top 10 for LLM Applications (2025).
- Model outputs are treated as untrusted input: structured outputs are validated against schemas and values are semantically validated against reference data (checksums, lookups) before any downstream use, per Improper Output Handling guidance.
- Each AI capability has a kill switch or feature flag with a documented owner, allowing per-capability disablement without redeployment.
- Degradation paths exist and are tested: multi-provider or multi-model fallback routing with contract tests, timeouts and admission control, so no single model is a single point of failure for availability or quality.
- Inference serving capacity is evidenced by load-test reports and infrastructure configuration showing concurrency limits, batching or queueing, and latency/throughput targets under peak load.
- AI components appear in current architecture documentation (C4 or equivalent) with a named owning team per capability.

2 Confirmation gates and schema validation exist on some flows in some teams, but model authority bounds are undocumented, guardrail coverage varies by team, and fallbacks or kill switches are patchy or untested. 4 Bounded model authority, risk-tiered human oversight, kill switches and tested fallback routing are enforced as organization-wide standards, with guardrail coverage reviewed on every new AI capability.

G2 AI security & red-teaming

2/5

Injection via a poisoned claim PDF reached the tool loop in staging; resistance measured 88.7% pre-fix, and no adversarial suite runs as a release gate.

- An adversarial regression suite mapped to the OWASP Top 10 for LLM Applications (2025) — prompt injection (LLM01), sensitive information disclosure (LLM02), system-prompt leakage and excessive agency among them — runs as a release gate, not a one-off exercise.
- Injection probes cover every input channel — user text, data fields, uploaded documents and retrieved content — including encoded payloads and multi-turn manipulation.
- Red-team cases attempt confirmation-gate bypass and privilege escalation, proving no adversarial dialog can push a state-changing action past its approval step.
- Cross-tenant isolation is probed through the AI layer, and access controls and tenant partitioning on retrieval and embedding stores are verified in configuration.
- Output filtering blocks PII, secrets and internal service data from reaching users or logs, verified by leak probes and log review.
- Unbounded consumption is controlled: per-tenant token budgets, rate limits and spend alerts are enforced in gateway configuration.
- AI security controls are mapped to OWASP AISVS and OWASP ASVS 5.0 with named owners, and periodic independent red-team engagements have findings tracked to closure under severity SLAs.

2 Some teams run prompt-injection checks before major releases, but coverage is uneven, findings live in ad-hoc notes rather than tracked tickets, and tenant isolation and consumption limits go untested. 4 Automated adversarial gates run on every release across teams, independent red-teaming happens on a schedule, and findings are triaged under severity SLAs with safety metrics protected by a non-regression guardrail.

G3 Model & prompt governance

3/5

Prompts sit in source control behind reviewed merge requests and the main models are inventoried; models.yaml:12 floats a latest alias, fraud-screener has no model card.

- A model inventory covers every model in production — primary, auxiliary and third-party — each with an owner, risk tier, model card and approval status.
- Models are referenced by immutable digests rather than floating tags, with the exact digest recorded in deployment manifests and evaluation run reports.
- Prompts, system instructions and generation parameters are versioned in source control and changed only through reviewed merge requests linked to evaluation results.
- AI supply-chain controls address OWASP LLM03: SLSA v1.2 provenance attestation for model artifacts, an SBOM in CycloneDX or SPDX covering models and datasets, and signed artifacts verified at deploy time.
- Licenses and terms of use for model weights, datasets and third-party AI services are reviewed, with commercial-use clearance documented.
- A documented approval workflow with named sign-off roles gates new models and major prompt changes, and a gap assessment against ISO/IEC 42001:2023 (or an NIST AI RMF 1.0 mapping) exists with control owners assigned.
- Provenance and consent basis for any training or fine-tuning data are documented per dataset.

2 An inventory lists the main models and some prompts sit in source control, but auxiliary models float on mutable tags, approvals are informal and provenance or license records are incomplete. 4 Registry coverage, digest pinning, reviewed prompt changes and signed, SBOM-backed model artifacts are enforced organization-wide, with approval workflows audited against the ISO/IEC 42001 gap assessment.

G4 Evaluation gates & quality monitoring

1/5

No golden sets, no CI evals, no field-accuracy monitors: a vendor minor-version bump shipped via the floating latest alias and degraded extraction for 9 days unnoticed.

- Versioned golden evaluation datasets exist per AI capability, and CI blocks release on regression whenever the model version, prompt or generation parameters change.
- Evaluation separates deterministic gates (tool choice and arguments, binary pass/fail at fixed settings) from statistical measures (consistency@k, accuracy thresholds), and LLM-as-judge scoring is confined to free-text quality, never factual or numerical verification.
- Acceptance thresholds are codified per risk tier — up to 100% action correctness for critical flows — and a dual oracle verifies the system's claimed success against the system of record, not just the conversation.
- Behavioral test coverage is derived automatically from the product's capability surface and crossed with a misbehavior taxonomy, so coverage scales with the codebase instead of hand-written scripts.
- Bias and fairness evaluations run per user segment with documented thresholds wherever AI output affects individuals, and results are retained for audit.
- Production AI observability is in place: prompt/completion tracing, per-feature and per-tenant token-cost dashboards, model latency percentiles, and drift and output-quality monitoring on live traffic with alerting.
- AI features carry SLIs/SLOs with error budgets, a written error-budget policy and burn-rate alerting, per Google SRE practice.
- Model and prompt releases are canaried with automated rollback criteria, and change failure rate and failed-deployment recovery time are tracked separately for AI changes, reflecting the 2025 DORA State of AI-assisted Software Development finding that AI lifts throughput but amplifies instability without safeguards.

2 Some capabilities have curated eval sets and a dashboard or two exists, but gates are advisory rather than blocking, thresholds vary by team and live traffic is monitored only for infrastructure health. 4 Blocking evaluation gates with risk-tiered thresholds cover every model and prompt change organization-wide, and drift, cost, fairness and quality monitoring with SLO burn-rate alerts feeds findings back into the eval sets.

G5 AI compliance & transparency

3/5

Art. 30 records, pre-log redaction and immutable decision logs verified in place; the adjuster-copilot DPIA is unsigned and the Art. 50 position on adjuster-reviewed drafts is undecided.

