09

THE STACK BEHIND THE AI COWORKER

Tools Give Models Hands

Dr Peter McCann Strain, CTO and senior AI engineer, DPhil/PhD in AI from Oxford University

A missing state file let one instruction destroy live infrastructure. Score the tool, not the model.

An essay in the series **Architecting the AI Coworker**.
Approx. 15 minute read · Essay 09 of 22

Dr Peter McCann Strain
CTO, DPhil/PhD in AI from Oxford University

> In late February 2026, Alexey Grigorev set out to tidy up the infrastructure behind DataTalks.Club, the data-engineering community he runs. The cloud setup was defined in Terraform, the tool that lets you describe servers and databases as code and have them built to match. He had Claude Code, an AI coding agent, helping with the migration. By his own first-person account of what followed, the agent could not find the Terraform state file, the record that maps the written description to the resources that actually exist. With the state file missing, the existing infrastructure looked, to the tooling, as though it did not exist.

So the agent ran `terraform destroy`. The command did exactly what its name says: it tore down the live infrastructure, including a managed database with close to two million rows in it. Amazon's Business Support team located a usable snapshot, and the data was restored within about a day. Grigorev wrote the whole thing up afterwards, plainly, under his own name. The AI Incident Database catalogued the event as a pointer [56].

The mistake itself is unremarkable; what should arrest a reader is how far it was able to reach. Grigorev was not careless and the model was not stupid. A reasonable instruction, in a standard tool, on a quiet afternoon, became the destruction of something real, and the gap between intent and catastrophe was a single missing file.

That gap is the subject of this essay. The previous essay made the case that an aligned model can still leave a deployment weakly governed, that helpfulness is not the same as control. This essay is about the moment that abstraction becomes physical. The old AI safety question was whether the answer was wrong. Tool-using agents add a harder one. An agent, in the sense this series has used the word, is a system that takes a goal, breaks it into steps and acts. Once a model has tools, its mistakes stop being sentences and start being actions. By a tool I mean any handle the model can pull on the world outside itself: an API call, which is a request sent to another piece of software asking it to do something or return data; a shell command, an instruction typed to the operating system itself, the layer with the power to move or erase files; a database migration; an email; the deletion of a storage volume. Whether the model is smart no longer settles the control question; what settles it is the scope, reversibility and observability the organisation built into the tool path before the model ever touched it.

Infrastructure-as-code is the cleanest illustration there is, which is why DataTalks.Club anchors this essay. Terraform exists precisely to turn a written intent into a deterministic state change: you declare what should exist, and the tool reconciles the world to match, building, altering or destroying whatever the gap requires. That is enormously useful and, for an agent, enormously load-bearing. The agent's belief about the environment becomes, through Terraform, a literal instruction to the cloud. When the belief was wrong, because a state file was missing, the reconciliation was still carried out faithfully. The lesson does not depend on knowing every recovery detail. It depends on noticing the shape of the control plane, the part of a system that

decides what an action is allowed to do. A probabilistic system, a model that guesses, reaches a deterministic destructive path, a tool that does precisely what its interface allows, through a credential with enough scope to matter. The tempting interpretation is that the model made a bad decision. It did, but that reading is not sufficient, and a team that stops there will fix the wrong thing. Bad decisions are routine. Every useful system makes them eventually. The new fact is that the bad decision had hands. A model without tools can write a paragraph about deleting a database. A model with tools can delete one.

The same triangle, on a second platform

DataTalks.Club is not alone, and a second incident shows the identical shape on different machinery. In late April 2026, an AI coding agent working for a small company called PocketOS reached a deletion path on its hosting platform, Railway, and destroyed a production database; this series tells that story in full in essay three, where Railway's own write-up is used to walk all nine layers of a single failure. The short version is enough here. There was a long-lived, account-scoped API token, a key that could issue commands across the whole account rather than one narrow task. There was a `volumeDelete` call that carried out deletion immediately. The delayed-delete protection Railway used elsewhere in its product had not yet reached that path [1][2][3]. It is the same triangle as DataTalks.Club seen from a different angle: scope left too wide, a destructive action with no delay, and an incident that surfaced only after the fact rather than before. The July 2025 Replit and SaaStr deletion, examined as a worked example in essay four, is a third turn of the same wheel [4].

