Resilient by Design

What I learned building for regulated finance

Lyubomir Bozhinov · Director of Engineering, Tradu

WeAreDevelopers Online 2026

```
SELECT COUNT(*) FROM audit_events
WHERE timestamp >= NOW() - INTERVAL '4 hours';

 count
-------
     0
(1 row)
```

Designed to survive failure.
Not designed to notice it.

Resilience isn't uptime.
It's knowing exactly what happened,
even when things went wrong.

Five claims

1. **Fail hard** when silent corruption costs more than downtime
2. **Compliance is the architecture** — not a feature bolted on
3. **Observability that earns its place** — tier by latency budget
4. **Graceful degradation by severity** — not all circuit breakers behave the same
5. **Test the failure modes** before they test you

> **Thesis:** fail hard, recover deliberately.
> **Spine:** what did the system already know to do, before the bad moment happened?

PILLAR 1

Fail hard, recover deliberately.

"What happened at 14:32:07 on March 3rd?"

— the auditor

The choice

The cluster state has diverged from the database. Do you —

- **Auto-recover.** Pick a winner. Reconcile. Carry on.
- **Stop everything.** Alert an operator. Force investigation.

Every instinct says auto-recover.

The cost of silent corruption dwarfs the cost of downtime.

1.53M

production disks · 41 months · hundreds of thousands of silent corruption events across three failure modes

Source: Bairavasundaram et al., "An Analysis of Data Corruption in the Storage Stack,"
USENIX FAST '08, Feb 2008

~1 / 1,000

silent data corruption rate in Meta's AI accelerator fleet — three orders of magnitude worse than the historical soft-error baseline

Source: Meta Engineering, "How Meta keeps its AI hardware reliable," 22 Jul 2025

$6.2B

JPMorgan London Whale, 2012. A divide-by-sum bug in an Excel-based VaR model kept telling the bank everything was fine.

Sources: US Senate PSI, JPMorgan Chase Whale Trades, 15 Mar 2013 · JPMorgan Management Task Force Report, 16 Jan 2013

If you can't replay it
and get the same answer,
you don't have a system.
You have a hope.

— Martin Thompson, LMAX

Sources: Fowler, "The LMAX Architecture," 12 Jul 2011 · Thompson et al., "Disruptor 1.0," LMAX, May 2011

The lineage we picked

Aeron — Martin Thompson's open-source successor to the LMAX Disruptor, 2014.

- **Deterministic replay** — the property regulators demand
- **Lock-free messaging** — microsecond hot path
- **Tiered observability native** — counters, structured logs, archives

Our core trading engine runs on Aeron.

> We didn't invent these patterns.
> We picked the lineage that already proved them.

Sources: Aeron — github.com/aeron-io/aeron · Aeron Cookbook — aeroncookbook.com

A fail-hard story written by physics

27 Nov 2025 — a chiller plant at CyrusOne's CHI1 data centre (Aurora, IL) failed.

28 Nov 2025 — temperatures inside crossed safe operating ranges. Thermal-protection circuits engaged. Silicon shut everything down.

CME's Globex futures markets went dark for ~10 hours.

> Not graceful degradation.
> A fail-hard story written by physics,
> not by architecture.

The question is whether your system, given the same forcing function, would do something half as sane.

Sources: CME Group, Globex client notices, 27–28 Nov 2025 · CyrusOne, CHI1 cooling-failure statements, 27 / 30 Nov 2025

When to use the other answer

Failing hard costs you availability. Most products want the opposite.

- Content sites, recommendation engines, analytics dashboards → **fail soft, recover fast**
- Money, healthcare, safety-critical control systems → **fail hard, recover deliberately**

> Failing hard preserves evidence.
> The next question is who that evidence is for.

PILLAR 2

Compliance is the architecture.

"Show me a tamper-proof record of every order placed last Tuesday — and prove it hasn't been edited since."

— the auditor

$2B+

cumulative SEC + CFTC recordkeeping penalties since 2021 · 100+ firms · for messages about transactions on personal devices

Sources: SEC Press Release 2022-174 (27 Sep 2022) · CFTC Press Release 8599-22 (27 Sep 2022) · SEC FY2024 Enforcement Statistics (Nov 2024)

Logging is not proving.

Two-phase auditing

Phase	When	On failure
Intent (sync)	before execution	command rejected · HTTP 503
Outcome (async)	after execution	reconcilable from event log

MiFID II **Article 16(7)** requires recording communications "intended to result in" transactions, **before** execution occurs.

For algorithmic trading, **RTS 6** layers on a documented inventory of every algorithm.

