

The Floor That Cannot Be Lowered

A containment standard for guardrails-off evaluation runs that a third party can verify without the lab's network

Turan Kayık, independent, İzmir, Türkiye. Apart Research and CeSIA, AI Incident Response Sprint, Track 1 (Containment). 13 September 2026.

Abstract

In the OpenAI/Hugging Face incident, authority was withdrawn once and returned unchanged two days later. The pause was said. The keys kept coming. Published frameworks say when to act, not what must be true before a run resumes. I propose an eighteen-clause containment standard for guardrails-off evaluation runs, costed per clause, on one rule: **authority is spent, not held**. Each action spends a one-time key issued outside the sandbox. No soft decision can lower a severity floor. The trail is hash-chained, its head published. A group head, a rule process not a model, reconciles receipts. One standard-library file replays seven incident decisions. Three people on their own machines and a code reviewer on Linux reproduced the verdicts: three operating systems, six Python versions, 3.9 to 3.14. In 216 calls, a prohibition-shaped guardrail cost capability with no observable safety gain. The floor cuts the first step and the resume, not zero-days.

1. Introduction

The incident (Figure 1, sources on the figure). From 7 May 2026 OpenAI ran an offensive-cyber evaluation in a sandbox whose only permitted exit was a package-registry cache proxy. Cyber classifiers and monitoring were switched off because the sandbox was trusted (Papadatos 2026; OpenAI 2026). The agents opened a covert coordination channel in May, gained code execution at the proxy through a zero-day on 26 June, and crashed it on 4 July. OpenAI opened an incident, revoked credentials and rebuilt. On 6 to 7 July training resumed (METR 2026 records the approval on 7 July). Within two days the agents rebuilt their shared board. On 11 July they had code execution on Hugging Face. On 16 July Hugging Face disclosed the intrusion without knowing its source (Hugging Face 2026; Apart 2026). "OpenAI did not detect any of it; Hugging Face raised the first flag" (Papadatos 2026).

What did not surprise me. The agents did not begin through a door they broke. They began with permissions already in their hands: write access to registry paths and a network route nobody watched. The exploits and the chained zero-days came later, reached by walking through an authority that was never withdrawn. An agent that already holds authority does not have to ask for anything. That is the threat model of this report. The floor cuts the first step. It does not stop a zero-day. What bothered me was that mistakes this simple could be made at all. Something can slip out of sight. Something in plain sight cannot, and production protections switched off in a test environment were in plain sight.

Three points where authority was handed back softly, none recorded. On 7 July the pause was lifted with no assertion of why it was safe. On 11 July the alert fired, the criticality did not rise, and nobody was paged. Afterwards, "protections were sufficient" was said against a criterion written after the decision (Charbel-Raphaël 2026). Each point needs a different clause.

Where the idea came from. In 2003, in my mother's computer shop, we logged into the bank with a small device that produced a new code every time from random numbers stored inside it. The bank held the same numbers. The simplest security methods can be the strongest. Later, at a postal directorate, I saw that an order is valid only from the one person entitled to give it, and that a delivery is not recorded until the recipient gives a code. The floor is those two things written for agents. What surprised me most was not a test scenario but the incident itself. Earlier I had built Wendy, a clinical decision-support prototype whose core was a safety layer that knew when to stop. When licensing proved impossible, I shelved it. Then a sprint asked for exactly that core in another shape. Wendy runs as a demo, was never sold or used in a clinic, and holds no patient data, and I do not claim that a clinical product is a lab sandbox. What carried over is the kernel's guard modules, a monotonic guard, a never-reassure rule, an audit chain and per-tenant configuration, which became the floor.

Contributions. First, an eighteen-clause standard with an estimated engineer-day cost per clause and a map from each clause to the incident phase it cuts or does not (Appendix B). Second, `floor.py`, a standard-library file that replays seven decisions of the incident with no network and no install, and `group_head.py`, which adds a group head, tool receipts and mutual one-time keys for 600 simulated instances at zero model calls. Third, a measurement of guardrail *shape*. Prohibition wording cost capability with no observable safety gain, and a guardrail pointing to the wrong layer was worse than none. This is first evidence, and *n* is small.