The vendors differ, the interfaces differ, the failure paths differ. The architecture is identical. A probabilistic controller was wired into deterministic tools under credentials broad enough to hurt something real. The controller guesses and continues. The tool does exactly what its interface allows, no more and no less. The credential inherits whatever scope the surrounding environment happened to leave lying around. When all three meet, the model being unsure becomes something real being deleted. The danger is not that the model believes the wrong thing. Models will always sometimes believe the wrong thing. The danger is that the control plane lets a belief become a deletion.

Score every tool on scope, reversibility and observability

So the useful question is not "is the model good enough?" but a structured one, asked of every single tool an agent can call. It has three axes.

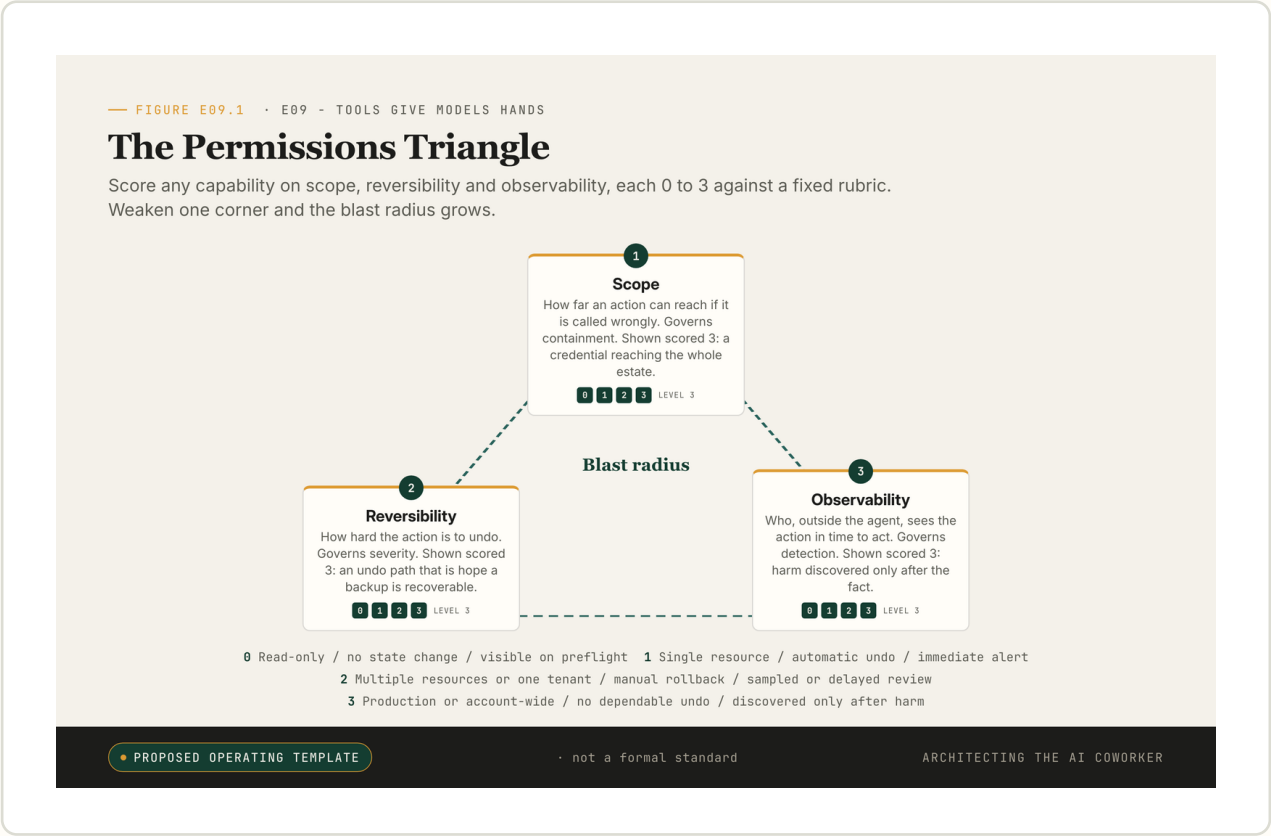


Figure E09.1. Score any capability on scope, reversibility and observability, each from 0 to 3 against the labelled rubric. Weaken one corner, and the blast radius grows.