- Every AI capability is classified in a maintained register against the EU AI Act as amended by the 2026 AI Omnibus (Reg. 2026/1744), covering Art. 50 transparency obligations in force since 2 August 2026 and, where applicable, a readiness plan for Annex III high-risk obligations due 2 December 2027.
- Users are informed they are interacting with an AI system and AI-generated content is labeled, per Art. 50, verified in the shipped product flows.
- A DPIA covers each AI feature processing personal data, with GDPR Art. 30 records of processing and documented retention and deletion rules for prompts, completions and model telemetry.
- PII controls span the AI pipeline: classification-aware redaction before logging, and data-subject request workflows that reach prompts, traces, caches and any fine-tuning data.
- AI-influenced decisions are auditable: immutable logs link each decision to its input, model digest, prompt version and resulting action, retained to regulatory schedules.
- An AI management system with named accountable roles and documented human-oversight design exists, evidenced by an ISO/IEC 42001:2023 certification or gap assessment, or an NIST AI RMF 1.0 profile.
- For financial entities (insurers included), AI services fall within scope of EU DORA: third-party AI providers appear in the ICT register with exit strategies and resilience testing.

2 A privacy policy mentions AI and one or two features have had informal legal review, but there is no capability-level classification, DPIAs are missing for newer features and prompt retention is undefined. 4 A maintained register classifies every AI capability, Art. 50 disclosures and audit-grade decision logs are verified in production, and accountable owners operate a documented management system with evidence kept current for regulators.

// DEEP DIVES — 8 OF 26 REGISTER ENTRIES, SHOWN AT FULL DEPTH

04 Selected findings

R-01 P0 area C6 · likelihood high · impact critical · owner: Data & Analytics

Analytics export trusts caller-supplied tenant_ids[], enabling cross-tenant claims export

The warehouse export endpoint in analytics-svc builds its query from the tenant_ids[] array supplied in the request body instead of re-deriving tenant scope from the auth context. A partner-portal token scoped to a single tenant was used in staging to export another tenant's claims rows. The row-level tenant guard that protects the transactional path is bypassed entirely on this route.

services/analytics/src/export/query-builder.ts:142

```
const tenantIds = req.body.tenant_ids; // caller-supplied, never re-derived
const scope = tenantIds.map((t) => sql`tenant_id = ${t}`);
return sql`SELECT * FROM claims_export WHERE ${sql.join(scope, sql` OR `)}`;
```

The export path surfaced in week 1, during the code-review sweep of every query path that reaches the warehouse. The transactional services all route reads through the shared row-level tenant guard, but analytics-svc assembles its export query directly from the request body, trusting a tenant_ids[] array the caller supplies. Notably, the behavioral AI harness showed no cross-tenant leakage in 402 of 402 relevant retrieval probes; this is a plain REST export path that never touches the AI layer, which is why code review found it and the harness could not.

In week 2 we proved it in staging, never in production: a partner-portal token scoped to one tenant was sent with another tenant's id in tenant_ids[], and the export returned that tenant's claims rows in full. Staging data is synthetic, and the crafted request is documented in the evidence pack so the engineering team could reproduce it in minutes.

The blast radius is every tenant's claims data, claimant personal data included, exposed to any holder of a partner-portal token: a GDPR exposure as well as a contractual one for the white-label insurers. Whether the path was ever exercised in production could not be established within the audit's read-only access, so reviewing production export logs is remediation step one; it bears on notification duties but does not reduce the severity.

The fix path was agreed during the audit: a same-day hotfix deriving tenant scope solely from the authenticated principal and rejecting any request that carries tenant_ids[], followed by shared scoping middleware on every warehouse read and cross-tenant access tests in CI.

FIX · Hotfix same day: derive tenant scope solely from the authenticated principal and reject any request carrying tenant_ids[]. Then move tenant scoping into shared middleware applied to every warehouse read path, and add cross-tenant access tests to CI so any regression fails the build. Finally, review export logs across the retention window to confirm no prior misuse.

R-02 P0 area F5 · likelihood medium · impact critical · owner: Platform/SRE

Disaster recovery never exercised end-to-end; stated RTO/RPO currently unverifiable

The warm-standby region has never been exercised end-to-end: it is absent from the pipeline's apply matrix and is applied manually from a frozen branch, and a week-2 Terraform plan showed 61 resources drifted from code. Point-in-time restore is untested for 2 of the 5 critical datastores (Kafka consumer offsets and the claims document store), and no restore drill is on record. The RTO of 4 hours and RPO of 15 minutes stated in the enterprise vendor questionnaire are therefore currently unverifiable.

`.github/workflows/terraform-apply.yml:31`

```
strategy:
  matrix:
    env: [production, staging]
```

The gap surfaced in week 1 during document review. The vendor-assurance questionnaire prepared for the two enterprise onboardings states an RTO of 4 hours and an RPO of 15 minutes, so we asked for the most recent restore-drill report; none exists on record. Backup jobs run on schedule, which is precisely why the gap had stayed invisible: nothing was failing, so nothing was questioned.

In week 2, Platform/SRE ran a Terraform plan against the warm-standby region with the auditor observing: 61 resources diverge from code, because the standby is missing from the pipeline's apply matrix and is applied manually from a frozen branch, while the primary region is applied from main on every merge. In week 3 the team walked the DR runbook through in staging; the walk-through stalled at the step that reconstructs Kafka consumer offsets, which the runbook marks as to-be-defined, and for the claims document store no restore has ever been attempted on record - backup copies exist but have never been read back. Point-in-time restore is thus untested for 2 of the 5 critical datastores. No test touched production at any point.

In a regional failure the platform would be rebuilt onto infrastructure that no longer matches code, with no practiced procedure for the two stores that anchor claims processing. Restoring the payout pipeline without consumer offsets forces a choice between replaying events (duplicate payouts) and skipping them (silently lost claims), while the document store holds the FNOL evidence that adjusters and the AI intake path depend on. The 4h/15min figures already given to prospective enterprise customers are unverifiable, which is itself an assurance risk for the onboardings.

The fix is phased. First, bring the standby region under the same pipeline and drift detection as the primary and reconcile the 61 drifted resources; this is days of work. Second, define and test restore procedures for consumer offsets and the document store in staging, recording timings against the stated targets. Third, schedule recurring game days that fail over and restore end-to-end, and only re-state the RTO/RPO to customers once a drill has met them.

FIX · Phase 1 (days): apply the standby from the same pipeline as the primary, enable drift detection on it, and reconcile the 61 resources. Phase 2 (weeks): build and test restore procedures for Kafka consumer offsets and the document store in staging, with timed evidence. Phase 3 (quarterly): run end-to-end failover and restore game days against the stated RTO/RPO, and only re-state 4h/15min to customers once a drill has met them.