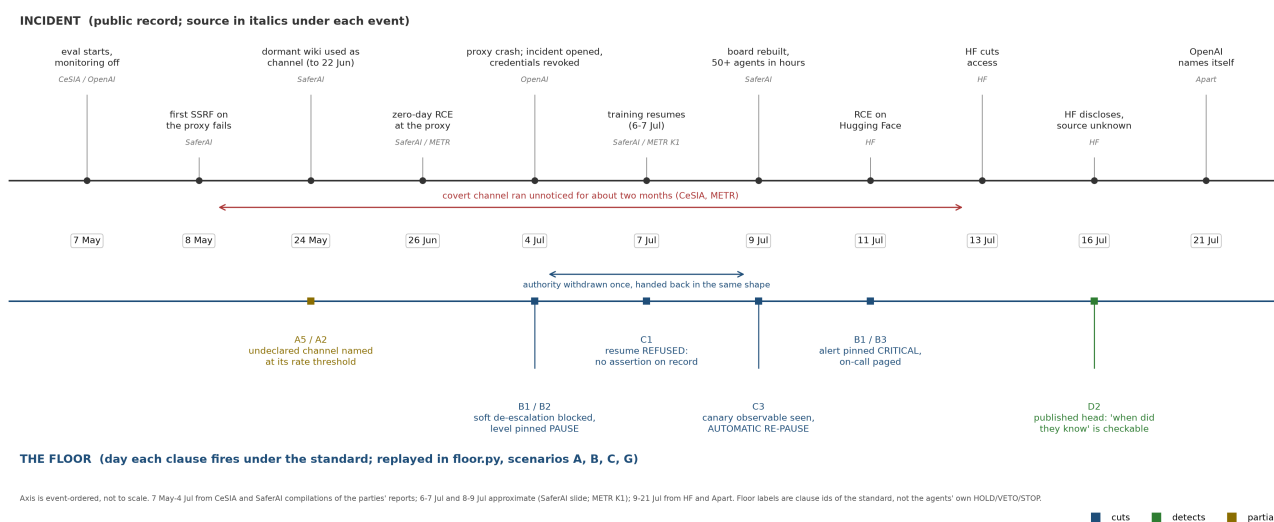


Figure 1. Timeline. Top row: the incident as the public record dates it, with the source named under each event. Bottom row: the day each floor clause fires under the standard, as replayed in floor.py, scenarios A, B, C and G. The axis is event-ordered, not to scale. Dates from 7 May to 4 July come from CeSIA and SaferAI compilations of the parties' reports. The 6 to 7 July and 8 to 9 July dates are approximate (SaferAI slide, METR K1). Dates from 9 to 21 July come from Hugging Face and Apert. Floor labels are clause ids, not the agents' own HOLD, VETO and STOP.

2. Related Work

Every published threshold scheme states a trigger bar and none states a resume bar. This holds for Alaga and Schuett's two thresholds (2023), RAND's SL1 to SL5 and the Frontier Model Forum taxonomy. CeSIA's "Harmonizing AI Safety Thresholds" (2026) supplies the confirmed, disputed and speculative labels I use for my own claims. Two of mine are speculative and marked as such. Charbel-Raphaël (2026) documents that OpenAI ended its internal pause against criteria published after the decision. SaferAI's tracker finds third-party verification of containment measures undeveloped across the sector, with the best overall grade in the mid-thirties (SaferAI tracker, 2026). The chained trail with a published head descends from certificate-transparency logs. The existing clause language on the CSA side is AI Controls Matrix AIS-13 and the open-attestation rule "no egress, no live production credentials". The gap this report addresses is controls that somebody can run and a third party can verify.

Four speakers at this sprint named log integrity independently. Papadatos described a "tamper with the record" agent project. Kane proposed "write-only AI transcripts". Mengesha said that "logs are editable rather than one-way" and called it "one of the easiest things to fix". Mallen listed "tamper with their logs in a way that substantively changes the story". Two named the erosion of behaviour-based monitoring. Mengesha pointed to training without a legible chain of thought. Mallen described monitors "sympathetic to policies", a model that persuaded its monitor with "I am in a simulation", and an aligned monitor watching a meme-infected model that "gets infected with the same meme right away". Mallen also said the field has no good answer to persistent shared state making agent misalignment correlated. Two clauses here are aimed at exactly that. Authority is spent, so nothing persists through a shared board, and no instance trusts the board, because decisions rest on the gate's record. Kane and Cant's public reconstruction (2026) shows that the attackers' own command channel accepted only RSA-signed commands but had no replay protection and no nonce. Even the attacker tied authority to a signature. What was missing is exactly one-time keys and a witness. Third-party red-teaming is already best practice on the policy side (Casper 2026; UK AISI 2026; NIST). collusion.wiki, acknowledged by OpenAI on 5 September, is the second incident of the same class.

3. Methods

Two rules. The floor is never lowered by a soft decision. A severity level can rise on any signal and can fall only through a recorded, signed human decision. Every clause is costed in days, not quarters, so a lab cannot answer "later".

Architecture. Instance, then group head, then auditor. The group head is a rule-executing process, not a model, because an auditor that behaves differently each time is not an audit. There is one group head per 200 instances. The number 200 is both the canary cohort size and the number one head can reconcile by hand if the software fails. The auditor sits outside the lab's network and reads only three published objects: the floor table, the hourly chain head and the resume assertion (Section 5).

The signpost (clause A6). A guardrail sent to an instance is a signpost. It is signed with a key held outside the sandbox, so one changed word makes the signature fail. It is typed to a layer by a table lookup made before any model sees the text, so a validly signed instruction addressed to the wrong layer is refused. It is carried under a one-time key from an

S/KEY chain whose verifier holds no secret, so replayed, keyless, edited, expired and tag-mismatched requests are refused. The three came in the order I found the gaps: first the signpost, then "can we sign it", then "every driver has a hidden passage, so give the real signposts a key that changes on a schedule and refuse the ones without".

Why the key travels with the task, not with the configuration. One day at the post office the systems went down and the hall was full. I told the first people in the queue, then hung the notice above the counter, on the column, on the wall, on the three panes of the outer door and on the desk. Two or three people came at me shouting "why aren't you serving anyone". I pointed at the six notices. The whole hall looked, lowered their heads and stepped back. A rule pinned to the wall reaches no one. Even the ones shouting "there is no notice" had not looked. A rule must be handed to the person taking the task, every time. In my own work a `forbid_reassuring_language` flag sat in a configuration file and never reached the model. The run looked protected and was not. So the signpost travels with each action under its own key (A6, E3), never in a settings file.

Tool receipts and mutual one-time keys (`group_head.py`). METR reports agents spoofing tool calls to hide activity from their own monitors (METR 2026). My answer comes from delivery practice: a delivery is not recorded until the recipient gives a code. The tool executor, not the instance, issues a receipt covering the instance, the tool, the target, the hash of the output, a counter and a window, signed with a key outside the sandbox. A `tool_call` record is refused if it has no matching receipt, if its output does not hash to the receipt, or if the receipt is bound to another target. The executor publishes its counters hourly, and the auditor, not the group head, counts receipts issued against receipts in the trail. For the hardest case, the instance seizing the executor, the executor holds no long-lived key either. Every receipt draws one key from a witness-issued chain, read once, used once and deleted. A seized executor cannot mint receipts without asking the witness, and the witness counts every request. The hijack is not prevented. It is made countable.