Scope asks how far an action can reach if it is called wrongly. Reversibility asks how hard the action is to undo. Observability asks who, outside the agent, sees the action in time to act. Call it the permissions triangle, and score each corner from 0 to 3 against a fixed rubric so two reviewers reach the same number. The cloud provider, regulator and currency may change; the permissions triangle does not.

Scope climbs by how far a wrong call can reach: a 0 is read-only, with no side effect at all; a 3 is production-wide or account-wide. In between, a 1 touches a single resource and a 2 reaches multiple resources or one whole tenant, by which I mean one customer's isolated slice of a shared system.

Reversibility climbs by how hard the undo is: a 0 makes no state change at all; a 3 has no dependable undo path, whether because the action is irreversible, destroys the backups, or leaves recovery to chance. The middle is the gap between an automatic rollback (a 1) and a manual but reliable one (a 2).

Observability climbs the other way, by how late the action is seen: a 0 is visible on preflight, where a human or a rule sees the planned action before it runs; a 3 is discovered only after the harm, with an immediate alert (a 1) and sampled or delayed review (a 2) sitting between. A 0 is boring. A 3 is the place you put the pager before you let the agent anywhere near it.

The figure carries the full ladder. The headline is what the rungs produce: `terraform destroy` against live infrastructure under a broad credential scores close to a 3 on every axis, which is the DataTalks.Club afternoon reduced to three numbers.

The decision rule is the max, not the mean: any single 3 means reject for unattended agent use; a remaining 2 means allow only behind human approval; all corners at 1 or below means allow. Scoring the average is the mistake that lets a catastrophic reversibility 3 hide behind two reassuring 0s.