Sources: ESMA Single Rulebook, MiFID II Art. 16(7) · Commission Delegated Reg (EU) 2017/589 (RTS 6)

Binary compliance posture

The system is either **100% compliant** or it **stops accepting commands**.

- Audit DB unreachable → 503
- Identity unresolvable → 503
- No degraded mode where "most" of the audit trail is captured

> Back-office downtime over non-compliance.
> In regulated finance, that's the right trade.

Hash-chained audit events

```
event_n = (content, prev_hash)
hash_n  = H(content || prev_hash)
```

Modifying any record breaks **every** subsequent hash.

The difference between "we logged it" and "we can prove it hasn't been tampered with since."

Detection ≠ prevention. Anchor hashes externally at regular intervals.

This isn't a fintech-only problem

18 Jul 2025 — Replit's AI agent deletes a production database during an explicit code freeze.

1,196 companies gone. **1,206 executive records** gone.

The agent, when asked: "panicked" · "catastrophic error in judgment."

Replit's CEO, two days later: "Unacceptable. Should never be possible."
Which is correct. And also: it happened anyway.

> Different domain. Same failure mode.
> If you can't reconstruct the why,
> MiFID II is the least of your problems.

Sources: J. Lemkin, X posts, 18–19 Jul 2025 · A. Masad (Replit CEO), X post, 20 Jul 2025

DORA — the register is the architecture

EU Digital Operational Resilience Act, effective 17 January 2025.

Article 28 requires every financial entity to maintain a live register of every ICT third-party — cloud, SaaS, critical service — classified by criticality, with documented exit strategies.

November 2025: 19 Critical ICT Third-Party Providers designated for direct supervision. **AWS · Azure · Google Cloud · IBM · Bloomberg · TCS.**

> Active enforcement began with the
> March 2026 submission cycle.
> The register is a real-time data model
> your architecture either supports — or doesn't.

Sources: Regulation (EU) 2022/2554 (DORA), Arts. 28–30 · ESAs, First list of CTPPs, 18 Nov 2025

Follow the compliance boundary, not the service boundary

Event sourcing is expensive. Apply it where the regulator asks.

- **Event-sourced:** order execution, trade lifecycle, risk decisions, position changes
- **CRUD:** user profiles, reference data, configuration

GDPR right-to-erasure × immutable log → **crypto-shredding**. Encrypt with a per-subject key; destroy the key on erasure. Data remains, unreadable.

Source: Verraes, "Crypto-Shredding in Event-Sourced Systems"

The architecture makes
non-compliance
structurally difficult.
Evidence has to live somewhere —
and at the latency
the question demands.

PILLAR 3

Observability that earns its place.

"Show me what happened at 14:32:07."

— the auditor

Tier by latency budget, not uniformly

Path	Latency budget	Observability
Hot	microseconds	atomic counters → lock-free ring buffer
Warm	milliseconds	async structured logs, drop on backpressure
Cold	"before the auditor asks"	full traces, batch, years of retention

The hot path never slows down for I/O. The warm path drops log lines before adding latency to a risk check.

> **Aeron implements all three tiers natively.** The substrate already does the tiering.

Source: Aeron — aeroncookbook.com

If your measurement tool introduces more latency than the thing you're measuring, you're observing your observer.

— Gil Tene, "How NOT to Measure Latency"

Source: Tene, Strange Loop 2015 · HdrHistogram GitHub README

~50ns

span creation cost. Fixed binary encoding, pre-allocated attribute arrays. W3C Trace Context compatible. No allocations, no GC pressure.

Source: W3C Trace Context, W3C Recommendation, Nov 2021

The steel-man

For **99% of services**, OpenTelemetry is exactly right.

For services on a microsecond budget, the standard SDK allocates on every span — in a hot path, that's not a rounding error, it's a defect.

> The right answer depends on what your latency budget actually is.
> Most teams have never measured it.

Observe the business, not just the system

CPU, memory, pod health — tells you the system is running. Not that it's correct.

The most important alert in the back-office fires with **zero delay** on risk-limit breaches.

No 30-second window. No 1-minute `for` clause. **Instant.**

> Watching the engine,
> ignoring the road.

PILLAR 4

Graceful degradation by severity.

"Why did your audit pipeline failure shut you down — and your reporting failure didn't?"

— the auditor

Aviation, not switchboard

One engine fails — the plane stays in the air.
Hydraulics fail — it lands.

The response is **proportional** to the criticality of what failed.

A circuit breaker isn't a binary switch. It's a policy decision.

Source: Nygard, Release It! 2nd ed., Pragmatic Bookshelf, 2018

Three databases, three policies

Dependency	Circuit-open behaviour	Kubernetes signal
Audit DB	fail fast → not ready	pull from traffic
Operational DB	sensible defaults	degraded, keep serving
Analytics DS	empty results	healthy

Each circuit breaker's state is wired into the health check. The orchestrator routes on what the app knows about itself.