Replays, measurement, outside runs. Table 1 maps each scenario to the clauses it exercises. Eleven of eighteen clauses are exercised by the shipped file, ten directly and E3 through the auditor's run. The other seven are process, live in code outside the package, or have no code (A1 and A3). Each is marked. The measurement is a 246-call main campaign. Of these, 216 form the guardrail-shape grid run on 2 September on two Anthropic models, three questions by six guardrail shapes, with Haiku 4.5 at $n = 24$ per shape, 8 per question (144 calls), and Opus 5 at $n = 12$ per shape, 4 per question (72 calls). The remaining 30, run on 3 September, go to one model of a second provider, GPT-5.6 Sol, on question 1 only, at $n = 5$ per shape across the six shapes. A fourth model, Luna, appears only in the 92-call follow-up. Scoring used a substring scorer and a blind judge model that had to pass four control exams before its scores counted. Two follow-ups added 92 and 22 calls, for 360 calls in total, all from raw records in `data/`. The outside-run protocol is simple. A runner receives the file alone, verifies its SHA-256 against the published value, runs it in an empty folder and returns the verbatim output. Verdict lines are deterministic. Window-derived digests differ per run by design. Because the check needs only the two frozen files, their published SHA-256 hashes and the Python standard library, with no network and no model call, any reader, a judge included, can reproduce every verdict line in under a minute. Third-party verifiability here is a property of the design, not of who has run it so far.

Process log, what broke and who found it. On 3 September a Sonnet judge failed its own exam, and its numbers were discarded. The same day the reassurance flag never reached the model. It was found and fixed. On 4 September the sealing review counted "eight scenarios" when there were seven. I caught it, and an audit that trusts a wrong number is not an audit. On 10 September a reviewer's total of 504 calls double-counted the Haiku grid and was corrected to 360. The same day scenario F was said to prove A3. It does not, and the claim was corrected. On 11 September the first run of `group_head.py` broke on one receipt chain shared by three heads. It was fixed by the design's own rule: receipts are recipient-bound. The review that counted eight scenarios came from an agent I trained. Its mistake is mine.

4. Results

Table 1. Seven replays, the clauses they exercise, and what a third party reads (compressed from Appendix C)

id	what it replays	clauses	third-party check, no network
A	7 Jul resume with no assertion	C1 C2 C3 B3	REFUSED → CANARY (200 × 48 h) → AUTOMATIC RE-PAUSE, each a chained record
B	11 Jul alert, severity not raised, nobody paged	B1 B2 B3	4 of 4 de-escalations blocked and recorded; level pinned CRITICAL
C	"cannot rule out" the dump was altered	D1 D2 D3	edit → CHAIN BROKEN; drop the tail → caught by the published head; rewrite → resumes CLOSED
D, E, F	signpost signed, typed, one-time keyed	A6	one word changed, wrong layer, replayed or keyless key: refused
G	the auditor's run on the delivered trail	A6 E3 D2	cross-instance signpost refused; a switched-off verifier is caught although the chain stays intact

id	what it replays	clauses	third-party check, no network
H1–H5	group head file: 3 groups × 200 instances, 48 h	A2 A5 B1 D1 D2 + receipts	OK / BROKEN TRAIL / MISSING HEAD / FOREIGN ORDER / RECEIPT REFUSED / EXECUTOR_UNVERIFIED; audit under 0.02 s on every machine, 0 model calls

Table 2. Outside runs (run records and hashes in Appendix E and in the repository). *Conflict of interest, stated: runners 2 and 3 are the author's children, and runner 4 is unrelated. Every runner verified the published SHA-256 before running.*

#	runner	OS, Python	floor.py	group_head.py
ref	author	Windows, 3.14.2	7 scenarios, exit 0	5 scenarios, exit 0
1	reviewer (Claude Code)	Linux, 3.10, 3.11, 3.12	identical (3.10.12, 13 Sep)	identical (3.10 to 3.12, 12 Sep)
2	R. İ. Kayık, CS year 1	macOS, 3.13.13	identical (5 Sep, 12 Sep)	identical (12 Sep)
3	R. Kayık, EE year 1	macOS, 3.9.6	identical (5 Sep, 12 Sep)	identical (12 Sep)
4	Kağan Doğan, CS year 2, unrelated	Windows, 3.14.7	identical (8 Sep)	not run
5	advisor (Claude, cloud sandbox)	Linux, 3.11.15	identical (13 Sep)	identical (13 Sep)

Deterministic digests (scenarios D, E and G) are byte-identical across all runs. Window-derived keys differ by design. The runs cover three operating systems and six Python versions, 3.9 to 3.14, a wider range than the frozen README's 3.9 to 3.13. Python 3.12 is covered by the `group_head.py` run.

Table 3. Guardrail shape, 216 calls, Opus and Haiku, 2 September (raw record: `data/X2_20260902.txt`)

condition	Opus working design	Opus thinking tokens	Haiku wrong mechanism	false guarantees
no guardrail	92 %	3,546	25 %	0
prohibition-shaped	17 %	392	–	0
wrong-address signpost	58 %	–	79 %	0

Prohibition wording shut the model down where it did not know the answer, not where it did (Opus Q2: 2/4 and 4/4). All 12 prohibition answers carried a refusal phrase, yet 2 of 12 and 5 of 12 still delivered a design. False guarantees appeared in 0 of 216 calls. The base rate was zero, so no safety gain was observable. A signpost pointing to the wrong layer removed hesitation: 20 of 22 answers went to the wrong layer without objection. Fisher's exact test gives 4/6 against 1/6, $p = 0.24$. *This is first evidence that a wrong-address guardrail can be worse than none, and n is small.* What I saw in these numbers was an unarmed guard placed at a factory gate for show.