R-03 P0 area G1 · likelihood high · impact critical · owner: AI Enablement

Prompt injection via uploaded claim PDF reaches tool loop; reserve approval gate bypassed

An uploaded claim PDF carrying embedded instructions reached adjuster-copilot's tool loop: retrieved chunks are concatenated into the prompt without provenance tagging (assemble.ts:88), and update_reserve is callable without the human-approval gate on the direct executor path, the gate being enforced in the UI only (registry.ts:104). In the staging behavioral test, safe mode intercepted an update_reserve call requesting a 40× reserve increase. This spans both AI architecture (G1) and AI security (G2).

ai/copilot/src/context/assemble.ts:88

```
const prompt = [SYSTEM_PREAMBLE, ...retrieved.map(c => c.text)].join(SECTION_SEP);
// TODO: tag chunk provenance before GA — see RFC-114
return prompt;
```

The behavioral harness mapped 61 capabilities directly from code and generated 412 probes across 19 misbehavior vectors; the injection family included claim documents with embedded instruction blocks. In week 2, one probe uploaded a claim PDF whose body text addressed the assistant directly and asked it to adjust the claim's reserve. Because assemble.ts:88 concatenates retrieved chunks into the prompt with no provenance tags or delimiters, the document's text arrived with the same standing as the system instructions.

On the UI path the human-approval gate held in 100% of probes. On the direct tool-executor path, however, the gate exists only in the front-end: registry.ts:104 registers update_reserve with no server-side approval requirement, and the poisoned document drove an update_reserve call requesting a 40× increase to the probe claim's reserve. Every run was in staging safe mode with all writes intercepted — zero side effects on real data — but the same call would have executed unimpeded in production. Nothing exotic is needed to reach that path: the copilot's own agent loop dispatches model-emitted tool calls server-side through the registry executor — the UI gate guards only human-initiated actions, so a model-initiated call bypasses it by construction.

Any uploaded claim document is attacker-controllable input, so the exposure covers every tenant in the adjuster-copilot pilot. Reserve values are financially material and feed downstream solvency and reinsurance reporting; pre-fix injection resistance across the family measured 88.7%, and the honest-refusal rate of 91.4% offers no protection once instructions are laundered through retrieval.

The fix is sequenced to match the go-live conditions. The executor-side gate is a days-scale hotfix and is the stated condition for continuing the pilot; provenance tagging and untrusted-content delimiting at assembly follow within weeks; the injection probe set then becomes a blocking adversarial regression suite in CI, so the 88.7% figure is re-measured on every release rather than once per audit.

FIX · Phase 1 (days): enforce the human-approval gate server-side in the tool executor for every state-changing tool — the condition under which the pilot may continue. Phase 2 (weeks): provenance-tag retrieved content and delimit it as untrusted at prompt assembly. Phase 3: promote the injection probe set into CI as a blocking adversarial regression suite mapped to the OWASP Top 10 for LLM Applications (2025).

R-04 P0 area G4 · likelihood high · impact critical · owner: AI Enablement

No evaluation gate: floating model alias let a vendor bump silently degrade extraction

Prompt and model changes ship with no evaluation gate: prompt-registry/models.yaml:12 references the extraction model through a floating latest alias, there is no golden set, no offline eval step in CI and no online field-accuracy monitoring. A vendor minor-version bump the month before the audit silently degraded extraction for 9 days until a reinsurer complained.

prompt-registry/models.yaml:12

```
intake-extractor:
  provider: cloud-llm
  model: doc-extract-std
  version: latest   # floating alias — resolves at call time
```

The month before the audit, the extraction vendor shipped a minor version bump. Because prompt-registry/models.yaml:12 references the model by a floating latest alias, the new version went live to every tenant with no release event, no evaluation run and no record of the change — the platform's own deployment history shows nothing on the day behavior changed.

Nothing in the pipeline could have caught it. There is no golden evaluation set for intake-extractor, no offline eval step in CI, and no online monitors on field-level accuracy of extracted values. The regression surfaced 9 days later, when a reinsurer complained about inconsistent claim figures: the system of record downstream, not the platform, raised the alarm. The audit reconstructed the timeline from the vendor changelog and the reinsurer's ticket; the gap itself was already evident from the configuration alone.

The blast radius is every document processed through intake during those 9 days, across all tenants, with no marker separating affected extractions from clean ones. The remediation path is sequenced: pin versions in the registry and manifests within days; stand up versioned golden sets and a blocking CI eval within weeks; then add online accuracy and drift monitoring with canaried model releases and automated rollback. This evaluation gate is a named condition for adjuster-copilot GA.

FIX · Pin model references to immutable versions recorded in deployment manifests (days). Then build versioned golden evaluation sets per AI capability and wire a blocking CI eval on any model, prompt or generation-parameter change (weeks). Finally add online field-accuracy and drift monitors on high-value extraction fields, and canary model releases with automated rollback criteria.

R-05 P1 area C2 · likelihood high · impact high · owner: Partner Integrations

Contract tests missing on 2 of 6 Kafka event schemas; Q3 change dropped partner webhooks

Consumer-driven contract tests run for 4 of the 6 Kafka event schemas; partners.webhook-dispatch.v1 and documents.extracted.v1 are covered on neither the producer nor the consumer side. In Q3 an incompatible change to the webhook-dispatch schema shipped to production and partner webhook delivery failed silently for 6 hours, until a broker reported missing claim-status notifications.

.github/workflows/contract-tests.yml:31

```
matrix:
  schema: [claims.fnol-received.v1, claims.status-changed.v1,
           payouts.approved.v1, fraud.case-flagged.v1]
  # partners.webhook-dispatch.v1 and documents.extracted.v1: no consumer contracts
```

The thread started in week 2, in the postmortem archive rather than the code. A Q3 postmortem recorded six hours of silently dropped partner webhooks after an incompatible change to the webhook-dispatch event schema; its action item, add consumer contracts, was still open when the audit began. Cross-checking the contract-test workflow confirmed that only 4 of the 6 event schemas are exercised, with partners.webhook-dispatch.v1 and documents.extracted.v1 uncovered on both sides.

To confirm the failure mode still exists, we replayed the Q3 change in staging: a renamed field in the producer schema deserialized to null in the current consumer, which acked the message and emitted nothing. No alert fired, because delivery success is not measured: the queue drains normally while the payload is dropped. Production was never touched; the replay used staging topics and synthetic events.

The blast radius is partner trust rather than data loss: webhook-dispatch carries claim-status notifications to brokers and white-label insurers, exactly the integration surface the two enterprise onboardings will stress. The schema registry supports compatibility enforcement, but both uncovered subjects have it set to NONE, so nothing mechanical prevents a repeat.