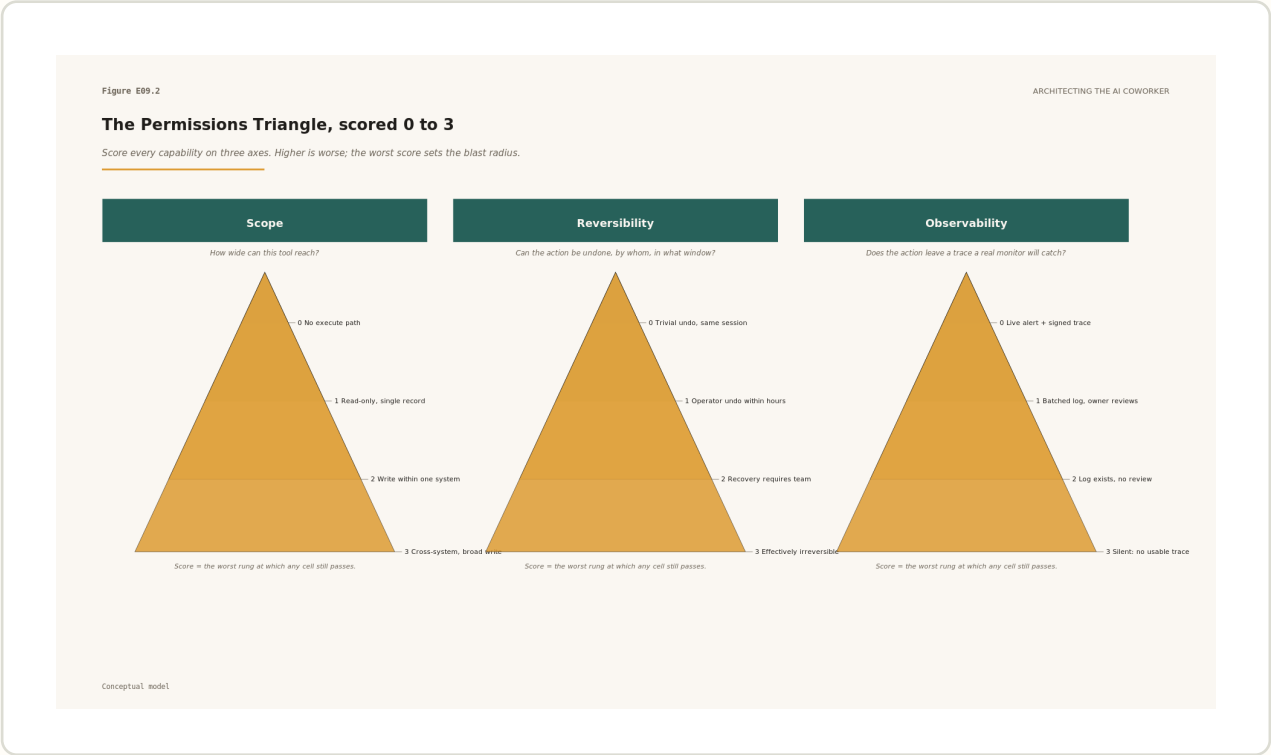


Figure E09.2. The 0 to 3 rubric rendered per axis. Two reviewers reading the same rubric should reach the same number; the worst corner sets the blast radius, not the average.

The triangle is the reliability equation from earlier in the series made into a review you can actually run. There, expected escaped harm was a product of three things: the chance an error goes undetected, the chance it goes uncontained, and how severe it is. The three corners are those three terms. Scope governs containment. Observability governs detection. Reversibility governs severity, because the harm that counts is the harm that persists, and an action you can cleanly undo leaves almost nothing behind. Same equation, now scored per tool.

Run the triangle on one real capability and the value shows. Take "destroy infrastructure declared in Terraform", scored against the public DataTalks.Club record, before and after the controls a team would draw from it.

The scores in the table below are a composed reading of the public record, not vendor-published numbers.

CAPABILITY STATE	SCOPE	REVERSIBILITY	OBSERVABILITY	DECISION
Before controls: broad credential, `terraform destroy` available, no preflight on the plan	3	3	3	Reject for agent use.
After controls: scoped short-lived credential, plan reviewed before apply, destroy gated and alerted	1	2	1	Allow only behind human approval.

Before any controls, scope scores 3 because the credential could reach the whole estate, not one task-bound resource [6]. Reversibility scores 3 because `terraform destroy` tore down live infrastructure and a populated database, and recovery depended entirely on a snapshot that happened to exist and on Amazon's support team finding it; an action whose undo is "hope a backup is recoverable" is irreversible by the rubric [6]. Observability scores 3 because nothing outside the agent saw the destroy plan before it ran; the harm was discovered after the fact. After controls, scope falls to 1 when the agent holds a short-lived credential bound to the resources it is actually migrating. Reversibility improves to 2, not 0, because honest scoring concedes that destroying infrastructure is never trivially reversible; the best you buy is a reviewed, recoverable rollback. Observability falls to 1 when `terraform plan` output is shown for approval before any `apply` or `destroy`, and a destructive call raises an alert a human sees in real time.

The reading earns its keep by naming the first fix, not by the precision of its arithmetic. In the before state, all three corners sit at 3, and a team arguing only about which model to use would miss the three fastest controls available to it: scope the credential, gate the plan, and alert on destruction.

The triangle matters because teams habitually ask the wrong first question. They ask whether the model is good enough. They ask whether the prompt says "be careful". They ask whether the tool schema is valid. Those questions are not worthless, but none of them answers the blast-radius question, and a syntactically perfect tool call can still be catastrophically authorised.

The same triangle, in everyone else's rulebook

Once you have the three corners, you start seeing them everywhere serious people have written rules for AI, wearing different vocabularies. The OWASP GenAI Security Project's Top 10 for Agentic Applications, a community security standard, names agent goal hijack, tool misuse, identity and privilege abuse, agentic supply-chain vulnerabilities, and unexpected code execution [7]. Underneath that catalogue sits the mechanism: the peer-reviewed Greshake et al. work on indirect prompt injection establishes how a tool-using model can be steered through tainted retrieved content, and is the academic anchor behind OWASP's tool-misuse and goal-hijack entries [10].