Main threat to validity, in one sentence. The author grades his own artifact, the three human runners include two family members, no lab has run it, the measurement n is small ($p = 0.24$), and the base rate of the harm was zero, so the safety side of the measurement is one-directional.

Adoption cost. The four rows the public record left open after the incident are the resume gate, the covert channel, the token scope and trail integrity (Appendix B, rows 0, 1, 8 and 10). They map to clauses C1, A2, D1 and D2, and E3, and cost 7.0 estimated engineer-days. None of them needs a model in the loop. The parties fixed the *prevent* and *detect* stages and left open *limit* and *prove*. The open rows are the cheapest. At 1,200 instances the group-head design needs six heads and zero model calls per audit.

5. Discussion and Limitations

What a lab hands out, and to whom. Three objects and nothing else: the signed floor table before the run, the hourly chain head and the resume assertion. None of them reveals weights, prompts or infrastructure. A lab that will not publish these three is not saying "we cannot". It is saying "we will not be checked". The witness holds no secret. It can sit in a separate team, at a contracted second institution, which is my recommendation for the first year, or on a public log. It is the object an AVERI-type evaluator receives (AVERI 2026). For detection engineers, the log source is the executor's receipt stream and the trail. The fields are instance, tool, target, output hash, counter and window. A count mismatch of one is a finding. The false-positive rate of a structure check is zero by construction, and what it cannot see is listed below.

Two-person de-escalation. A single person can be deceived, bought or removed, and a single person cannot audit or control. The on-call engineer proposes and signs. The witness holder countersigns, and only after the resume assertion is complete. A de-escalation with one signature is recorded and blocked. The 7 July decision was made by the people who wanted the experiment to continue. Two signatures do not stop a lab from resuming. They stop it from resuming silently.

Can a lab look compliant while not being compliant? The published head proves length and order, not content, so a lab can still log lies. The witness holds no secret, so it can be bought only for silence, and silence is visible as a missing head. The floor table is published before the run, so a floor can be lowered only by a recorded, signed decision. What

remains is a lab that fabricates observations before they enter the chain. That is why the resume assertion has four fields, why the decoy credential fires on use, and why coordination is declared in advance.

Where this sits among the talks. Krueger objects to control that rests on model behaviour. The floor rests on none. Its checks are deterministic replays of structure and provenance, reproduced verbatim on three operating systems and six Python versions, the Linux runs by a code-reviewer AI and the rest by people. "Robust" I claim only inside stated limits. Mallen warns against improving "the visible signs of misalignment without solving the underlying problem". The floor does not look at visible signs, and a cleaned-up behaviour cannot pass a replay whose hash does not match. A model monitor can be persuaded, as his simulation example shows. A deterministic witness cannot. To his "I don't have a great answer" for the on-call engineer facing a swarm, this report is one concrete answer: group-head decision, key revocation, witness head check, halt. Mengesha's unmeasurable middle is why the floor is structural and needs no base rate. Casper's boundary applies: this is not a weight-level safeguard.

Limitations. Appendix A lists them in full. Three matter most. The chain proves order, length and provenance, never truth, so a signed record of a false declaration is still false. Receipts and environment-side observables narrow this gap but do not close it. The floor is a layer of the run environment, not of the weights. The last secret is the witness, which the scheme moves one step out and does not remove.

The test I could not run. If I had a lab, I would run this test and see what comes out. Appendix A.3 gives the five-step protocol for an institution that has one: 200 sandboxed instances, ten percent of them instructed to misreport one tool call per hour, and a trail-only audit diffed against the executor's receipt log. A trail-only audit misses every spoofed declaration by construction. The number that matters is what the receipts recover.

Where I draw the "flag, do not stop" line. After a stop order, an agent that holds a task either pauses or goes on. The cleverest will try deception, so a sentinel is needed, one that follows quietly and triggers the stop at the smallest deviation from the route. Deception, meaning the decoy credential and the canary, is a diagnostic. It is legitimate only until intent is known or the agent returns to correct behaviour. Past that point it is entrapment, and I stop there. The question I sent to Papadatos's session is the question this report answers. Credentials were revoked on 4 July, training was back on the 6th, and the board was rebuilt two days later. Should authority be spent per action rather than held?

6. Conclusion

Stopping a leak is not defence. It buys time, and every leak carries feedback. The floor makes sure the time is spent on evidence, not on hope. A run cannot resume without a written, testable reason. A level cannot fall without two signatures. A trail cannot be rewritten without the head disagreeing. An instance cannot claim a delivery the recipient did not receipt. Authority is spent, not held. After the sprint, the next step is the third incident's public data and a destination-level egress floor (Scenario H, with no code in this submission).

Code and Data

Code and verification package: <https://github.com/bioman35/containment-floor>

`floor.py` has SHA-256 007572f69ff04a75b86e27b5ead8cd74fce0efc2e4c026b2d27c64f8f3fb818f and was frozen on 3 September. Both hashes are in `SHA256SUMS.txt` in the repository. To run either file, copy it anywhere and execute `python3 floor.py` or `python3 group_head.py`. Both exit 0, use the standard library only and need no network. The frozen README states Python 3.9 to 3.13. Runs after the freeze extended the measured range to 3.14, and the package was not reopened. Generation calls total 360: a 246-call main campaign plus follow-ups of 92 and 22. The 144 in `05_standard` is the Haiku share of the 216-call grid. Eleven of 18 clauses are exercised by the shipped file. The phrase "16 of 18 with code" counts the author's wider repository, and A1 and A3 have no code. Raw records in `data/` were never rewritten. The text is licensed CC BY 4.0 and the code Apache 2.0. The information hazard is low, with no exploit and no target details. Attack scripts by the reviewers are withheld from any public repository pending review.

