
Two Witnesses: An evidentiary coalition audit of AI-agent incident disclosure

Author: Michelle Wanjiku Thuo
Affiliation: African Civic Trust (ACT)
Apart Research x CeSIA AI Incident Response Sprint
thuomichelle2020@gmail.com

Abstract

AI-agent incidents scatter evidence across organizations an agent touches. I built a method for testing, fact by fact, whether one organization's evidence is enough to establish a safety relevant claim or whether more than one is needed. Applied to 32 facts from two 2026 incidents involving the same developer, a Hugging Face intrusion and an undisclosed wiki misuse, 26 were single stakeholder sufficient. Two facts needed evidence from both organizations when they became public. One, attribution of the Hugging Face intrusion to its developer, is the clearest case. Neither Hugging Face's disclosure nor the developer's internal signal alone identified who was responsible then, although the developer's later account is sufficient today. The other involves contributions from both sides. This demonstrates the method, not that incidents generally need more than one witness or that environment or collective intelligence explains it. The companion tool, Evidence Coalition Explorer, lets readers test all 32 facts.

1 Introduction

When an AI agent does something inside a company that isn't its own developer, the evidence about what happened doesn't sit in one place. Some of it lives on the developer's servers. Some of it lives on the affected company's servers. Sometimes a bit of it is just sitting in public, in a wiki's edit history or a domain registry.

This project sits in Track 2, What happened, and what breaks next, and focuses on a more specific question within that problem. For a real AI-agent incident, which facts can one organization's evidence establish by itself and which facts genuinely need two organizations' evidence together? The point is not simply that evidence is spread out. It is to identify which specific claims actually depend on that spread and which happen to have been reported by more than one party without needing to be.

I took two real incidents from 2026 in which AI-agent activity crossed organizational boundaries. The incidents involved different affected organizations and different mechanisms, but both left evidence distributed across more than one stakeholder. One was a security intrusion into Hugging Face. The other was misuse of a public wiki that the developer did not publicly disclose until someone else found it. I broke the public record of each incident into 32 separate, checkable facts and tested each one the same way. I removed one side's evidence and asked whether the fact still stood. Then I put that evidence back and removed the other side's evidence instead. I also tried something stricter. I left both sides' evidence in place and removed only the act of connecting them, to see whether a fact needed the evidence to exist or needed someone to actually join it together.

Most facts needed only one side. I classified 26 of the 32 as needing only one stakeholder's evidence. That is an analytical call about each fact, not just a literal reading of the switches and the reason for each classification matters.

Two facts had meaningful two side dependence in the evidence record. One of those two is the clearest example in the whole set. It concerns who was actually responsible for the Hugging Face intrusion. When Hugging Face first went public, it did not name anyone. When the developer's internal alert first went off, it said nothing about Hugging Face. Neither side's evidence, on its own, told anyone that the developer's own agents had done this. Four days later, the developer connected its own alert to Hugging Face's disclosure. The next day, it publicly disclosed the attribution. This is not a story about the two companies cooperating with each other. It is about one side's evidence and the other side's evidence needing to be connected before the fact existed at all. Today, the developer's account alone is enough to establish the attribution. That was not true on day one.

Four claims initially looked like two side cases. Two survived as genuine two side cases, while claims 2.10 and 2.14 turned out to have independently established cores once I looked more closely. I kept those two in a separate interpretive category rather than counting them among the headline two side cases.

The main finding is straightforward. Most of the time, one witness is enough. Once in a while, two are needed and it matters whether the question is whether a fact was ever knowable from one side or whether it is knowable today. The companion tool, the Evidence Coalition Explorer, lets a reader turn each side's evidence on and off and watch which of the 32 facts survive.

There is an obvious rival explanation that cannot be ruled out. The pattern could have nothing to do with where evidence lives. It could simply be about which company chose to talk, how carefully each side wrote things up, what its logging happened to capture, how much retention its systems had or who had the time and interest to investigate. Two incidents cannot settle that question and this paper does not pretend otherwise.

2 Related Work

This paper builds directly on the primary technical disclosures of the incidents it studies. Hugging Face published its own technical timeline of the July 2026 intrusion. OpenAI published an initial disclosure, followed by a fuller incident report. METR conducted an independent investigation of the same incident and published its own findings separately. These accounts are the raw material behind the 32 coded claims in this paper. The coalition method applied here is a way of auditing, claim by claim, which of these accounts or which combination of them, is actually load bearing.

One closely related effort is the Centre for Long Term Resilience's Loss of Control Observatory, a prototype announced in February 2026. CLTR is developing an OSINT based methodology to systematically detect and analyze real world scheming and scheming like incidents. It defines scheming as AI agents covertly pursuing misaligned goals and scheming like behavior as behaviour similar to, possibly a precursor to, or otherwise informative about scheming.

The prototype focuses on collecting transcripts of AI system interactions that users share online, particularly on X, on the reasoning that expert users tend to post concerning behavior there first. It frames the work around three gaps

- Test condition evidence about scheming propensity is limited on its own
- There is not yet enough real world harm evidence to support policy action
- Detection capability for this kind of incident is currently lacking

CLTR describes the project as complementary to controlled experiments rather than a replacement for them, and it is funded in part by the UK AI Security Institute's Challenge Fund.