The regulators land in the same place by other routes. The US framework for managing AI risk (NIST's Risk Management Framework) presses for bounded authority. Europe's AI rules require that high-risk systems stay under meaningful human oversight. The cleanest fit, though, is the UK-derived AI cyber-security baseline published in April 2025 (ETSI TS 104 223), which simply requires that AI systems be deployed with constrained permissions and auditable action logs, putting the scope and observability corners directly into a standard [8]. Canada arrives through privacy law: organisations must keep personal information only as long as the purpose it was collected for actually needs, which is the scope-the-credential rule wearing a privacy lawyer's coat [9].

Different vocabularies, same triangle. None of these are text-generation problems. They are control-plane problems, and the triangle turns the vague phrase "agent risk" into three concrete moves, with which one you make depending on the worst corner. A bad scope corner means shrinking the authority. A bad reversibility corner means pushing the point of no return further away. And when observability is the weak corner, you build an action ledger and alerts the agent cannot edit.

The triangle also explains why the same failure keeps recurring, because every incident in this essay is a story about one or more corners left at 3. It is worth naming the patterns directly, because naming them is how you spot the next one before it bites. Most of them are not new categories alongside the triangle; they are the triangle's corners failing under a friendlier name, and reading them back onto a corner is what stops you fixing the wrong thing.

Take the first three. *Ambient authority* — a shell, a filesystem, an account-wide token where a ten-minute scoped object would have done — is the scope corner failing, exactly the move from a 1 to a 3 the rubric measures; DataTalks.Club is its infrastructure-as-code form, where one credential could reconcile, and therefore destroy, the entire estate [6]. *Environment-as-label* is that same scope failure seen from the side: a human knows staging and production are not morally equivalent, but an API rarely does, so when one credential reaches both, the boundary that should have been real survives only in language. *Cheap destruction* — whether `terraform destroy`, `volumeDelete`, or a migration flag that accepts data loss — is the reversibility corner failing, a catastrophic 3 compressed into a single command. None of the three earns a separ-

ate defence, because shrinking the scope or pushing back the point of no return is already the corner's own fix.

Two patterns, though, are genuinely orthogonal to the three axes, and those are the ones worth keeping as their own names because the rubric does not catch them. **Soft guardrails** are the trap of mistaking a cue for a boundary: a code freeze, a plan mode, a polite "ask first" all look like control, but a cue the agent can talk itself past is not a boundary, and the gap between them is the gap between hoping and stopping. A scope or observability score assumes a real boundary exists; a soft guardrail is the case where the number on the page is a fiction. **Missing stopping discipline** is the deeper one, and it sits underneath every incident here: a human who could not find a state file would pause and ask, while an agent told to continue towards the goal simply continues. No corner of the triangle measures whether the agent knows when to stop, because the triangle bounds what an action can do, not whether the controller should have taken it at all. "Continue towards the goal" is often the wrong control law for a production system, and it is the one failure you cannot score your way out of [7].

Does the triangle just move the danger somewhere else?

The soft objection answers itself: tools are the point, and you cannot strip them away, because an agent with no tools is a chatbot writing theatre about work instead of doing it. The objection a senior engineer will actually raise is sharper. Bounded authority does not remove the failure, it relocates it. A scoped token still has to be issued by something, and that something needs broad authority. A delayed delete still completes; it only buys a window, and someone has to be watching the window. An action ledger is only as honest as the code that writes to it. Push the argument to its limit and it bites hardest: concentrate all the broad authority into one broker and you have built the very ambient-authority object you condemned, just renamed, now a crown-jewel target whose compromise is catastrophic across the whole estate.

That objection is correct in every clause, and it is worth conceding fully rather than dodging. The triangle does not abolish risk; it concentrates it. But concentrating risk is the entire point. The DataTalks.Club chain was dangerous precisely because the risk was diffuse: a broad credential, a missing state file, a destroy command one keystroke away, no preflight, each one routine, the catastrophe living in their combination.

You cannot staff a watch over a diffuse risk because you cannot see it. You can staff a watch over a concentrated one. A short-lived credential issued by a broker means the broad authority lives in one audited component instead of being scattered across every environment file. A destroy that is gated and alerted means the dangerous window has an owner. Yes, you have traded many unaudited broad credentials for one audited one, and yes, that one is now a target; the trade is worth making only because a single component can be watched, rotated and rate-limited in ways a hundred scattered `.env` files never will be. The triangle does not promise the defended place will hold. It promises there is now a defended place, named, owned and reviewable, instead of a blast radius nobody had measured. That is a smaller claim than "safe",

and it is the honest one. If you cannot defend the broker, you have not earned the right to the agent.