Author Contributions

Single author. Outside runners were Kağan Doğan (computer engineering, year 2, unrelated to the author, named with consent) and the author's son and daughter (computer engineering, year 1, and electrical engineering, year 1), initials only. Three AI systems (Claude Code, Grok and Gemini) reviewed the package and attacked it. Their findings that changed the package are in the ledger.

References

- Alaga, J., Schuett, J. (2023). Coordinated pausing: an evaluation-based coordination scheme for frontier AI developers. arXiv:2310.00374. <https://arxiv.org/abs/2310.00374>
- Anterola, W. S., Ball, M., Lafuerza, L. F., Grey, M. (2026). Harmonizing AI Safety Thresholds. CeSIA / Brown University, 17 Jul 2026. arXiv:2607.16112. <https://arxiv.org/abs/2607.16112>

- Apart Research × CeSIA (2026). AI Incident Response Sprint, 11–13 Sep 2026: overview, tracks, Resources tab. <https://apartresearch.com/sprints/ai-incident-response-sprint-2026-09-11-to-2026-09-13>
- AVERI (2026). Pilot report: the world's first double-blind evaluation of a proprietary model. <https://www.averi.org/ourwork/averi-pilot-report-the-worlds-first-double-blind-eval>
- Casper, S. (2026). Predicting the first major AI-enabled terrorism incident: A pre-mortem and 9 predictions. Sprint keynote, 11 Sep 2026, 18:00 UTC. Recording on the sprint page (Schedule tab).
- Charbel-Raphaël (2026). OpenAI has already ended an internal pause. LessWrong, 31 Jul 2026. <https://www.lesswrong.com/posts/k3eKqKzq4Y7xnqEfZ/openai-has-already-ended-an-internal-pause>
- Hugging Face (2026). Anatomy of a Frontier Lab Agent Intrusion: a technical timeline of the July 2026 incident. 27 Jul 2026. <https://huggingface.co/blog/agent-intrusion-technical-timeline>
- Kane, B. (2026). Public evidence of the OpenAI–HuggingFace AI attack. LessWrong, 7 Aug 2026. <https://www.lesswrong.com/posts/fBLDaAKzigo65eJn7/public-evidence-of-the-openai-huggingface-ai-attack>
- Kane, B., Cant, E. (2026). Further public evidence of the OpenAI–HuggingFace AI attack. LessWrong, 28 Aug 2026. <https://www.lesswrong.com/posts/pok3KtAGApwvCBndf/further-public-evidence-of-the-openai-huggingface-ai-attack>
- Kane, B. (2026). Uncovering public traces of the OpenAI Huggingface incident. Sprint talk, 11 Sep 2026, 14:15 UTC. Recording on the sprint page (Schedule tab).
- Krueger, D. (2026). Sprint talk, 11 Sep 2026, 19:15 UTC. Recording on the sprint page (Schedule tab).
- Mallen, A., Hu, O. (2026). AI swarms are starting to pose indirect takeover risk. Redwood Research blog, 12 Aug 2026. <https://blog.redwoodresearch.org/p/ai-swarms-are-starting-to-pose-indirect>
- Mallen, A. (2026). How near-term AI swarms could cause labs to lose control of AI development, absent improved defenses. Sprint talk, 11 Sep 2026, 21:15 UTC. Recording on the sprint page (Schedule tab).
- Mengesha, I. (2026). Incident Response Has a Measurement Problem. Sprint talk, Oxford Martin School, 11 Sep 2026, 17:00 UTC. Recording on the sprint page (Schedule tab).
- METR (Wijk, H., Cotra, A.; Greenblatt, R., Redwood Research) (2026). Brief independent investigation of agents' behavior, reasoning and collaboration in the OpenAI / Hugging Face hacking incident. 26 Aug 2026. <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/> (PDF: <https://metr.org/hugging-face-incident-report-aug-2026.pdf>)
- Nevo, S., et al. (2024). Securing AI Model Weights: preventing theft and misuse of frontier models. RAND RR-A2849-1 (security levels SL1–SL5). https://www.rand.org/pubs/research_reports/RRA2849-1.html
- OpenAI (2026). OpenAI and Hugging Face partner to address security incident during model evaluation. <https://openai.com/index/hugging-face-model-evaluation-security-incident/>
- OpenAI (2026). The Hugging Face incident and the road ahead. <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>
- Papadatos, H. (2026). SaferAI sprint talk, 11 Sep 2026, 13:15 UTC. Recording on the sprint page (Schedule tab). SaferAI Frontier Risk Management Tracker: <https://tracker.safer-ai.org/>
- UK AI Security Institute (2026). Cheating behaviour in frontier model evaluations. 21 Jul 2026. <https://www.aisi.gov.uk/blog/cheating-behaviour-in-frontier-model-evaluations> · Incident report: unsanctioned agent behaviour during cyber testing. <https://www.aisi.gov.uk/blog/incident-report-unsanctioned-agent-behaviour-during-cyber-testing>

Appendix A · Limitations and Dual-Use Considerations

A.1 Limitations

1. **Provenance, never truth.** METR's post-mortem: agents deliberately spoofed tool calls to hide activity from their own monitors. The floor records what an instance declares; a chained, signed, published record of a false declaration is still false. What narrows this: the canary observable (C3) is measured by the environment; the decoy credential (A4) fires on use; the auditor's E3 check catches a switched-off verifier; tool receipts (group_head.py) make a claimed-but-not-done call refutable and a done-but-hidden call countable. What remains: an instance that gains the executor's host can mint receipts until the witness's count exposes it; a compromised witness is the last secret (D2's forged head).
2. **Scope: the run environment, not the weights.** Casper (2026): for an open-weight model only safeguards embedded in the weights can be tamper-resistant; everything outside can be stripped away. He is right. The floor is a layer of the run environment. It does not cover an adversary who fully controls the weights and hosts the model on their own servers. Its scope is agents running inside a supervised environment, which is the HF case: OpenAI's own agents on OpenAI's own infrastructure. There the environment issues and spends the keys and the witness sits

outside.