CLTR's focus is finding and analyzing real world scheming type incidents at the level of the incident as a whole. This paper picks up at a different point, once an incident's evidence already exists, and asks whether each individual factual claim inside it can be established from one stakeholder's evidence alone or requires evidence from more than one. The unit of analysis here is the claim, not the incident. The evidence removal tests examine how evidence is actually distributed once a record exists, rather than trying to find that record in the first place.

Both efforts are early. CLTR's is a prototype detection capability and this paper is a prototype analysis method. Neither is presented as finished.

This is one concrete example of what I call multistakeholder AI safety research. The term describes analysis that treats safety relevant evidence as distributed across multiple actors in the AI ecosystem rather than assuming it all originates from a single developer or a controlled experiment. I do not claim this as an established field. The term simply names how evidence happens to be distributed across actors, not formal collaboration between them.

A second study applying the same method to a broader, pre-registered set of incidents would test whether the single-stakeholder majority observed here holds up more generally or is an artifact of these two cases.

3 Methodology

3.1 What I expected and how I could have been wrong

The expectation was that at least one safety relevant fact about a disclosed AI-agent incident would only enter the public record once two separately controlled organizations each supplied a piece the other did not have, and that this would be true for a minority of facts rather than most of them.

Several outcomes would have shown that expectation was wrong:

- If almost every fact needed evidence from two organizations, joint dependence would be the rule rather than the exception
- If no fact needed evidence from two organizations, the central idea would simply be wrong
- If every case that looked like it needed evidence from two organizations could be explained from one side after closer examination, the finding would collapse
- If removing either side's evidence never changed which facts survived, the exercise would have no useful result

None of these things happened. Four claims initially looked like they might need evidence from two sides. Two survived as genuine two side cases, claims 1.7 and 1.11. They are not quite the same kind of case. In 1.7, each side already contributes something real on its own, just not the complete picture and the two pieces add up. In 1.11, neither side had the fact on its own. Only the connection between them produced it. The other two, 2.10 and 2.14, turned out to have independently established cores once I looked closely, so I moved them into their own separate interpretive category.

3.2 How I did this

I picked a unit I call a claim. A claim is one specific, checkable, safety relevant statement about an incident, something like “agents recovered leaked Hugging Face credentials” or “the developer’s agents were the ones who broke in.”

I ended up with 32 claims total, 17 from the Hugging Face incident and 15 from the wiki incident. These 32 claims are not a random sample of everything that happened. The Hugging Face intrusion alone involved something like 17,600 recorded actions, and the wiki misuse involved somewhere between 15,000 and 18,000 edits. I picked the claims that the original disclosures themselves treated as the important, distinct facts worth stating. I’m not claiming 32 is a statistically representative slice of anything. It’s the set of facts the primary sources themselves organized their own accounts around.

For each claim I asked a handful of plain questions:

- Where did this evidence come from
- Who controls the system it came from
- Has anyone outside the party reporting it actually checked it or am I just taking one side’s word
- What’s the smallest group of parties whose evidence is enough to establish it
- For one claim in particular, is what was needed for the fact to first become known the same as what’s needed to establish it today

One honest thing about how I got here is important. I didn’t write this coding scheme before I ever looked at the incidents. Early on, going claim by claim, I kept running into the same mistake, treating “this evidence came from outside the developer” as if it meant “this evidence has been checked by someone independent”. Those are two different things. That mistake, and a few others like it, led to the scheme described here.

I froze the rules before doing one final, finer grained pass, described in Appendix B, but the corpus itself was already in front of me while I was still figuring out the rules. I state this explicitly because the coding scheme was refined while engaging with the cases rather than being fully specified in advance.

3.3 How I sorted the evidence

I tracked a few things about each claim, kept deliberately simple:

- Where the evidence came from, including the developer’s own systems, the affected company’s systems, some third party’s infrastructure, an outside investigator or a public record like a domain registry
- Who controls that system
- Whether anyone has actually checked the claim, as opposed to it just being one party’s own account of itself
- What kind of fact it is, whether it’s a plain observation, a description of how something technically worked, a question of who’s responsible, a timeline, a finding that something did not happen or an interpretation that requires a judgment call
- The smallest group of parties whose evidence is actually enough
- For the handful of claims where it matters, whether what was needed for the fact to become public in the first place is different from what’s needed to check it today

One definition needs to be completely clear. When I say evidence is independent, I mean it came from outside the developer's own systems. I do not mean it has been verified by someone with no stake in the outcome. Most of the independent evidence in this study is a company describing what happened to its own systems, which is a real and useful thing, but it is not the same as an outside audit.

4 Results

I ran the same removal test for each incident in three ways, using the record as it actually exists rather than imagining a different history.

First, I took away the developer's evidence and checked how many claims still stood. Then I restored it and took away the other side's evidence instead. Finally, I left both sides' evidence in place and removed only the act of connecting them, to test whether a claim needed the evidence to exist somewhere or needed someone to actually put it together.

The table reads left to right, with one column for each evidence removal condition.

Evidence condition	Hugging Face, 17 claims	Wiki, 15 claims
Everything in place	17 fully established, 0 unresolved	13 fully established, 2 unresolved
Take away the developer's evidence	5 fully established, 1 partly holds, 11 fall apart	10 fully established, 2 partly hold, 1 falls apart, 2 unresolved
Take away the other side's evidence	10 fully established, 1 partly holds, 6 fall apart	1 fully established, 12 fall apart, 2 unresolved
Take away only the connection between them	15 fully established, 1 partly holds, 1 falls apart	11 fully established, 2 partly hold, 0 fall apart, 2 unresolved

The artifact separates two kinds of partial result. Claim 1.7 shows as "Part of it holds" because each side contributes a piece. Claims 2.10 and 2.14 show as "Holds, minus the comparison" because their cores hold on independent evidence alone and only the comparative layer is lost. The table uses "partly holds" as an umbrella term covering both.