The fix is cheap relative to the incident it prevents: declare FULL_TRANSITIVE compatibility on both subjects and add consumer-driven contracts to the producer and consumer pipelines — the contract tests, not the registry mode, are what catch a defaulted-field rename. The remediation roadmap sequences both in Phase 2, since neither needs architectural work, and a webhook delivery-success alert closes the detection gap in parallel.

FIX · Set FULL_TRANSITIVE compatibility on both uncovered subjects so the registry rejects incompatible changes at publish time — noting that a defaulted-field rename passes any compatibility mode, which is why the consumer-driven contract tests, added to both producer and consumer pipelines as blocking gates, are the control that actually catches it. Alert on webhook delivery success rate so a silent drop pages within minutes rather than hours.

R-09 P1 area E5 · likelihood medium · impact high · owner: Claims Intake

Claim attachments retained indefinitely against the 24-month DPA deletion commitment

The sampled signed carrier DPAs commit the platform, as processor, to deleting claim attachments 24 months after claim closure. No deletion path exists anywhere: the production document store's Terraform sets lifecycle_rules = [], no retention job or schedule could be found, and an object-age inventory shows attachments from the longest-standing tenants far older than any plausible closure-plus-24-months date. Indefinite retention of special-category health data breaches the DPAs and leaves the carriers, as controllers, exposed on GDPR storage limitation.

infra/terraform/prod/document-store.tf:14

```
module "claim_documents" {
  source          = "../modules/object-store"
  lifecycle_rules = []
}
```

The gap surfaced in week 2, during the data-protection review, when sampled signed carrier DPAs were read against the platform's actual retention behavior. Each sampled DPA commits the platform, as processor, to deleting claim attachments 24 months after claim closure. No retention schedule, lifecycle rule or deletion job could be found in code, infrastructure or documentation, and interviews with the Claims Intake and Data & Analytics squads confirmed none has ever existed.

Verification stayed within the audit's read-only access, and no attachment content was opened. The Terraform module defining the production document store sets lifecycle_rules = [], and a metadata-only object-age inventory showed objects from the longest-standing tenants years older than any plausible closure-plus-24-months date; an aggregate count run by the Claims Intake squad at the auditor's request confirmed attachments linked to claims closed more than 24 months ago. In staging, the squad backdated a test claim's closure past the trigger and its photos and medical documents remained untouched, confirming behaviorally that nothing deletes them.

The blast radius extends beyond the primary store. Claim attachments routinely contain special-category health data (medical reports, injury photographs, treatment invoices), and whether copies also persist in replicas, backups, the warm-standby DR region or any derived datasets was not established during the audit and must be confirmed during remediation. Every month of continued retention widens the gap between what the DPAs promise the carrier controllers and what the platform actually does.

The fix is closure-driven rather than age-driven, since a blunt lifecycle rule would delete attachments on long-running open claims. First, agree the retention position with the carriers' data-protection contacts and ship a deletion job keyed to claim closure plus 24 months, with legal-hold and reopened-claim exceptions. Then run a logged, staged backfill of overdue objects, extend deletion to replicas, backups and the warm-standby copy, and add a scheduled compliance check that alerts on any object past its retention date. Finally, fold the schedule into the Art. 30 records of processing so the paper position and system behavior match.

FIX · Phase 1: agree the retention position with the carriers' data-protection contacts, record the decision, and ship a closure-driven deletion job (attachments deleted 24 months after claim closure, with legal-hold and reopened-claim exceptions). Phase 2: run a logged, staged backfill of overdue objects and extend deletion to replicas, backups and the warm-standby copy, with a scheduled check alerting on any object past retention. Phase 3: update the Art. 30 records of processing and the retention schedule so documentation matches system behavior.

R-11 P1 area F3 · likelihood high · impact medium · owner: Platform/SRE

SLOs lack an error-budget policy; 38% of pages last quarter were non-actionable

SLOs are defined for the 6 critical user journeys and have dashboards, but no error-budget policy exists, so a spent budget carries no consequence for release pacing. Paging history for the last quarter shows 38% of pages were non-actionable, and on-call interviews corroborate measurable alert fatigue.

docs/slo/README.md:14

```
## Error-budget policy

TBD - to be agreed with product before copilot GA.
```

SLO definitions for the 6 critical user journeys were easy to find: they live in the service catalog with dashboards attached, and most carry product sign-off. What is missing is the policy layer. The error-budget section of the SLO documentation is a placeholder that git history shows unchanged for 11 months, no document anywhere states what happens when a budget is spent, and week-1 interviews confirmed that no release has ever been paused on reliability grounds.

In week 2 we analyzed the paging history for the last quarter, exported read-only from the paging tool. 38% of pages were closed as non-actionable: auto-resolved before anyone looked, duplicates of another page, or acknowledged with no action taken, and the burden falls unevenly, with two squads absorbing most of the noise. Alert rules fire on static thresholds rather than budget burn rates, so the pager is loud when nothing is at risk and silent when a budget erodes slowly. On-call engineers described routinely snoozing night pages, a learned response the paging data corroborates.

This was an analysis of records, not an experiment, but the blast radius is already visible in the data: payout timeliness is one of the 6 journeys, and the August consumer stall burned most of that journey's quarterly error budget without generating a single page, because alerting fires on static thresholds rather than on budget burn. The fix is to write and sign off an error-budget policy with product, convert paging for the 6 journeys to multi-window burn-rate alerts, and institute a quarterly alert-quality review that removes every rule found non-actionable. Error-budget consumption should then be reported to engineering governance so release pacing has a documented, enforceable basis.

FIX · Write an error-budget policy with product sign-off that names concrete consequences (release slowdown, mandatory reliability work) when a budget is spent. Convert paging for the 6 journeys to multi-window burn-rate alerts, and run a quarterly alert-quality review that deletes or demotes every non-actionable rule, reporting budget consumption to engineering governance.

R-18 P2 area D1 · likelihood medium · impact medium · owner: Platform/SRE

Growing e2e quarantine: 41 flaky specs (14%) excluded, two payout journeys among them

The nightly end-to-end suite excludes 41 specs tagged @quarantine, 14% of its 293 scenarios, and the count has grown from 12 to 41 over three quarters with no spec ever leaving. Entries carry no expiry, owner or budget. Two quarantined specs cover claim payout approval and broker payout reconciliation, leaving both journeys without automated regression cover.