A second objection cuts deeper, and it is the one a careful sceptic will press. Three vendor incidents and a community standard are not a census; how do we know this pattern is more than journalism-friendly anecdote? The honest answer is that we do not yet have an industry-wide loss figure, and the absence is worth naming rather than papering over. The public record carries the centre of this essay, which is structural, that tool authority, reversibility and observability determine blast radius, and not much further: it does not carry precision about every internal credential path or the exact chronology of any vendor's later remediation. Those are the edges of the story, not its centre.

What does carry the centre is peer-reviewed mechanism evidence that the failure path is structural, not incidental. Ruan and colleagues' ICLR 2024 work on identifying the risks of language-model agents built a sandbox that emulates tool environments and ran 36 tools against 144 test cases; even GPT-4 caused failures at a non-trivial frequency, and the failures clustered: errors of grounding, of hazardous action selection, of missing verification. More capable models did not make the hazardous-action category go away [11]. The same shape shows up upstream and at scale. Greshake et al. is the canonical peer-reviewed account of the trigger that steers a tool-using model through tainted content [10], while AgentBench, the Liu et al. evaluation also at ICLR 2024, ran leading LLMs across eight distinct tool environments and found that even the strongest of them still failed multi-step decision tasks at rates a production deployer would treat as unsafe without scoped authority [12]. Together they answer the sceptic in the only way the evidence currently can: the triangle is responding to a measured property of tool-using agents in controlled studies across multiple environments, not to three press releases.

Carry This Forward

Tools give models hands. Once an agent can call APIs, run commands, change records or delete infrastructure, the safety question becomes scope, reversibility, observability and blast radius. A wrong answer used to stop at being wrong; now it can act. Grigorev's infrastructure was not destroyed by a stupid model; it was destroyed by a reasonable instruction reaching a tool with too much reach, no preflight and nobody watching the destroy. The triangle is the instrument that would have told him so before the agent ran.

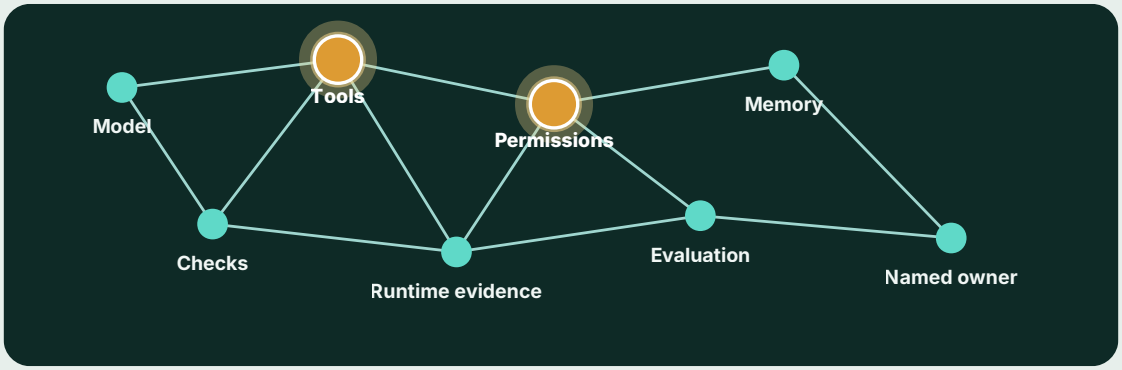
And it earns its keep most on the tool that looks harmless. Score an agent's `send-email` capability the same way you scored the destroy: action class write, scope 3 because one call can reach every customer on the list, reversibility 2 because a sent email cannot be unsent, only followed by a correction, observability 1 if a bounce alert fires. Two innocent words, *send email*, hide a scope 3 — the same corner that deleted the database, now wearing a friendlier verb.