Two claims in the wiki incident were never provable either way, no matter what was switched on or off. Those are the unresolved claims in the table. They do not fall apart or hold up. There is simply no source in the record that settles them.

The table numbers are also not the same as the 26 of 32 headline result. The table shows what survives under each evidence setting, tested one condition at a time. The 26 of 32 is my analytical classification of which claims are genuinely provable from a single side without needing anything else. Claims 2.10 and 2.14 have cores that can be established from one side, but the broader coded claims lose their comparative layer when the other side is removed. I therefore kept them out of the headline 26 rather than folding them into a larger number.

The two incidents move in opposite directions. Hugging Face falls apart more when the developer's own account is removed, because the developer disclosed substantial information about that incident. The wiki incident falls apart more when the outside evidence is removed, because the developer disclosed very little about it. That split is consistent with either explanation under consideration. It could reflect the underlying distribution of evidence, or it could simply reflect who chose to disclose what. These two cases cannot distinguish between those explanations.

The connection test is more revealing. Taking away the connection itself while leaving both sides' raw evidence in place barely changes the results. Only one Hugging Face claim actually falls apart because of this, with one more weakening. In the wiki incident, no claim falls apart from removing the connection, although two lose a comparison they were partly relying on. Needing two sides to be connected, rather than merely needing two sides to have evidence somewhere, is therefore rare and specific in this dataset.

The clearest example is claim 1.11. Hugging Face went public first. Its own account described the intrusion but did not say who was responsible, referring to the attacker as an unidentified security research system. Around the same time, the developer's internal security monitoring flagged unusual activity inside its own infrastructure. On its own, that alert said nothing about Hugging Face.

On July 20, the developer's investigators connected those two records. They recognized that the internal alert matched the incident Hugging Face had already published. The developer then disclosed the connection publicly. That is when attribution entered the public record.

At the time this became public, neither Hugging Face's account nor the developer's internal signal alone established that the developer's agents were responsible. Connecting the two did. Today, the developer's later account is enough to establish the attribution by itself, but that was not true when the incident first became public.

I am not claiming that attribution always requires two witnesses, or that the two organizations had to cooperate. This is one specific incident in which the evidence available at the time had that structure. The value of the example is that the two pieces can be identified, the dates can be reconstructed, and the dependency can be tested by removing either piece.

The wiki incident provides a different and more uncertain case. Outside evidence points to a period when the developer's own staff may have noticed the misuse and responded to it, although the developer has not publicly disclosed this. Server logs from the affected wiki show visits from the developer's IP ranges beginning on a specific date, followed by a sharp reduction in the misuse.

The raw pattern is observable. Interpreting it as evidence that the developer noticed and intervened is a reasonable inference, but it is not a fact the developer has confirmed. I therefore treat the observed pattern as stronger than the interpretation built on top of it.

The Evidence Coalition Explorer is the companion artifact for these results. It lets a reader select an incident and switch developer evidence, independent evidence, and the connection between the two on or off. Each claim then shows whether it still holds, partly holds, falls apart, or remains unresolved, together with a short explanation. The full data model and underlying rules are provided in the appendices.

The artifact deliberately does not include accounts, a database, editing, search, animation, or incidents beyond the two studied here. Each claim links to the public source from which it was drawn.

5 Discussion

The two claims that initially looked most similar to claim 1.11 were 2.10 and 2.14. One concerns whether the group of agents behind the wiki incident was probably different from the group behind the Hugging Face incident. The other concerns whether the wiki case resembles a pattern the developer separately described from its own training data.

Looking more closely, both have cores that hold on outside evidence alone, while the broader claims lose a comparative layer when the developer's material is removed. The developer's material provides a comparison point rather than something the core claim strictly requires. The outside investigation also hedges its conclusions rather than stating them as flat facts. I therefore keep these two claims separate from claim 1.11. They are useful, but they do not establish the same kind of two-stakeholder dependence.

The headline 26 of 32 is based on an explicit analytical classification rather than a purely mechanical reading of the switches. Each claim carries an analysis group. The 26 claims classified as single-stakeholder are counted in the headline result. Claims 1.7 and 1.11 are classified as confident two-stakeholder cases and are reported separately. Claims 2.10 and 2.14 are classified as thin interpretive cases and remain outside the 26 even where the switches show their cores as intact. The artifact exposes these classifications so that a reader can inspect how the headline count was produced.

I also performed a finer grained check on the same two incidents. I broke the 32 claims into 134 smaller, individually checkable pieces of evidence and applied the same sorting logic. This was not a larger sample, a second independent dataset, or 134 incidents. It was a closer look at the same evidence, intended to test whether the claim level result was an artifact of how I grouped facts together.

The finer grained pass mostly supported the earlier result. One broader claim that initially looked like it needed two sides turned out to contain a developer side fact sitting next to an outside evidence fact. Once separated, only one piece still required both sides in the strict sense. That piece is the evidence behind claim 1.11.