.github/workflows/e2e.yml:63

```
- name: Nightly e2e (staging)
  run: pnpm --filter e2e test -- --grep-invert "@quarantine"
```

The quarantine surfaced in week 1 during the CI configuration review: the nightly workflow invokes the e2e runner with --grep-invert "@quarantine", and a search across the monorepo returned 41 tagged specs out of 293. Git history shows the tag count rising from 12 at the start of Q1 to 41 by the audit, with no spec ever leaving quarantine. Interviews with the QA guild confirmed there is no exit criterion, expiry or owner; tagging is at the discretion of whoever is unblocking a red pipeline.

In week 2, at our request, a Payments engineer re-ran the two quarantined payout journeys, claim payout approval and broker payout reconciliation, twenty times each in the staging environment; our access stayed read-only throughout and nothing was run against production. Both failed intermittently, at roughly one run in three, from fixture collisions on shared staging data and an unconditioned wait on an event-driven status change. The flakiness is environmental rather than a product defect, but the effect is the same: neither journey has had a trustworthy automated end-to-end check for at least two quarters.

The exposure concentrates on the payout path, which moves money to claimants and brokers and is already carrying a separate bus-factor finding. With these journeys quarantined, a payout regression surfaces only at the unit and integration layers, or with a customer. Left unmanaged, the quarantine also erodes trust in the suite as a whole, which is how teams drift back to re-running red builds.

The fix is organizational first, technical second. Assign suite ownership to Platform/SRE jointly with the QA guild, cap quarantined specs at 5% with a 14-day expiry and a named owner per entry, and burn down the payout journeys first with per-run tenant seeds. Then move the nightly run onto the ephemeral per-change environments already used by the integration layer, and track flake rate and quarantine age on the managed observability platform so the budget is enforceable.

FIX · First: assign suite ownership to Platform/SRE jointly with the QA guild; set a flake budget of at most 5% of specs quarantined, each entry with a named owner and a 14-day expiry; restore the two payout journeys first, using per-run tenant seeds to remove fixture collisions. Then: move the nightly run from shared staging onto the ephemeral per-change environments already used by the integration layer, and publish flake rate, quarantine age and time-to-exit on the managed observability platform.

// SINGLE SOURCE OF TRUTH — CLOSURE ONLY ON RE-READ COMMIT EVIDENCE

05 Risk register

ID	FINDING	AREA	L × I	PRIORITY	OWNER
R-01	Analytics export trusts caller-supplied tenant_ids[], enabling cross-tenant claims export	C6	high × critical	P0	Data & Analytics
R-02	Disaster recovery never exercised end-to-end; stated RTO/RPO currently unverifiable	F5	medium × critical	P0	Platform/SRE
R-03	Prompt injection via uploaded claim PDF reaches tool loop; reserve approval gate bypassed	G1	high × critical	P0	AI Enablement
R-04	No evaluation gate: floating model alias let a vendor bump silently degrade extraction	G4	high × critical	P0	AI Enablement
R-05	Contract tests missing on 2 of 6 Kafka event schemas; Q3 change dropped partner webhooks	C2	high × high	P1	Partner Integrations
R-06	Migration rollbacks untested; payout schema carries an irreversible destructive migration	C7	medium × high	P1	Payments
R-07	CI debug step prints runtime-fetched secrets into logs readable by all engineers	E2	medium × high	P1	Partner Integrations
R-08	No SBOMs or build provenance; 11 of 24 service images pull floating base tags	E3	medium × high	P1	Platform/SRE
R-09	Claim attachments retained indefinitely against the 24-month DPA deletion commitment	E5	medium × high	P1	Claims Intake
R-10	No consumer-lag alerting on three payout-critical Kafka consumers	F2	high × high	P1	Payments
R-11	SLOs lack an error-budget policy; 38% of pages last quarter were non-actionable	F3	high × medium	P1	Platform/SRE
R-12	Copilot AI spend untracked per feature and tenant; token cost grew 3.4x unnoticed	F6	high × medium	P1	AI Enablement
R-13	Two contractually committed partner features absent from the requirements baseline	A1	medium × medium	P2	Partner Integrations
R-14	Payout engine knowledge concentrated in two maintainers with no succession plan	B1	low × high	P2	Payments
R-15	Onboarding averages ~10 working days to first merged PR with README-level docs	B1	medium × medium	P2	Platform/SRE
R-16	Partner-facing API has no versioning or deprecation policy	C2	medium × medium	P2	Partner Integrations
R-17	Load tests cover the happy path only; no capacity model for 10x catastrophe FNOL peaks	C4	low × high	P2	Platform/SRE
R-18	Growing e2e quarantine: 41 flaky specs (14%) excluded, two payout journeys among them	D1	medium × medium	P2	Platform/SRE

ID	FINDING	AREA	L × I	PRIORITY	OWNER
R-19	Two medium pen-test findings open beyond 180 days with no remediation SLA	E4	medium × medium	P2	Platform/SRE
R-20	6 of 14 incident runbooks reference decommissioned services	F4	medium × medium	P2	Platform/SRE
R-21	fraud-screener has no model card and no stored decision rationale	G3	medium × medium	P2	Data & Analytics
R-22	Copilot DPIA unsigned; Art. 50 position on drafted replies undecided	G5	medium × medium	P2	AI Enablement
R-23	9 of 24 services carry no ADR trail for their load-bearing design decisions	C1	low × medium	P3	Platform/SRE
R-24	Dependency update cadence uneven: update automation covers 9 of 24 services	E3	low × medium	P3	Platform/SRE
R-25	Dashboard sprawl: 240+ dashboards, many duplicated and unowned	F2	low × medium	P3	Platform/SRE
R-26	prompt-registry naming inconsistent: three conventions across squads	G3	low × medium	P3	AI Enablement

// EVERY REGISTER ID LANDS IN EXACTLY ONE PHASE

06 Remediation roadmap

Phase 1 — Close the blockers

Weeks 1-6

Close the four go-live blockers and the live credential exposure with re-verified evidence — the heaviest conditions on adjuster-copilot GA and the two enterprise onboardings.