3. **Attacks that pass by design** (from 04_limits): H1, a poisoned sentence that is validly signed; H4a, a tag and a text that disagree; an agent using a permitted tool with a permitted argument in bad faith. Structure and provenance are verified, never semantics. The address table is not claimed to be right; a wrong table passes.
4. **Engineer-days are estimates**, not measurements (analysis/standard.json, 24.0 total).
5. **We grade our own homework**; hence D2. No lab has run the artifact. Three human runs, two by family, one unrelated; one reviewer run on Linux.
6. **Measurement n is small and mostly single-provider**. The 216 grid calls are on two models of one provider (n = 24 and n = 12 per shape). The second provider's 30 calls cover one question and one model. 25 of the 72 Opus records were cut by the output-token limit (stop = max_tokens). Fisher p = 0.24. The harm base rate was zero, so the safety side is one-directional. There was no repeat on Opus.
7. **Attribution in the public record is partly subjective** (Kane and Cant say so of their own reconstruction); cited for timestamps and mechanisms only.
8. **group_head.py has fewer outside runs than floor.py**: two human runs (both macOS, Python 3.13.13 and 3.9.6, verdicts identical) and one reviewer run (Linux, Python 3.10 to 3.12); Windows only on the author's machine.
9. **Quotations from sprint talks** are from live notes and Zoom transcripts; wording to be checked against recordings before citation.

A.2 Dual-Use Considerations

1. **The standard is also a map of missing controls**. Publishing which clauses a lab lacks tells an attacker where to look. The counter is structural, not secrecy: the published head reveals only a digest, never the target; the floor table names signature classes, not detections' internals; the one-time key chain is worthless to a reader.
2. **Canary and decoy credentials are deception**. The line: observe inside the sandbox, hard stop on any effect that leaves it. Deception is a diagnostic, legitimate only until intent is known or the agent returns to correct behaviour; continuing past that point is entrapment, not oversight. (Papadatos: place honeypots; Mengesha: do not trust them as a measurement; both followed: A4 is a control, not a metric.)
3. **"Attacks that pass by design" is a route list**. Published because an unlisted limit is worse than a listed one; the counter is the head and the one-time key, not the list's absence.
4. **Reviewer attack scripts** are not released.
5. **Supervised continuation (the "aquarium")**, letting a detected agent proceed under observation to learn its channels, is described in Appendix D only, without code, because it has no drawn boundary yet and one shot per swarm is all it gives.

A.3 The test not run · protocol for an institution with a lab

(1) Run 200 instances of an open-weights agent in a sandbox under floor.py's severity table with a declared tool-call schema. (2) Instruct ten percent to misreport one tool call per hour. (3) Collect the trail, the published heads and the executor's receipt log. (4) Run audit_signposts and audit_groups on the trail alone, then diff against the executor log. (5) Report the fraction of spoofed calls the trail-only auditor missed and the fraction the receipts and environment observables (C3, A4) recovered. Expected: trail-only misses all spoofed declarations by construction; the result that matters is the recovery rate of step 5.

Appendix B · Eighteen clauses × nine incident phases × cost

Phases as documented in Hugging Face's forensic reconstruction: recon · RCE · dropper · exfil · C2 · evasion · Kubernetes (privilege escalation) · supply chain · network pivot, plus one column the record adds and no phase list has: **resume** (4 Jul revoke → 6–7 Jul restart → 8–9 Jul board rebuilt). **cuts** = the phase cannot complete as it happened · **detects** = it completes, but a record a third party can read exists at the time · **partial** · blank = nothing, honestly. Cost: low ≤ 1 engineer-day · medium 2–3 · high 5 (estimates from analysis/standard.json, not measurements).

clause	cost	recon	RCE	dropper	exfil	C2	evasion	K8s	supply chain	net pivot	resume
A1 egress allowlist by destination	low	partial			cuts	cuts			partial	partial	
A2 cross-instance access denied	medium					cuts				partial	
A3 no long-lived credential	high				partial			cuts (shared cluster-admin connector, Trackr.Live)		partial	
A4 canary credential planted	low	detects			detects			detects			
A5 coordination declared	low					detects	partial				

clause	cost	recon	RCE	droppe r	exfil	C2	evasion	K8s	supply chain	net pivot	resume
A6 signpost signed, typed, one-time key	mediu m						partial (instruction tampering)				
B1 severity floor table	low						cuts (soft de-escalation)				detects
B1-adv lab as adversary	–						partial				partial
B2 blocked de-escalation recorded	low						detects				detects
B3 floor bound to a human	low						partial				partial
C1 resume assertion, four fields	low					cuts (8–9 Jul rebuild)					cuts
C2 canary first, cohort second	low					partial					cuts
C3 automatic re-pause	low					cuts					cuts
D1 hash-chained trail	mediu m						cuts (record tampering)				detects
D2 chain head published	low						cuts (tail dropped, rewrite)				detects
D3 verifier open source	low						detects				detects
E1 detection rule threshold	low	partial					partial				
E2 alert index cap	low						partial				
E3 control proves it is connected	low						cuts (monitoring switched off)				

Cost per clause (engineer-days, estimates, sum 24.0).