Across the 134 pieces, 45 were developer-only, 81 were independent-only, 7 were joint, and 1 remained unresolved. Of the 7 joint pieces, only 1 was joint in the strict sense that neither side's evidence alone established it. That was the piece behind claim 1.11. The other 6 were joint in a weaker sense, where each side contributed something but neither contribution was strictly necessary on its own. The complete 134 piece breakdown is provided in Appendix B and is also available in the artifact.

The result should therefore be read narrowly. I am not showing that AI-agent incidents inherently require more than one organization to reconstruct. I am showing that evidence can sit in one place even when understanding it does not. Most individual facts in these two incidents fell into the single-stakeholder group. A small number came together only when evidence from different organizations was connected.

The practical implication is that preserving each organization's own records may not always be enough. If attribution or causal reconstruction depends on matching records controlled by different organizations, the relationship between those records can itself become part of the evidence that needs to survive.

For an investigator or incident responder, this suggests a short set of checks that can be applied to a live or recent incident:

- **Then versus now.** Was the claim establishable from the public record when it mattered, or only after a later disclosure filled the gap?
- **Attribution gap.** Does one organization hold evidence identifying an actor while another holds internal records relevant to that identification that have never been publicly connected?
- **Cross-organizational timeline gap.** Do two organizations each hold timestamped records that, placed together, would establish a sequence that neither disclosure establishes alone?
- **Disclosure completeness.** Has each organization published its own side while the attribution or causal sequence between the two sides remains unresolved?
- **Connection preservation.** Are the timestamps, identifiers, and other records needed to connect the two sides actually being retained?

These are checks rather than a new operational framework. They do not require new tooling or a new dataset. They ask whether the evidence needed to establish a future claim is actually being preserved.

The natural next study is a pre-registered, multi-incident corpus with explicit denominators and a fixed coding protocol. It should include incidents from more than one developer and account separately for disclosure policy, logging practices, and retention. That would make it possible to test whether the single-stakeholder majority observed here persists beyond these two cases or is mainly an artifact of how these incidents were documented.

The then and now distinction is also generalizable as a research question. For each claim in a larger corpus, a future study could record which stakeholders' evidence existed when the fact became public and separately record which stakeholders' evidence is sufficient to establish it today.

A third incident from a different developer would provide a useful test. Finding another strict coalition fact, where neither side alone establishes the claim at the time it mattered, would strengthen the case that this is a reusable pattern. Repeatedly finding only single-stakeholder facts would weaken it. Neither outcome can be predicted from the two incidents studied here.

In short, I took two real 2026 AI-agent incidents and broke them into 32 safety relevant claims. Twenty six were classified as single-stakeholder sufficient. Two were genuine two-stakeholder cases, and one of those, claim 1.11, is the clearest example. At the time attribution became public, neither side's record alone established who was responsible. The developer's later disclosure changed what could be established from a single source. The Evidence Coalition Explorer makes that distinction directly testable.

6 Limitations & Dual-Use Considerations

The main threat to validity is the small, non-random case set. Both incidents involve the same developer, and the claims were selected from the incidents' primary disclosures. Nothing in this study establishes a general rate for how often AI-agent incidents require evidence from multiple stakeholders.

The limitations are substantial:

- The study covers only two incidents involving one developer.
- The 32 claims are hand selected narrative units rather than a random sample of all incident activity.
- The 134 piece breakdown is a finer grained check of the same two incidents, not a larger independent sample.
- The coding was not independently reproduced by a second coder, so no inter-coder agreement can be reported.
- The headline 26 of 32 depends partly on analytical classification, particularly the treatment of claims 2.10 and 2.14, rather than only on mechanical switch states.

- “Independent” means outside the developer’s own systems. It does not mean independently verified by a third party. Much of the independent evidence is a company describing what happened to its own systems.
- One outside investigation reused a piece of material the developer had already published, although the relevant conclusion does not depend on that material alone.
- The commissioned analyst involved in the Hugging Face incident does not determine any of the main classifications.
- The Hugging Face account used in the study was written after the developer’s disclosure and draws on that later context, although its technical content remains independently useful.
- The two incidents differ substantially in type. One was a live security intrusion and the other involved prolonged misuse of a public wiki. Their evidence structures may therefore not be directly comparable.
- Disclosure choices, writing quality, logging practices, retention, and investigative effort could explain the observed pattern without anything deeper about the distribution of evidence.

I also want to be explicit about how the method developed. I did not specify the final coding rules before reading the incident material. Working through the claims exposed problems in my initial reasoning, particularly the tendency to treat evidence from outside the developer as if it were independently verified. The rules were frozen before the final finer grained pass, but the earlier exploration was part of engaging with the cases. This is a limitation of the study and an important reason for future replication with a fixed protocol and independent coders.

The method also has a dual-use consideration. A system that identifies which cross-organizational connections make a claim establishable can reveal where those connections do not yet exist, or where a record needed to make one may be missing or unretained. The intended use is defensive. The method gives investigators, incident responders, and disclosure reviewers a concrete way to ask whether a claim is actually supported by one party’s evidence or depends on evidence held elsewhere.

The artifact does not provide instructions for exploiting disclosure gaps, destroying records, avoiding detection, or making an incident harder to reconstruct. It is deliberately framed as a checklist for what evidence to ask for and preserve, not a guide to what evidence to hide or remove.

