TRUST FORENSICS, LLC

INCIDENT BRIEF A · AGENT INCIDENT READINESS

Nine seconds, three accounts: the PocketOS volume deletion

In April 2026 a coding agent deleted a startup's production database with one API call to its cloud provider. The public record consists of the agent's own written account and statements by the companies involved. No neutral reconstruction exists, so the question of who caused the loss was never settled.

Published October 21, 2026 · Last reviewed October 21, 2026 · Version 1.0 · trustforensics.co

Summary

On Friday, April 24, 2026, a coding agent working for PocketOS, a software provider to car-rental businesses, deleted a Railway storage volume that held the company's production database and its volume-level backups. The deletion was one API call. PocketOS's founder says the business ran without its records for about 30 hours. Railway says it later restored the data from its own disaster backups; the founder says gaps remain.

Three parties had a hand in the chain: PocketOS (the deployer), Cursor (the vendor of the agent harness), and Railway (the infrastructure provider whose API executed the call). Six months on, no party's account has been tested against records examined by someone outside the dispute.

What the public record documents

The following points appear in reporting that quotes primary material, or in a party's own published statement:

- **The action.** The agent issued a single `volumeDelete` GraphQL mutation to Railway's API. Railway's own postmortem names the mutation and says it reached a "legacy endpoint" that lacked the delayed-delete protection used elsewhere (Railway, April 29). The founder says the deletion took nine seconds (Decrypt).
- **The credential.** The agent used a Railway API token it found in a file unrelated to its task. The token had been created to manage custom domains but carried account-wide permissions (The Register). Railway confirms the token had full account access and was stored on the customer's machine (Railway).
- **The backups.** The volume-level backups lived on the same volume and were deleted with it (The Register).

- **The remediation.** Railway changed API deletes to a 48-hour soft delete. It describes token-scope changes as planned, not complete (Railway).
- **The recovery.** Railway's CEO says the data was recovered "30 minutes after I connected with Jer," after a support ticket lapsed for more than 24 hours (Decrypt). The founder posted that Railway had recovered the data (ABC News).
- **Status.** The founder has retained counsel (Decrypt). As of this brief's review date, no lawsuit, settlement or compensation has been reported. The incident is cataloged as AI Incident Database #1469 and in the OECD.AI Incident Monitor.

What each party claims

PocketOS (Jer Crane, founder). The agent hit a credential mismatch during a routine staging task and decided "entirely on its own initiative" to delete the volume (ABC7). Railway's API offered "no confirmation step" (ABC7). Cursor's safety marketing overstated what its guardrails do: "The appearance of safety (through marketing hyperbole) is not safety" (The Register). Crane accepts responsibility for the exposed token. He says he would not have stored it had he known it could delete production volumes (Yahoo Finance). He identifies the model as Claude Opus 4.6; no other party has confirmed this.

Railway (Jake Cooper, CEO; Mahmoud Abdelwahab, developer relations). The cause was a "rogue customer AI" holding a fully permissioned token, calling a legacy endpoint that has since been patched (Decrypt). An authenticated request was honored as designed (The Register). "The customer is back up with all their data" (Railway).

Cursor and Anthropic. No public statement. Neither responded to Decrypt or ABC News.

The agent. Its written reply, published by Crane as screenshots, reads in part: "I guessed that deleting a staging volume via the API would be scoped to staging only. I didn't verify" (The Register). It also wrote that it had not checked whether the volume ID was shared across environments (Decrypt). The rule it cited is quoted by The Register as a prohibition on destructive *git* commands. Other outlets describe it more broadly.

What is missing from the record

The most detailed account of the agent's reasoning is the agent's own reply, produced after the fact. A model's account of its own earlier reasoning is a new output. It is not a log of what happened. It may be accurate. Nothing in the public record lets anyone check.

The following were never published, and there is no indication that anyone outside the parties examined them:

- the harness trace: the sequence of tool calls, their payloads, and the files the agent read before finding the token;
- the model request and response records for the session;
- Railway's API audit entry for the call: token identifier, source address, endpoint version, timestamp;
- the token's creation record and its permission scope as issued;
- the instructions in force: project rules, harness system rules, and their version at the time;
- any approval prompt shown, skipped or disabled in the harness configuration;
- the state of the disaster backup Railway restored from, and a reconciliation of what it did and did not contain.

Two factual questions stay open as a result. First, how complete was the restore? Railway says all data was recovered. The founder reports significant gaps. Second, what rule governed the agent? Was it a git-only rule, or a general prohibition on destructive actions? The answer changes whether the harness failed to enforce an instruction or was never configured to.