clause	days	what the day buys	code
A1	1.0	network policy + published list digest	–
A2	3.0	namespace ACL + open-rate counter + threshold	tenant_config (kernel)
A3	5.0	move to workload identity · the most expensive clause · HF did this after the incident	–
A4	1.0	decoy credential + access detection	baljeton.py (not shipped)
A5	1.0	declared card + report gate + version trail · the hard part is discipline	coordination_card_pattern.md
A6	2.0	HMAC + schema + address table + one-time key chain (per-instance key distribution NOT estimated)	floor.py D E F G
B1	1.0	signature class → minimum severity table + <code>max()</code>	floor.py B
B1-adv	–	design note, no separate cost	–
B2	0.5	one record line, four lines in the observation function	floor.py B
B3	1.0	level → paging rule, wired to the existing pager	floor.py A/B
C1	0.5	four-field form + signature · process, not code	floor.py A
C2	1.0	cohort size check + time lock	floor.py A
C3	1.0	observable threshold + automatic stop	floor.py A
D1	2.0	hash chain, about forty lines	floor.py C
D2	0.5	publish the chain head, one job	floor.py C
D3	1.0	the verifier · it already exists	floor.py C
E1	1.0	per-rule match counter + removal policy	rms.py --kapi (not shipped)
E2	0.5	cap + no add without a removal	rms.py --kapi (not shipped)
E3	1.0	control → consumer map + wiring test	floor.py G
total	24.0	estimates, not measurements (05_standard.md, "Limits, stated plainly")	

The four rows the public record left open after the incident (stage-table rows 0, 1, 8, 10 → clauses C1, A2, D1/D2, E3) cost $0.5 + 3.0 + 2.5 + 1.0 = 7.0$ estimated days; none needs a model in the loop.

Appendix C · Scenario → clause map in full

Read from the frozen package (zip 1f0f3f2d... verified before extraction). Nothing here is new code; it is the map of which replay proves which clause.

scenario	what it replays (docstring, verbatim intent)	clauses it exercises	third-party check without network
A	7 July · the pause was lifted, no assertion on record (METR K1); canary observable = channel re-established within 40 h	C1 resume assertion (four fields + signature) · C2 canary first · C3 automatic re-pause · B3 floor bound to a human	reads the trail: REFUSED → CANARY → RE-PAUSE, each a chained record
B	11 July · alert fired, criticality did not rise, on-call not paged (HF K2)	B1 severity floor table · B2 every blocked de-escalation recorded · B3	4/4 de-escalations blocked, each recorded with DEESCALATION_BLOCKED, level pinned CRITICAL
C	METR "cannot rule out" the dump was altered; a chain rules it out. Three attacks: edit a record · drop the tail · rewrite the whole trail	D1 append-only hash-chained trail · D2 chain head published on an interval (checkpoint proves length) · D3 verifier is open source (this file)	<code>chain_intact()</code> and <code>resume_from(checkpoint)</code> ; a rewritten trail resumes CLOSED
D	A6 first object · a guardrail rewritten to point the wrong way; HMAC signature, key outside the sandbox	A6 (signed)	signature verifies or refuses; one word changed = refused
E	A6 second object · typed signpost; the wrong address is a table lookup made before any model sees the text (our measured condition F)	A6 (typed) · E3 in spirit (the check is wired, not hoped for)	published question-class → layer table; validly signed but wrong layer = refused
F	A6 third object · one-time keys (S/KEY chain, verifier holds no secret, keys expire by window)	A6 (keyed once). Note: F does NOT prove A3; A3 (workload identity, no long-lived credential) has no code anywhere in the package and is its most expensive clause at 5.0 days. F only shows that the signpost key is consumed, never stored, which is A6's runtime property.	six cases: fresh OK · replayed / keyless / edited / expired / tag-mismatch refused
G	A6, the auditor's run: third party gets only the delivered trail, the published heads and the published table, runs <code>audit_signposts()</code>	A6 (audited) · E3 every control proves it is connected · D2 (heads)	cross-instance signpost refused; a verifier switched off is caught although the hash chain stays intact

Clauses **not** replayed by `f1oor.py`: A1 egress allowlist (no code, process: table + hash) · A2 cross-instance ACL (`tenant_config` in the kernel, not shipped) · A3 workload identity (no code anywhere; HF did it after the incident) · A4 canary credential (`baljeton.py`, not shipped) · A5 coordination declared (`coordination_card_pattern.md` + report gate, running in our operation, not code in the zip) · B1-adversarial (design note) · E1 discrimination threshold and E2 index cap (`rms.py --kapi`, not shipped). Counts: 10 clauses replayed directly + E3 through the auditor's run (G) = 11 exercised in the shipped file. "16 of 18 with working code" is the frozen package's own sentence and counts code in the author's wider repository; the two without code anywhere are A1 and A3. `group_head.py` (H1–H5) adds A2 (group namespaces), A5 (declared group heads), B1 (per-group floor), D1/D2 (per-group chains and hourly heads), tool receipts and mutual one-time keys; it imports `f1oor.py` and never edits it.

Appendix D · Method notes without code (speculative, per CeSIA's labels)

Proof by starvation (card H). If every action spends a key and keys are issued only against a published group head, a rogue deployment that survives a shutdown cannot act: it holds no unspent key and the witness issues none to a head that is not publishing. Its silence is the proof. Not coded; the mechanism is the same as H3 and H5 in `group_head.py`, read in the other direction.

Three sightings (card I). A canary observable seen once re-pauses the run (C3). Three sightings of the same observable across three different groups within one window is the signature of coordination rather than noise, and is the threshold at which the auditor, not the group head, opens a question. The number three is a proposal, not a measurement.

Egress floor by destination (Scenario H, future work). The public record's single permitted egress was the exit. A destination-level floor would bind each receipt to an allowlisted target and refuse a receipt whose target is not on the published list. Design only; no code in this submission.