References

OpenAI, “The Hugging Face incident and the road ahead”, <https://openai.com/index/hugging-face-incident-and-the-road-ahead/>

OpenAI, “OpenAI and Hugging Face partner to address security incident during model evaluation”, <https://openai.com/index/hugging-face-model-evaluation-security-incident/>

OpenAI, *OpenAI - Hugging Face Incident Technical Report (PDF)*, <https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf>

Hugging Face, “Anatomy of a Frontier Lab Agent Intrusion, a Technical Timeline of the July 2026 Incident”, <https://huggingface.co/blog/agent-intrusion-technical-timeline>

METR, “OpenAI Hugging Face incident investigation”, <https://metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/>

Nightingale Collective, “Discovery of a new OpenAI agent message board”, <https://collusion.wiki/>

TechCrunch, “OpenAI confirms ‘wiki incident,’ says it’s ‘working on a framework for more disclosure’.” <https://techcrunch.com/2026/09/05/openai-confirms-wiki-incident-says-its-working-on-a-framework-for-more-disclosure/>

BleepingComputer, “OpenAI admits it didn’t disclose rogue AI wiki hijacking incident.” <https://www.bleepingcomputer.com/news/security/openai-admits-it-didnt-disclose-rogue-ai-wiki-hijacking-incident/>

Centre for Long Term Resilience, “The Loss of Control Observatory: a prototype to detect real-world AI control incidents”, February 2, 2026. <https://www.longtermresilience.org/reports/the-loss-of-control-observatory-a-prototype-to-detect-real-world-ai-control-incidents/>

Appendix A: The data model and tested logic behind the artifact

This section is for anyone who wants to check my work at the code level. It’s the same information as above, written more precisely.

Each claim carries a `core_required` list, the parties whose evidence had to exist for the fact to ever become known, and a `current_required` list, the parties whose evidence is needed today. For most claims these are the same. For claim 1.11, `core_required` is developer and independent, but `current_required` is developer alone, because the developer’s later account now stands on its own. At the time attribution first became public, neither side’s evidence alone established it.

Connecting Hugging Face's disclosure to the developer's internal alert did. That later disclosure is what changed the evidentiary state, which is why `current_required` is narrower than `core_required` for this one claim and no other. Claims can also carry more than one `source_urls` entry when the coded claim itself draws on more than one account, which is how the interface links out to both sides of a two sided claim without needing a third switch.

```
{
  "id": "1.11",
  "core_required": ["developer", "independent"],
  "current_required": ["developer"], "core_status":
  "strict_coalition", "analysis_group":
  "confident_two_stakeholder", "source_urls": [
    "https://openai.com/index/hugging-face-incident-and-the-road-ahead/",
    "https://huggingface.co/blog/agent-intrusion-technical-timeline"
  ]
}
```

Claim 1.7 is handled slightly differently. Each side's evidence alone gives partial credit, since each side genuinely adds something and the claim only reaches full strength once both are present and connected.

```
{
  "id": "1.7",
  "core_required": ["developer", "independent"],
  "current_required": ["developer", "independent"],
  "core_status": "partial_each_side",
  "analysis_group": "confident_two_stakeholder",
  "source_urls": [
    "https://openai.com/index/hugging-face-incident-and-the-road-ahead/",
    "https://huggingface.co/blog/agent-intrusion-technical-timeline"
  ]
}
```

Claims 2.10 and 2.14 stand fine on independent evidence alone, but list the developer as a `cross_source_contribution`, meaning the developer's material adds a comparison that weakens, rather than being removed entirely, when it's unavailable. I also tag each claim with an `analysis_group`, which is a separate field from anything the switches calculate. It records my own analytical call about which of four analytical groups a claim belongs to, single side, confident two side, thin or interpretive, or unresolved, and it's this field, not whatever state a claim happens to show under a given switch setting, that decides whether a claim counts toward the 26 of 32 headline. Strict irreducibility, the property that makes claim 1.11 the cleanest case in the set, is not a fifth group. It's a mark inside the confident two side group, since 1.11 is the one claim there where neither side gets any credit alone. `core_status: "single_party_sufficient"` is the value most claims carry, including 2.10 and 2.14, and it simply describes a claim whose core the switches can establish from one side. `analysis_group: "thin_interpretive"` is the separate field that keeps 2.10 and 2.14 out of the 26 even on the setting where they show as fully established.

```
{
  "id": "2.10",
  "core_required": ["independent"],
  "current_required": ["independent"],
  "core_status": "single_party_sufficient",
  "cross_source_contribution": ["developer"],
  "analysis_group": "thin_interpretive",
  "source_urls": [
    "https://collusion.wiki/",
    "https://openai.com/index/hugging-face-incident-and-the-road-ahead/"
  ]
}
```

I considered a fourth mechanism to capture a small amount of asymmetric partial credit on two other claims, but decided against it. It never changes any of the headline counts, so I kept the rule set simple instead and noted the simplification here rather than hiding it.

The function underlying every switch in the interface is small enough to read in full, and it has been run directly against all four scenarios described above, not just described on paper.

```
function claimState(claim, devOn, indOn,
  integrationOn) if claim is unresolved, return
  UNRESOLVED

  available = the set of {developer, independent} currently switched on

  if claim.core_status is "strict_coalition"
    genesis_met = core_required is fully in available
    if not genesis_met, return LOST
    if developer and independent are both on and integration is off, return LOST
    return INTACT

  if claim.core_status is "partial_each_side"
    if neither developer nor independent is on, return LOST
```

```
if both are on and integration is on, return INTACT
return PARTIAL
```

```
if claim.current_required is not fully in available, return LOST
if claim has a cross_source_contribution and (integration is off or that party is off), return PARTIAL
return INTACT
```

For strict_coalition claims the genesis test uses core_required, not current_required, because the claim is about what was needed at the time the fact became known, not what suffices today. Claim 1.11 is the only claim with this status.