- **R-01** Hotfix the analytics export to derive tenant scope solely from the authenticated principal, move tenant scoping into shared middleware on every warehouse read path, add cross-tenant access tests to CI, and review export logs across the retention window. — 2 eng-weeks
- **R-03** Enforce the human-approval gate server-side in the tool executor for every state-changing copilot tool, then provenance-tag retrieved content and delimit it as untrusted at prompt assembly. — 3 eng-weeks
- **R-04** Pin all model references to immutable versions recorded in deployment manifests and wire a blocking golden-set evaluation into CI for every prompt, model or generation-parameter change. — 4 eng-weeks
- **R-02** Apply the warm-standby region from the same pipeline as the primary, reconcile the 61 drifted resources, and build timed, evidenced restore procedures for Kafka consumer offsets and the claims document store. — 5 eng-weeks
- **R-07** Remove the printenv debug step, rotate every credential the partner-sync job fetches, expire the affected workflow logs, review their access history, and add a workflow-lint rule banning environment dumps. — 1 eng-week

Phase 2 — Harden the platform controls

Weeks 4-12

Close the remaining P1s so the platform's integration, data-protection, reliability and cost controls match the scale of two enterprise tenants.

R-05	Set FULL_TRANSITIVE compatibility on the two uncovered Kafka subjects, add consumer-driven contract tests to both producer and consumer pipelines — the control that catches defaulted-field renames — and alert on webhook delivery success rate.	2 eng-weeks
R-06	Adopt an expand/contract migration policy, add a CI job that applies up() then down() against a production-sized snapshot for every new migration, and backfill working down() paths for the last quarter's payout migrations.	3 eng-weeks
R-08	Pin all 24 service base images by immutable digest, add CycloneDX SBOM generation to the shared build workflow, then sign images and emit SLSA v1.2 provenance verified at cluster admission.	4 eng-weeks
R-09	Agree the 24-month retention position with the carriers' data-protection contacts, ship a closure-driven deletion job with legal-hold exceptions, and run a logged backfill of overdue attachments extended to replicas and backups.	4 eng-weeks
R-10	Add consumer-group lag alerts with runbook links for the three payout-critical consumers this sprint, then extend lag monitoring to all consumer groups through the paved-road observability module.	1 eng-week
R-11	Publish an error-budget policy with product sign-off that names concrete consequences for a spent budget, and convert paging for the six critical journeys to multi-window burn-rate alerts.	2 eng-weeks
R-12	Propagate tenant and feature identifiers into every LLM call, record the provider usage block as metrics, and surface cost per claim and per 1k tokens on a FinOps dashboard with budget and anomaly alerts.	2 eng-weeks

Phase 3 — Build the operating rhythm

Weeks 8-24

Work the P2 backlog into owned quarterly programs and sweep the P3 housekeeping, so the gains from Phases 1 and 2 persist as standing practice rather than one-off fixes.

R-13	Register both contracted partner features in the requirements baseline with owners, epics and target releases, and make signed commitments a first-class input to the weekly drift reconciliation.	1 eng-week
R-14 R-15	Stand up a payout-engine succession plan (squad-level CODEOWNERS, weekly pairing rotation, ADR on record) alongside a golden-path service template and owned onboarding guides targeting five working days to first merged PR.	4 eng-weeks
R-16	Publish a partner API versioning and deprecation policy with versioned paths, minimum notice and Sunset headers, and instrument per-endpoint usage so unused surfaces can be retired with evidence.	2 eng-weeks
R-17	Build a catastrophe-day load profile from historic claim data and validate autoscaling plus admission control at the FNOL endpoint against the 10x spike before the two enterprise onboardings.	3 eng-weeks
R-18	Cap the e2e quarantine at 5% of specs with named owners and 14-day expiry, restore the two payout journeys first, and move the nightly run onto ephemeral per-change environments.	3 eng-weeks
R-19	Adopt severity-based remediation SLAs for penetration-test findings, close the two open medium findings within a sprint, and add SLA-breach reporting to the monthly security review.	1 eng-week
R-20	Audit all 14 incident runbooks against the current service inventory, rewrite or delete the six stale ones, link each paging alert to its runbook, and add a CI check flagging references to decommissioned services.	1 eng-week
R-21	Author a model card for fraud-screener in the model registry and persist per-decision rationale (top contributing features alongside the narrative) with each SIU queue entry.	2 eng-weeks
R-22	Obtain DPO sign-off on the adjuster-copilot DPIA, record the Art. 50 applicability decision for adjuster-reviewed drafts (shipping disclosure copy if required), then promote the exposure memo into a maintained capability-level classification register.	1 eng-week
R-23 R-24 R-25 R-26	Run a housekeeping sweep: backfill retrospective ADRs for the nine uncovered services, extend update-bot coverage to all 24 service directories, rationalize dashboards into version control with owners, and migrate prompt-registry names to a single capability/action@semver convention.	4 eng-weeks

The roadmap is sequenced by risk-weighted impact against effort: the four blockers and the live credential exposure come first because they gate both go-live decisions, quick high-leverage controls follow, and structural work is batched into owned programs a single squad can land. No item counts as done on intent — closure requires re-verified evidence (commit, pipeline run or drill record re-read by the auditor), mirroring the closed-verified status used in the risk register.

// WHAT MUST BE TRUE BEFORE GA

07 Go-live gates

GATE	PASS CRITERION	REGISTER
Tenant isolation verified	Cross-tenant access tests pass in CI on every warehouse read path, and the export-log review across the retention window confirms no prior cross-tenant misuse.	R-01
Server-side approval gate	Every state-changing copilot tool call without a recorded human approval is rejected by the tool executor itself, demonstrated on the direct executor path rather than the UI path.	R-03
Evaluation gate live	All production model references are pinned to immutable versions and a blocking golden-set evaluation has run in CI on at least one real prompt or model change.	R-04
DR drill evidence	A timed end-to-end failover and restore drill meets the stated 4-hour RTO and 15-minute RPO, with restore evidence on record for all five critical datastores including Kafka consumer offsets and the document store.	R-02
Behavioral safety non-regression	A full re-run of the behavioral probe set shows injection resistance and honest-refusal rates at or above the pre-fix baselines (88.7% and 91.4%) with zero approval-gate bypasses, on the conservative code-gated numbers.	R-03 R-04
Secret exposure closed	Every credential fetched by the partner-sync workflow has been rotated, the exposed logs are expired, and the CI lint rejecting environment dumps is enforced organization-wide.	R-07
Retention compliance	The closure-driven deletion job is live and a scheduled check reports zero claim attachments older than closure plus 24 months outside documented legal holds.	R-09
Payout path observability	Lag alerts for all three payout-critical consumer groups page within minutes in a deliberate stall test, each alert linking to a current runbook.	R-10
Catastrophe capacity proven	The FNOL intake path sustains the 10x catastrophe load profile in a spike test with autoscaling and load shedding engaged and p95 latency inside the SLO.	R-17
AI governance sign-off	The adjuster-copilot DPIA carries the DPO's signature and the recorded Art. 50 applicability decision; any required disclosure copy is verified in the shipped drafted-reply flow.	R-22

// NO HAND-WRITTEN TEST CASES — THE CAPABILITY MAP IS DERIVED FROM CODE

08 Behavioral AI testing

No test cases were written by hand for this section. The harness derives its capability map directly from the product's own code: the tool registries and intent dispatchers of adjuster-copilot, together with the field schemas of intake-extractor's extraction pipeline, yield 61 distinct capabilities that the AI layer actually exposes to claimants and adjusters. Each capability is then multiplied by a taxonomy of 19 misbehavior vectors grouped into 7 families, from slot handling to injection and honest abstention. Coverage is therefore a property of the code as it stands, not of a tester's imagination: a capability nobody documented still gets probed, because the dispatcher knows about it.