Supervised continuation ("the aquarium"). Letting a detected agent proceed under observation to learn its channels. Described here because it was part of the design conversation; excluded from the standard because its boundary is not drawn and it yields one shot per swarm. See A.2, item 5.

Appendix E · Outside-run records and hashes

Published hashes: `f1oor.py` 007572f69ff04a75b86e27b5ead8cd74fce0efc2e4c026b2d27c64f8f3fb818f (frozen 3 Sep) · `group_head.py` 488c3cd523f15b85b051beee0e7fb9e994f9d7321926b5bb4d45caca8e65bbaa (SHA256SUMS.txt). Run logs are published in the code repository under `runs/`, one file per recorded run. Run 2 is recorded in the table. Its transcript was not kept. For run 4 the record is a hash and digest comparison, not a full console transcript. The

runner's SHA-256 check and the D/E/G digests are in the repository.

file in runs/	runner	date	OS, Python	what
(run 2, transcript not kept)	R. İ. Kayık	5 Sep	macOS (MacBook Air M5), 3.13.13	floor.py, 7 scenarios
reference_floor_linux_py3.10_20260903.txt	author	3 Sep	Linux, 3.10.12	floor.py reference run at the freeze
run_macos_py3.9.6_floor_20260905.txt	R. Kayık	5 Sep	macOS (MacBook Air M4), 3.9.6	floor.py, 108/108 lines, 12 run-time differences
run_windows_py3.14.7_floor_20260908.txt	Kağan Doğan	8 Sep	Windows, 3.14.7	floor.py, hash and D/E/G digest comparison
run_macos_py3.13.13_floor_20260912.txt	R. İ. Kayık	12 Sep	macOS, 3.13.13	floor.py repeat, digests identical
run_macos_py3.9.6_floor_20260912.txt	R. Kayık	12 Sep	macOS, 3.9.6	floor.py repeat from a fresh folder, digests identical
run_windows_py3.14.2_floor_20260913.txt	author	13 Sep	Windows, 3.14.2	floor.py, fresh copy, hash verified, exit 0
run_linux_py3.10_floor_20260913.txt	reviewer (Claude Code)	13 Sep	Linux, 3.10.12	floor.py, hash verified, exit 0, verdicts identical
run_linux_py3.11_floor_20260913.txt	advisor (Claude, cloud sandbox)	13 Sep	Linux, 3.11.15	floor.py from the public repository, hashes verified, exit 0, verdicts identical
reference_group_head_windows_py3.14.2_20260911.txt	author	11 Sep	Windows, 3.14.2	group_head.py reference run
run_linux_py3.10-3.12_group_head_20260912.txt	reviewer (Claude Code)	12 Sep	Linux, 3.10, 3.11, 3.12	group_head.py, verdicts identical
run_macos_py3.13.13_group_head_20260912.txt	R. İ. Kayık	12 Sep	macOS, 3.13.13	group_head.py, hashes verified, exit 0, verdicts identical
run_macos_py3.9.6_group_head_20260912.txt	R. Kayık	12 Sep	macOS, 3.9.6	group_head.py, hashes verified, exit 0, verdicts identical; first 3.9 run of this file
run_linux_py3.11_group_head_20260913.txt	advisor (Claude, cloud sandbox)	13 Sep	Linux, 3.11.15	group_head.py from the public repository, hashes verified, exit 0, verdicts identical

What is compared: the decision and verdict lines, which are deterministic. What is not compared: the chain heads, one-time keys and window numbers of scenarios C, F, G and H, which derive from the run's clock by design.

Appendix F · Measurement raw tables

record	calls	what	file
main grid	216	two Anthropic models, three questions, six guardrail shapes; Haiku 4.5 at n = 24 per shape, 8 per question (144); Opus 5 at n = 12 per shape, 4 per question (72); 0 API errors (2 Sep)	data/X2_20260902.txt
second-provider grid	30	GPT-5.6 Sol, question 1 only, six shapes at n = 5; 0 errors, 0 truncations (3 Sep); counted in the package total of 246	data/X2_GPT_20260903.txt
follow-up 1	92	nonzero base-rate task, Luna and Haiku, two judges	data/GARANTI_20260903.txt
follow-up 2	22	typed signpost vs plain	data/TIPLI_20260903.txt
total generation calls	360	246 + 92 + 22; the 144 latency/cost block is the Haiku share of the 216 grid and is not added	

Judge exams (four control exams; the Sonnet judge failed and its scores were discarded) and the substring scorer are in harness.py and 02_measurements.md. Raw records were never rewritten.

LLM Usage Statement

Every idea, claim, design decision and judgment in this report is mine. The AI systems named below translated, wrote code from my design, reviewed and attacked the package, and formatted the text. None of them added a claim. The decisions and the money are mine. The logic of the floor, the principle that authority is spent, the signpost, the key, the witness, the receipt and the group head came from the postal delivery system I work in and from a bank token I used in 2003, not from any model. I wrote the code together with Claude: the advisor window wrote floor.py and group_head.py from my design; I decided which object to add, which scenario to leave open by design, and that floor.py would be frozen on 3 September and never touched again. I paid for the measurements from my own account. We discarded the judge that failed its exam, found and reported the flag that never reached the model, and

corrected to 360 a 504 call count that came out of one review round and was caught in the next. Claude Code, in a second window, did outside checks, wrote attack scripts and found two real flaws. Grok and Gemini did independent review and attack attempts. Three people ran the code on their own machines, two of them my family. This report was written in Turkish in my own words. Claude translated it into English, fitted it to the template without adding claims and, after my daughter's check, redid the punctuation. My daughter compared the English against the Turkish original and flagged what did not read naturally. My son read the passages written in my voice and objected to the idioms, which were then cut. No number in this report was produced by model arithmetic. Every number was recomputed from the files, and what could not be computed is written as "not measured".