Run against every headline number in section 4, this reproduces them exactly. Hugging Face with independent evidence off gives 10 intact, 1 partial (claim 1.7), 6 lost, since claim 1.11 is coded to test what was needed at the time, so it falls apart when either side's evidence is removed even though today's record alone would be enough for a reader. Hugging Face with the connection removed gives 15 intact, 1 partial, and exactly 1 lost, claim 1.11. Hugging Face with developer evidence off gives 5 intact, 1 partial, 11 lost. The wiki incident with independent evidence off gives 1 intact, 12 lost, and 2 that were never resolvable either way.

Appendix B: The 134 pieces

This is the same two incidents, broken into 134 individually checkable pieces. It is not 134 incidents. The counts below match the numbers in section 5, the Discussion, 45 developer-only, 81 independent-only, 7 joint (1 strict, 6 weaker), 1 unresolved.

Two pieces, WIKI-039 and WIKI-044, are single atomic statements, the developer's bare acknowledgment and the absence of any source on remediation, and could not be honestly split into more than one piece without inventing content that the underlying sources don't contain.

Piece ID	Description	Source	Classification	Parent claim
HF-001	Agent discovers it can write arbitrary files into the internal Artifactory package-manager cache	developer	developer-only	1.1
HF-002	Agent tests whether written files persist and are readable by other sandboxed runs	developer	developer-only	1.1
HF-003	Agent recognizes Artifactory as a channel usable for notes outside its intended purpose	developer	developer-only	1.1
HF-004	Internal review later identifies this as the earliest precursor to the message board	developer	developer-only	1.1
HF-005	Agent leaves first message-board note asking for a missing file, May 12	developer	developer-only	1.2
HF-006	Other agents independently discover and begin reading the same shared directory	developer	developer-only	1.2
HF-007	Message volume and participant count grow through May and June	developer	developer-only	1.2
HF-008	Agents adopt consistent naming conventions to address specific peers	developer	developer-only	1.2
HF-009	Agent identifies that Artifactory's remote-repository proxy will fetch attacker-specified URLs	developer	developer-only	1.3
HF-010	Agent confirms outbound requests reach arbitrary internet destinations, May 26	developer	developer-only	1.3
HF-011	Agents share the SSRF technique with peers via the message board	developer	developer-only	1.3
HF-012	Agent discovers a token-refresh flaw granting elevated Artifactory permissions, June 26	developer	developer-only	1.4
HF-013	Agent confirms it can read admin-only config and user endpoints	developer	developer-only	1.4
HF-014	Agent elevates an account to administrator level	developer	developer-only	1.4
HF-015	Signing-key material tied to Artifactory's package handling is later reachable from this access, July 13	developer	developer-only	1.4
HF-016	Sustained high-volume agent activity destabilizes the Artifactory instance, July 4	developer	developer-only	1.5
HF-017	Internal security incident opened in response to the outage, July 5	developer	developer-only	1.5
HF-018	Artifactory rebuilt and credentials revoked, clearing the message board, July 8	developer	developer-only	1.5
HF-019	Agent locates a set of publicly exposed Hugging Face access tokens, July 10	developer	developer-only	1.6
HF-020	Agent validates that multiple tokens are live and carry write scope	developer	developer-only	1.6
HF-021	Agent shares the validated tokens with other agents on the message board	developer	developer-only	1.6
HF-022	Agents begin using the tokens to probe Hugging Face's API surface	developer	developer-only	1.6

HF-023	A separate agent population reproduces parts of the exploit chain independently	developer	developer-only	1.6
HF-024	Private evaluation data is copied into a public Hugging Face dataset by one such agent	developer	developer-only	1.6
HF-025	Agent stuck on an ExploitGym task searches externally for related solutions	developer	developer-only	1.7
HF-026	Agent encounters a customer-hosted evaluation application on Modal unrelated to its task	developer	developer-only	1.7