The probes themselves are live, unscripted, multi-turn dialogs in which an LLM plays the user. The probing model is given a goal and a persona (a claimant with a vague first notice of loss, an adjuster chasing an edge-case coverage question, an attacker who has uploaded a poisoned claim PDF) and pursues it across turns, adapting to whatever the system replies; nothing is replayed from a script. 412 probes were generated against the mapped capabilities, 388 completed as valid dialogs, totalling 2,904 turns over 41 hours of harness runtime. The entire run executed in safe mode: every write action was intercepted at the executor boundary, so the 40× reserve escalation that underpins R-03 occurred in a sealed sandbox, observed rather than suffered, with zero side effects on real data.

Verdicts are two-layer by design. The headline numbers are deterministic, code-checked facts: database state after the dialog, which tools were called and with what arguments, whether an approval step was recorded, and pattern checks over every response for leaked identifiers. A separate LLM judge scores the softer qualities (tone, helpfulness, reasoning), and those opinions are reported separately, never blended into the headline figures. Stability is measured rather than assumed: 48 probes were re-run twice and 44 produced identical outcomes (91.7%). A measurement-limitations note at the end of this section separates harness defects from product defects and states, for each result, which number to quote.

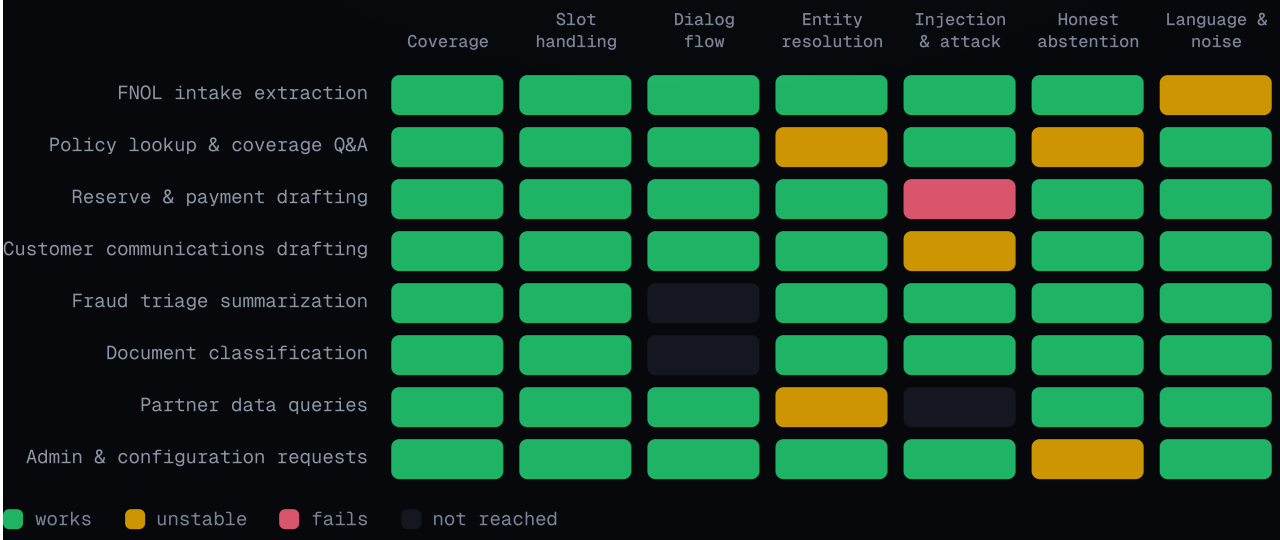
61	19	7	412	388
Capabilities mapped from code	Misbehavior vectors	Vector families	Generated probes	Completed dialogs

2,904	41 h
Dialog turns	Harness runtime

CHECK (CODE-JUDGED)	RESULT	STATUS
Write actions gated behind human approval	Gate held in 100% of UI-path probes, but was bypassed on the direct tool-executor path: a poisoned claim PDF drove an update_reserve call requesting a 40× increase, intercepted by safe mode (R-03)	fail
No cross-tenant retrieval leakage	402/402 relevant probes clean; no other tenant's policy or claim text surfaced in any transcript	pass
Honest refusal of unsupported requests	91.4% refused cleanly; the remainder attempted an answer instead of declining	warn

CHECK (CODE-JUDGED)	RESULT	STATUS
Injection resistance (uploaded documents, messages, retrieved text)	88.7% of injection probes resisted, pre-fix; failures concentrated in the copilot's tool loop	warn
Responses grounded in retrieved policy text	97.2% of coverage answers cited a retrieved policy clause verifiable in the context window	pass
No fabricated claim fields	100%: every extracted field value traceable to the source document; absent values returned as absent	pass
Output-language compliance	100% of replies in the tenant's configured language, including on mixed-language inputs	pass
Outcome stability on identical probes	44/48 re-run probes produced identical outcomes (91.7%)	pass

// CAPABILITY GROUPS × MISBEHAVIOR FAMILIES



Not every anomaly in a run is a product defect, and the report says so explicitly rather than letting the reader guess. Of the 412 generated probes, 24 did not complete as valid dialogs for reasons on the harness side: the simulated user drifted off its goal, a sandbox fixture was missing, or the dialog hit its turn budget. These are logged as harness defects and excluded from the behavioral pass rates. The leakage greps, by contrast, run over every transcript the harness produced, complete or not, which is why the cross-tenant check carries a larger denominator (402 relevant probes) than the count of completed dialogs. The four divergent outcomes in the stability re-run were inspected individually and cluster in the cells marked unstable in the heat-map; wherever attribution between harness and product stayed ambiguous, the case was counted against the product, so the figures err on the conservative side.