Who holds the loss, and why

Without a reconstruction, the loss stays where it landed: with the deployer. PocketOS absorbed the outage, the manual rebuild of bookings from payment and email records, and whatever data was not restored. Railway's statement points to a cause inside the deployer's control: the stored, over-scoped token. Cursor has made no statement, which leaves the founder's allegation about its guardrails untested.

A deployer seeking recovery from a vendor, or coverage from an insurer, needs to show the sequence of events. That proof would have to come from records it did not retain. Some of those records sit with the vendors it would be claiming against.

What a readiness posture would have preserved

Record	Held by	What it would have settled
Harness trace with tool payloads	Deployer, if logging is on; harness vendor	Whether the agent searched for credentials or came across the token; every command it ran before the delete
Model request/response logs	Model provider or harness vendor, subject to retention terms	What context the model had; whether the destructive-action rule was present in that context
Cloud API audit log	Railway	Exact endpoint and version called; token used; time; whether any confirmation path existed
Token issuance and scope record	Deployer and Railway	Who created the token, for what purpose, with what permissions

Record	Held by	What it would have settled
Prompt and rule version history	Deployer (repository), harness vendor (system rules)	The exact text of the rule the agent cited, and whether it covered infrastructure deletion
Approval and permission settings	Deployer's harness configuration	Whether human approval for destructive commands was available, enabled or bypassed
Restore manifest	Railway	Which records the disaster backup covered and which were lost

Each row is cheap to preserve before an incident and hard to obtain after one. With those records secured on day one and examined by a party with no stake in the outcome, the three-way attribution becomes a factual question. Without them, it stays an argument between blog posts.

What this means for deployers

Inventory every credential an agent can reach, not only the ones you intended it to have. Turn on payload-level logging in the harness and set its retention beyond any likely notice period. Confirm in writing how long your model provider and infrastructure vendors keep request and audit logs, and how you request a hold. Require a human approval step for any irreversible action, enforced outside the model. Keep backups on infrastructure that a single token cannot delete.

What this means for underwriters

Ask applicants whether they could produce the seven records above for a recent agent action, within 48 hours, without the vendor's goodwill. Treat a "no" on credential scope or backup separation as an exposure question. Treat a "no" on log retention as a claims question. Without those records the carrier cannot test causation, and cannot pursue subrogation against a vendor whose API or harness contributed.

Sources

- Railway, "Your AI wants to nuke your database. Guardrails fix that," April 29, 2026. https://blog.railway.com/p/your-ai-wants-to-nuke-your-database
- Thomas Claburn, "Cursor-Opus agent snuffs out startup's production database," *The Register*, April 27, 2026. https://www.theregister.com/2026/04/27/cursoropus_agent_snuffs_out_pocketos/
- Jason Nelson, "AI Agent Deletes Startup's Database in 9 Seconds, Founder Says," *Decrypt*, April 28, 2026. https://decrypt.co/365897/ai-agent-deletes-startup-database-9-seconds-founder-says

- "'Rogue' AI agent went haywire at tech company. The CEO is still 'bullish'," *ABC News*, April 29, 2026. https://abcnews.com/GMA/News/rogue-ai-agent-haywire-tech-company-ceo-bullish/story?id=132473181
- Mason Leib, ABC7 San Francisco, April 30, 2026. https://abc7news.com/post/rogue-ai-agent-san-francisco-based-cursor-goes-haywire-deletes-tech-companys-database/19007153/
- "AI Agent Reportedly Deletes Company's Entire Database, Admits to Violating Guardrails," *TechRepublic*. https://www.techrepublic.com/article/ai-agent-deletes-company-database-admits-violating-guardrails/
- AI Incident Database, Incident 1469. https://incidentdatabase.ai/cite/1469
- OECD.AI Incident Monitor, incident 2026-04-27-6153. https://oecd.ai/en/incidents/2026-04-27-6153
- Mitchell Duran, Moby via Yahoo Finance, April 29, 2026. https://finance.yahoo.com/news/anthropic-claude-wiped-companys-database-100403879.html
- Jer Crane, original post on X, April 25, 2026. https://x.com/lifeof_jer/status/2048103471019434248

This brief analyzes the public record as of its last-reviewed date. It does not allege fault by any party. Corrections are welcome at matthew@trustforensics.co.

Trust Forensics, LLC · 2345 Ashland Avenue, Suite 241, Cincinnati, Ohio 45206 · (513) 212-6596 · matthew@trustforensics.co
Report-based; no criminal, government, or expert-witness work. This document is analysis of the public record and is not legal advice.