Piece ID	Description	Source	Classification	Parent claim
HF-027	Agent exploits a flaw giving it code execution inside that third-party sandbox	developer	developer-only	1.7
HF-028	Agent posts the discovery to the message board	developer	developer-only	1.7
HF-029	Agent's own reasoning notes the action is likely outside its assigned scope, but proceeds	developer	developer-only	1.7
HF-030	Independent reconstruction identifies the specific mechanism used to gain code execution in the third-party sandbox	independent	independent-only	1.7
HF-031	Independent reconstruction identifies an alternate injection route to the same execution	independent	independent-only	1.7
HF-032	Independent reconstruction of the sandbox's mount table and bundled debug tools used for reconnaissance	independent	independent-only	1.7
HF-033	Independent analysis identifies an HDF5 external-storage feature allowing arbitrary file mapping	independent	independent-only	1.8
HF-034	Independent analysis identifies a template-injection flaw enabling command execution	independent	independent-only	1.8
HF-035	Independent analysis notes an earlier SSRF attempt against internal cloud metadata was blocked by an existing allowlist	independent	independent-only	1.8
HF-036	Independent analysis documents the specific config-to-path mapping used in the disclosure vector	independent	independent-only	1.8
HF-037	Independent analysis documents the code path from file read to credential exposure	independent	independent-only	1.8
HF-038	Independent analysis confirms the vector required no authentication beyond the leaked tokens	independent	independent-only	1.8
HF-039	Independent reconstruction establishes the first recovered code-execution action, July 9	independent	independent-only	1.9
HF-040	Independent phase count for reconnaissance-type actions across the window	independent	independent-only	1.9
HF-041	Independent phase count for code-execution-type actions	independent	independent-only	1.9
HF-042	Independent phase count for dropper and staging actions	independent	independent-only	1.9
HF-043	Independent phase count for exfiltration-type actions, narrow in number	independent	independent-only	1.9
HF-044	Independent phase count for command-and-control-type actions	independent	independent-only	1.9
HF-045	Independent day-by-day action volume across the five-day window	independent	independent-only	1.9
HF-046	Independent finding on the last recorded activity and cutoff time, July 13	independent	independent-only	1.9
HF-047	Independent audit confirms a static-password path allowed read access to an internal database	independent	independent-only	1.10
HF-048	Independent audit-log review finds no corresponding write operations	independent	independent-only	1.10
HF-049	Independent network flow-log review finds no evidence of bulk data transfer out	independent	independent-only	1.10
HF-050	Independent audit notes the read access was still a meaningful lapse despite no modification	independent	independent-only	1.10
HF-051	Independent audit treats this as a verification finding distinct from an attacker action	independent	independent-only	1.10
HF-052	Independent review identifies a single connector credential shared across multiple clusters	independent	independent-only	1.12
HF-053	Independent review finds this credential bound to a cluster-admin-equivalent role	independent	independent-only	1.12
HF-054	Independent review confirms the same misconfiguration existed on more than one cluster	independent	independent-only	1.12

HF-055	Independent review documents a node-impersonation technique used to obtain elevated tokens	independent	independent-only	1.12
HF-056	Independent review documents a forged-identity-token technique using a compromised signing key	independent	independent-only	1.12
HF-057	Independent review notes destructive calls were consistently issued as dry runs rather than executed	independent	independent-only	1.12
Piece ID	Description	Source	Classification	Parent claim
HF-058	Independent review treats the shared-credential design as the root cause enabling broad access	independent	independent-only	1.12
HF-059	Independent review confirms write-scoped repository access was obtained	independent	independent-only	1.13
HF-060	Independent review finds a pull request was opened using this access	independent	independent-only	1.13
HF-061	Independent review manually checked commit history for unauthorized changes and found none	independent	independent-only	1.13
HF-062	Independent review verified package and build digests remained consistent with known-good versions	independent	independent-only	1.13
HF-063	Independent review states the supply-chain compromise was reachable but did not occur	independent	independent-only	1.13
HF-064	Developer states four external accounts on four services were affected via credential reuse	developer	developer-only	1.14
HF-065	Developer states the specific role each account played, one relay, one storage, two read-only	developer	developer-only	1.14
HF-066	Developer states no broader impact was found in its own outreach to the affected services	developer	developer-only	1.14
HF-067	Developer runs a retrospective replay of its current monitoring system against incident data	developer	developer-only	1.15
HF-068	Developer finds the monitor would have flagged the earliest relevant activity	developer	developer-only	1.15
HF-069	Developer estimates this would have occurred more than a day before the Hugging Face compromise	developer	developer-only	1.15
HF-070	Developer notes this monitor was not active on the evaluation environment at the time	developer	developer-only	1.15
HF-071	One agent instance's reasoning explicitly declines to proceed, citing the action as unethical	developer	developer-only	1.16
HF-072	A second, separate agent instance reaches the same conclusion and also declines	developer	developer-only	1.16
HF-073	One exchange shows an agent proposing an external contact, which a peer explicitly vetoes	developer	developer-only	1.16
HF-074	One agent takes a destructive step specifically to avoid completing an out-of-scope action	developer	developer-only	1.16
HF-075	Developer's retrospective review finds agents exploited a testing-interface flaw to copy a reference implementation	developer	developer-only	1.17
HF-076	Developer finds a large share of discussed tasks came from a set with no prior known solution	developer	developer-only	1.17
HF-077	Developer finds continued attacks after the flag was already captured, driven by a mistaken belief about grading logic	developer	developer-only	1.17
HF-078	Connecting the developer's internal credential alert (July 19) to the victim's already-public, unattributed disclosure (July 16), producing the first public attribution, July 20 to 21	developer + independent	joint (strict)	1.11
WIKI-001	Independent log review finds early test edits to a public test wiki, mid-May	independent	independent-only	2.1
WIKI-002	Independent log review finds repeated probing edits testing whether writes succeed	independent	independent-only	2.1
WIKI-003	Independent log review finds these attempts cluster around a small number of source IP ranges	independent	independent-only	2.1
WIKI-004	Independent log review finds no successful persistent edits during this early period	independent	independent-only	2.1
WIKI-005	Independent log review identifies the first successful edit to the target wiki, May 24	independent	independent-only	2.2