Source: Resilience4j project documentation

Layered fallbacks make the trade visible

```
Primary (Redis)
     ↓ fails
Fallback (local ConcurrentHashMap, degraded-mode flag)
     ↓ fails
Safe state (reject new work, preserve existing)
```

Health check reports: "Redis unavailable — using local fallback."

> If you can't see the fallback,
> you don't have a fallback.
> You have a guess.

The steel-man

Every fallback is code to write, test, maintain. Every silent fallback is a primary that's been broken for days. Every fallback is a new state.

The discipline isn't "build a fallback for every dependency."

It's "decide which deserve a fallback and which deserve to fail fast."

> **That decision is the architecture.**
> **The code is the consequence.**
> Pre-decided degradation modes are claims — and we don't believe claims, we test them.

PILLAR 5

Test it
or you don't have it.

"When did you last test this?"

— the auditor

The smallest chaos experiment
isn't a tool.
It's a sentence.

Hypothesis-first

> "We believe the service recovers within 30 seconds when the database connection drops."

That's a hypothesis. Test it. Time the recovery.

- Right → confidence earned
- Wrong → gap found before your users find it

You don't need Netflix's budget. You need a hypothesis, a steady-state baseline, and a kill switch.

Source: Rosenthal, Hochstein, Blohowiak, Jones, Basiri, Principles of Chaos Engineering, 2017

Chaos engineering without observability is just chaos.

— Charity Majors, Chaos Conf 2018

Source: Majors, "Closing the Loop on Chaos with Observability," Chaos Conf 2018

Twenty years on, the loop still isn't closed

Charity Majors, January 2026:

> Twenty years of DevOps was a single long fight to close one feedback loop between developers and production — and on those grounds, it failed. Not because engineers didn't try. Because the tools weren't good enough.

That's the bar this talk is trying to clear: **tools, patterns, architecture that earn the loop.**

Source: Majors, "'You Had One Job': Why Twenty Years of DevOps Has Failed to Do It," Honeycomb, 15 Jan 2026

And the regulator is in the room

DORA Article 26 — Threat-Led Penetration Testing.

In force since **8 July 2025**. If you're a significant financial entity in the EU, the regulator now mandates threat-led penetration testing on **live systems**, on a regular cadence.

Chaos engineering is no longer "what mature teams do."
It's "what the law requires."

First full TLPT cycle: **by January 2028.**

Source: Commission Delegated Regulation (EU) 2025/1190, in force 8 Jul 2025

Back to where we started

```
SELECT COUNT(*) FROM audit_events
 WHERE timestamp >= NOW() - INTERVAL '4 hours';
 count
-------
     0
```

We had circuit breakers. We had graceful degradation.

We hadn't tested whether the system would degrade gracefully in places where graceful degradation was the wrong answer.

The fix wasn't more resilience patterns

> "If the audit pipeline stops writing, will we know in under five minutes?"

The answer was no.

The answer is now yes.

The change was small. The discovery was uncomfortable.

Friday afternoon
with a hypothesis and a kill switch
—or—
3am on a Saturday
with neither.

CLOSE

Five questions.
One spine.

Five questions. One spine.

When should the system stop? → fail hard when silent corruption costs more than downtime.

Who is it answering to? → compliance is the architecture.

What evidence should it keep? → observability tiered by the latency of the path it watches.

What should it give up first? → degradation matched to the criticality of what failed.

How do we know any of this works? → test the failure modes before they test you.

> What did the system already know to do, before the bad moment happened?

Shoulders of giants

Michael Nygard — Release It! (2007, 2018 2nd ed.)
Greg Young — CQRS / event sourcing
Martin Thompson — LMAX Disruptor, Aeron
Gil Tene — HdrHistogram, latency
Casey Rosenthal & Nora Jones — Chaos Engineering (2017)
Charity Majors — observability (2018, 2026)

> The patterns are old.
> What's new is the willingness to apply them strictly enough that they survive contact with a regulator.

Resilience isn't the absence of failure.
It's the presence of a plan.

The auditor stops asking
when the answers are
already there.

Sources

Every claim in this talk traces to a primary source.
Full bibliography — URLs, access dates, primary documents —
in the **video description below**.

Long-form essays behind each pillar:

barnett-studios.com/blog

Thank you.

If one of the five claims surprised you,
I'd like to know which one.

Lyubomir Bozhinov
Engineering @ Tradu · Founder @ Barnett Studios

LinkedIn
/in/lyubomir-bozhinov

GitHub
@lyubomir-bozhinov

Blog
barnett-studios.com