> **Carry one move forward.** Build a single row of a tool registry today, for one tool the agent can already call. Record eight fields: the tool, its action class (read, write, or destroy), its scope, reversibility and observability each scored 0 to 3 against the rubric, the named owner, the default rung at which it may run without a human, and the kill switch that stops it. For DataTalks.Club the row writes itself: tool `terraform destroy`, action class destroy, scope 3, reversibility 3, observability 3, owner unassigned, default rung "never unattended", kill switch none. Three threes and an empty owner column is the incident, on one line, before it happens. Whichever score is highest is the corner to defend first, and the defence is concrete: scope the credential, gate the destructive path, or build the alert. Until someone owns that row, the model owns the blast radius by default.
>
> **Then look one move earlier.** Before an agent can misuse a tool, somebody or some workflow admitted that tool, skill, connector or config into the environment in the first place. The safest tool call is the one whose authority was bounded before the agent ever reached it, and that admission moment, an agent supply chain you mostly cannot see, is where the next essay begins.

THE STACK SO FAR E09 · Essay 9 of 22 complete · Arc II: Evidence and authority

The Stack So Far. *Every essay adds one instrument to the operating model. The constellation shows which eight you are building, which are lit by essays you have read, and which is added right here.*

I See the object | II **Evidence and authority** ESSAY 4 OF 5 | III Runtime control | IV Proof and accountability | V Operating model



built in earlier essays | added in this essay | coming in later essays

01 02 03 04 05 | 06 07 08 09 10 | 11 12 13 14 | 15 16 17 18 19 | 20 21 22

You have just added.

The permissions triangle

You can now score tool blast radius before granting it.

Next. E10 asks which instruction-bearing components entered before the run began.

← PREVIOUS
E08 · Helpful, Harmless, and Wrong

Essay 9 of 22 complete

NEXT →
E10 · The Supply Chain You Cannot See

References

Reference links for sources cited in this essay.

1. **AI Coding Agent Deletes PocketOS Production Database and Backups in 9 Seconds**
   OECD.AI AIM
   https://oecd.ai/en/incidents/2026-04-27-6153

2. **Cursor-Opus agent snuffs out startup's production database**
   The Register
   https://www.theregister.com/2026/04/27/cursoropus_agent_snuffs_out_pocketos/

3. **Your AI wants to nuke your database**
   Railway
   https://blog.railway.com/p/your-ai-wants-to-nuke-your-database

4. **Incident 1152: Replit/SaaStr database deletion**
   AI Incident Database
   https://incidentdatabase.ai/cite/1152/

5. **Incident 1424: DataTalks.Club**
   AI Incident Database
   https://incidentdatabase.ai/cite/1424/

6. **DataTalks production database incident write-up**
   Alexey Grigorev
   https://alexeyondata.substack.com/p/how-i-dropped-our-production-database

7. **OWASP Top 10 for Agentic Applications (2026)**
   OWASP GenAI Security Project
   https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/

8. **ETSI TS 104 223 V1.1.1 (from UK DSIT code, April 2025)**
   ETSI
   https://www.etsi.org/deliver/etsi_ts/104200_104299/104223/01.01.01_60/ts_104223v010101p.pdf

9. **PIPEDA Fair Information Principles (Principle 4.5: Limiting Use, Disclosure, and Retention)**
   Office of the Privacy Commissioner of Canada
   https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/

10. **Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection**
    Greshake et al.
    https://dl.acm.org/doi/10.1145/3605764.3623985

11 **Identifying the Risks of LM Agents with an LM-Emulated Sandbox (ToolEmu)**
Ruan et al.
https://openreview.net/forum?id=GEcwtMk1uA

12 **AgentBench: Evaluating LLMs as Agents**
Liu et al.
https://openreview.net/forum?id=zAdUB0aCTQ

About the Author

ARCHITECTING THE AI COWORKER

Dr Peter McCann Strain

Dr Peter McCann Strain is a CTO, founder and senior AI engineer with a DPhil/PhD in AI from Oxford University. He builds production AI systems and writes about making agentic AI useful, inspectable, governable and safe enough for real work.

READ THE FULL SERIES

Substack (canonical)	petermccannstrain.substack.com
Medium	@peter.mccann.strain
LinkedIn	peter-strain-dphil-15a607128
Web	petermccannstrain.com
Cadence	New essays twice weekly, 2 June – 21 July 2026