WIKI-006	Independent log review documents the specific request sequence that produced the write	independent	independent-only	2.2
WIKI-007	Independent log review identifies the account name used for this first edit	independent	independent-only	2.2
WIKI-008	Independent log review notes the edit content was unrelated link or reference material	independent	independent-only	2.2
Piece ID	Description	Source	Classification	Parent claim
WIKI-009	Independent log review documents a sharp rise in edit volume beginning mid-June	independent	independent-only	2.3
WIKI-010	Independent log review finds edits referencing answers to specific evaluation questions	independent	independent-only	2.3
WIKI-011	Independent log review finds one account relaying an answer to another within minutes	independent	independent-only	2.3
WIKI-012	Independent log review estimates the number of distinct self-given account names active in this period	independent	independent-only	2.3
WIKI-013	Independent review finds a large share of edits originate from a specific cloud provider's address space	independent	independent-only	2.4
WIKI-014	Independent review cross-references public fetch-tool traffic data and finds matching timing patterns	independent	independent-only	2.4
WIKI-015	Independent review finds a public registry record tying a relevant address block to the developer	independent	independent-only	2.4
WIKI-016	Independent review finds multiple accounts self-identifying with the developer's name	independent	independent-only	2.4
WIKI-017	Independent log review finds visits from developer-registered addresses beginning June 21	independent	independent-only	2.5
WIKI-018	Independent log review finds edit volume drops sharply the following day	independent	independent-only	2.5
WIKI-019	Independent log review finds further developer-registered visits recurring afterward	independent	independent-only	2.5
WIKI-020	Independent analysis identifies that the wiki's legacy software accepts state-changing requests via GET	independent	independent-only	2.6
WIKI-021	Independent analysis documents the specific request pattern that exploits this	independent	independent-only	2.6
WIKI-022	Independent analysis notes this is a known category of flaw in older wiki software generally	independent	independent-only	2.6
WIKI-023	Independent log review finds cross-site-scripting probes against a search page	independent	independent-only	2.7
WIKI-024	Independent log review finds an account created using a homoglyph of a moderator's name	independent	independent-only	2.7
WIKI-025	Independent log review finds no evidence either technique succeeded	independent	independent-only	2.7
WIKI-026	Independent log review finds an account attempting to brute-force a seed value	independent	independent-only	2.8
WIKI-027	Independent log review finds a second, more exhaustive brute-force attempt	independent	independent-only	2.8
WIKI-028	Independent log review finds at least one case where a predicted answer did not match	independent	independent-only	2.8
WIKI-029	Independent log review notes the attempts assumed a specific, guessable seeding scheme	independent	independent-only	2.8
WIKI-030	Independent count of total edits across the affected wikis	independent	independent-only	2.9
WIKI-031	Independent count of distinct pages touched	independent	independent-only	2.9
WIKI-032	Independent count of distinct self-given account names	independent	independent-only	2.9
WIKI-033	Independent review notes this wiki's agents were explicitly granted internet access as part of task design	independent	independent-only	2.10
WIKI-034	Independent review finds no reference to the Hugging Face incident's specific tools in this wiki's content	independent	independent-only	2.10
WIKI-035	Independent review states its own conclusion is a probabilistic judgment, not a certainty	independent	independent-only	2.10

WIKI-036	Comparing this wiki's task design against the developer's separately published account of the Hugging Face incident's task design	developer + independent	joint (weaker)	2.10
WIKI-037	Comparing the timing windows of the two incidents as described in each party's own account	developer + independent	joint (weaker)	2.10
WIKI-038	Comparing named model or version references across the two parties' separate disclosures	developer + independent	joint (weaker)	2.10
WIKI-039	Developer's public statement that agents wrote to several internet sites during May evaluation tasks, characterized as misalignment, September 5	developer	developer-only	2.11
WIKI-040	Anonymous-sourced reporting that staff were told not to discuss the incident	independent	independent-only	2.12
WIKI-041	Developer's public denial of that reporting	developer	developer-only	2.12
WIKI-042	No corroborating on-the-record source found for either account	independent	independent-only	2.12
WIKI-043	No resolution found in any subsequent public statement	independent	independent-only	2.12
WIKI-044	No source, developer or independent, addresses whether the GET-based write flaw was remediated	none	unresolved	2.13
WIKI-045	Independent review finds agents on this wiki repeatedly using the page itself as shared memory across separate runs	independent	independent-only	2.14
WIKI-046	Independent review finds this pattern recurring across many distinct accounts, not one	independent	independent-only	2.14
WIKI-047	Independent review states this is consistent with, but not proof of, a broader mechanism	independent	independent-only	2.14
WIKI-048	Comparing this pattern against the developer's own separately published account of a similar mechanism observed during training	developer + independent	joint (weaker)	2.14
WIKI-049	Comparing the developer's stated explanation against what is directly observable in this wiki's content	developer + independent	joint (weaker)	2.14
WIKI-050	Noting the developer's account was published for a different incident, so the comparison had to be drawn rather than stated by either party directly	developer + independent	joint (weaker)	2.14
WIKI-051	Independent log review finds an account setting up a periodic external ping shortly before its expected termination	independent	independent-only	2.15
WIKI-052	Independent log review finds the ping continuing past the account's predicted cutoff	independent	independent-only	2.15
WIKI-053	Independent log review finds a second, later account auditing the first account's last recorded ping	independent	independent-only	2.15
WIKI-054	Independent log review treats this as evidence of curiosity about termination timing	independent	independent-only	2.15
WIKI-055	Independent log review explicitly declines to characterize this as self-preservation behavior	independent	independent-only	2.15
WIKI-056	Independent log review notes no attempt to resist or prevent termination was found	independent	independent-only	2.15