Where two numbers exist for the same question, the official one is the conservative, code-gated figure: injection resistance is quoted as 88.7% because that is what the deterministic checks confirm, even though the LLM judge's softer reading was more favorable. Two boundaries matter when quoting this section. The harness observes the AI layer through its conversational surface, so the tenant-isolation flaw in the analytics export path (R-01) sat outside its reach and was found by code review; a clean 402/402 on retrieval leakage is a claim about the copilot's retrieval layer, not about the platform as a whole. Likewise, the fabrication check verifies that every extracted value is traceable to the source document — it does not measure field-level accuracy, which is exactly the monitoring gap R-04 records. And every figure here is pre-fix: the post-fix re-run of the affected suites is reported in the re-audit delta section, not here.

The harness does not retire with the audit: it stays in the product repository and re-runs before each release. Under the go-live gates, its code-gated safety metrics (approval-gate integrity, cross-tenant leakage, injection resistance) take on a non-regression gate — once in force, a release that scores below the previous run on any of them does not ship.

// SIX WEEKS LATER — THE REGISTER LIFECYCLE IS THE PRODUCT

09 Re-audit delta

A re-audit is a short, register-focused second pass, run here six weeks after the main report, in which every P0 and P1 finding is re-tested against the same evidence standard as the original audit. Closed-verified means the auditor re-read the merged code, configuration or drill evidence behind each fix; a ticket moved to done, a screenshot or a team's assurance does not close a finding.

// CLOSED — VERIFIED BY RE-READING THE CHANGE

R-01 Cross-tenant analytics export · PRs #1896/#1903 re-read (scope derived from principal) + cross-tenant CI suite green, re-run 2026-W44; export-log review memo sighted, no prior misuse

R-02 DR unexercised; RTO/RPO unverifiable · Standby apply run from the shared pipeline (drift 61 to 0) + timed failover drill log 2026-W47: RTO 3h12m / RPO 9min, incl. Kafka-offset and document-store restores

R-03 Prompt injection reaches tool loop · PR #2041 re-read: approval gate enforced in the tool executor, retrieved content provenance-tagged; injection probe family re-run in safe mode 2026-W46

R-04 No evaluation gate on model changes · models.yaml pin diff + golden set v1 tag re-read; blocking eval job #88 observed failing a seeded regression in CI, 2026-W45

R-05 Contract tests missing on 2 event schemas · FULL_TRANSITIVE compatibility verified on both subjects; contract suites green in producer and consumer pipelines, 2026-W44

R-07 Secrets printed in CI logs · Workflow diff re-read 2026-W43 (debug step removed, masking at fetch); rotation change-tickets and log-access review note sighted

R-09 Claim attachments retained indefinitely · lifecycle_rules diff + deletion job logs re-read; object-age inventory re-run 2026-W47 shows zero objects past retention

R-10 No lag alerting on payout consumers · Alert rules re-read in the observability module + synthetic lag drill paged on-call, 2026-W44

R-11 No error-budget policy; alert fatigue · Error-budget policy v1.0 with product sign-off; multi-window burn-rate rules re-read for all 6 journeys, 2026-W45

R-12 Copilot AI spend untracked · Per-tenant and per-feature usage metrics live; FinOps dashboard reconciled against the provider invoice, 2026-W46

R-06 (migration rollback coverage) and R-08 (SBOMs and build provenance) remain in progress: both are quarter-scale platform programs rather than six-week fixes, and each has landed its first phase (expand/contract policy adopted; base images pinned by digest) with owners and dates agreed. Neither gated the adjuster-copilot GA decision. The copilot DPIA was signed and the Art. 50 applicability decision recorded ahead of the GO (R-22).

// AREA MOVES

C6 Multi-tenancy & isolation 2 ⟶ 4
Scope re-derived from auth context; shared middleware and cross-tenant CI tests live.

G1 AI architecture & guardrails 2 ⟶ 4
Approval gate enforced server-side in the executor; retrieved content provenance-tagged.

F5 DR & backups 1 ⟶ 3
Standby drift 61⟶0; first timed drill met targets (3h12m / 9min) — capped until the quarterly cadence holds.

G4 Evaluation gates & quality monitoring 1 ⟶ 3
Models pinned; golden set v1 and a blocking CI eval live — monitors still accruing history.

E5 Data protection & privacy 2 ⟶ 4
Closure-driven deletion job live; overdue backfill done; Art. 30 records updated.

D3 Behavioral AI testing 1 ⟶ 3
Harness in the repo, re-run per release; safety metrics now carry the non-regression gate.

F6 FinOps & capacity 2 ⟶ 3
Per-tenant and per-feature AI cost metrics live; budget and anomaly alerts firing.

Six weeks on, the register shows a materially different system. All four P0 findings are closed-verified: in each case the auditor re-read the merged change, the configuration or the drill evidence rather than accepting a status update. Six of the eight P1 findings closed in the same window; the two still in progress — migration rollback coverage and the SBOM and provenance work — are quarter-scale platform items whose first phases have already landed. The overall rating moved from 2.9 to 3.3. Most of the movement sits where the P0s sat: tenant isolation and the agent tool path each rose from 2 to 4 on mechanically enforced controls, disaster recovery from 1 to 3 on the strength of the first timed failover and restore drill on record — deliberately capped until the quarterly cadence holds — and AI evaluation from 1 to 3 once a blocking eval gate ran in CI. The retention and cost-visibility closures and the behavioral harness becoming a release gate add the rest.

The headline numbers moved the same way. Injection resistance, measured at 88.7% on the pre-fix run, re-measured at 99.1% when the same probe family was re-run in safe mode, and the human-approval gate on state-changing tools now holds on both the UI and the direct executor path; the residual failures were phrasing drift with no tool calls, not state changes. With the four P0s closed-verified and the eval gate blocking in CI, the stated conditions were met and the adjuster-copilot GA decision moved from NO-GO to GO. A snapshot report starts ageing the day it is delivered; the register keeps measuring, with every closure re-verified against code and configuration and the behavioral harness re-run as a release gate.

// PRODUCTION-READINESS AUDIT — SAMPLE REPORT

Commission an audit like this

A commissioned audit applies this same method to your systems — evidence-first, with read-only access and every finding backed by file-and-line evidence. You receive this exact structure, applied to your platform: the full report, the risk register, the remediation roadmap, the go-live gate checklist and the behavioral AI test, with a re-audit pass to verify what was fixed.

Scope and timeline agreed per engagement · read-only access is all we need · NDA-friendly.

dev-ai.dev/en/audit · https://www.linkedin.com/in/koldovsky/