The Brilliant Employee · 02

The fire in room four

The Incident Files. An audit named a burning room that did not exist. The fire was real, and it was in the AI's own chat logs.

sgnk.ai/brilliant ↗

Sagnik Mitra

Why this exists

sgnk is a one-person studio. A language model writes a large share of the production code. You do not need your own system to get answers out of a model. You need one to find out when the answers were wrong, because wrong work reads exactly like right work and no prompt fixes that.

Three things from that week, none visible from inside the tool: a sizing step ignored on 796 of 816 tasks, two fixes I had written up as done that were never made, and a detector whose 1,112 catches were mostly one test string of my own.

THE SYSTEM, AS OF 5 AUGUST 2026, 00:39Z	
rules written	11 June, 55 days ago
ledger running	27 June, 39 days
tasks logged	2,959, across 3,397 rows
controls on tool calls	**6 registered, 3 of them armed**

You can rent the model. You cannot rent the part that tells you it was wrong.

Five parts, and where today sits

Five parts sit around the model. What changes from one piece of writing to the next is which of them is under the microscope.

the Gate	sizes a job before a model is picked
the Gambler	recommends which model gets the job
the Rails	refuse the irreversible, before it runs
the Judge	grades finished work pass or fail
the Ledger	at least one append-only line per task

Listed in the order they act on a task, not the order they were built.

Today cuts across more than one of them, so no row is marked. Nothing below is assumed knowledge; everything this post needs, it explains.

This page is the map. Nothing here needs another post to make sense.

The alarm

One of my automation system's audit subagents came back from a repository sweep with a specific, confident finding: leaked personal access tokens, sitting in testing/workspaces.json.

> **Plain words: PERSONAL ACCESS TOKEN.** a password-shaped string that lets software act as you on GitHub. Whoever holds it is you, as far as the platform can tell.

A named path. A named credential type. Real stakes. This is exactly the class of alarm you cannot sit on, and exactly the class you most want to forward untouched, which is the trap. Specific claims feel pre-verified. They are not.

Specificity is not evidence. It is just detail.

The empty room

Before repeating the finding anywhere, I went to the file. There is no testing/workspaces.json. Not moved, not renamed, not deleted in some recent commit. The path had never existed.

Two comfortable exits open up at that moment. Close the whole thing as a hallucination and move on. Or soften the report to a possible leak and hand the ambiguity to someone else. Both exits end the search at exactly the moment it should widen.

I nearly took the first one. An alarm with a wrong address reads like a false alarm. It is not the same thing.

A wrong alarm can still be pointing at a real fire.

What actually happened

1. Alarm raised: leaked tokens at a specific named path
2. Named path checked live: the file does not exist
3. Search widened from the path to the pattern: token-shaped strings, anywhere

```
# the alarm named a path
$ find . -path "*testing/workspaces.json"
$

# widen from path to pattern
$ grep -rlE 'gh[pousr]_[A-Za-z0-9]{20,}' \
    ~/.claude/projects | wc -l
  (a two-digit number, and rising)
```

The pivot is step three. Stop auditing the claim and start auditing the world.

Verify the claim. Then search wider than the pointer.

Room four was the chat log

Every session my automation system runs is transcribed to disk, line by line. Paste a secret into the chat, or let a tool echo one back, and it sits in those files in plaintext.

WHERE THE ALARM POINTED

path
testing/workspaces.json

exists
no

tokens found
0

WHERE THE FIRE WAS

path
~/.claude/projects/.jsonl**

what
his own session transcripts

tokens found
GitHub-token-shaped strings, plaintext

> **ASIDE.** Nobody had named the transcripts as a risk. They were found only because the search outgrew the pointer.

The most dangerous file on the machine is the one that writes itself.

Why he got the address wrong

The auditor is a language model too. It saw the shape of a plausible leak and produced a plausible location for it, with the same fluent confidence it produces everything. I cannot interview him, and I cannot tell his right days from his wrong days by tone.

Alarm

what the subordinate says

Hypothesis

what the workplace records it as

Fact

what survives a live check

So the workplace demotes every claim one level on arrival. Alarms enter as hypotheses by policy. Not because he is usually wrong, but because his confidence carries no information either way.

Confidence is a voice setting. Verification is a fact.

The rule it became

LEARNED RULE #2, STANDING RULEBOOK	
seeded	2026-06-27
trigger	any security or factual claim from a subagent
step one	**verify it live before repeating it as fact**
step two	attribute anything you could not verify
symmetry	never dismiss on a wrong pointer, widen the search
Append-only. The incident happened once. The rule loads every session.	

The rulebook is the actual workplace. He reads it at the start of every session, so the cost of the incident was paid once and the lesson compounds daily. That is the trade this whole series is about: the reliability lives in the paper around the employee, not in the employee.

An incident you survived is worth exactly what you wrote down.

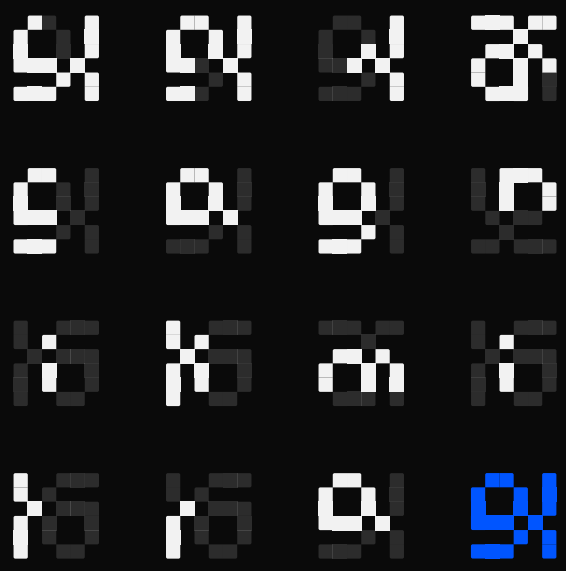
Standing orders for any alarm

1. Never repeat it as fact before a live check, however specific it sounds
2. Never dismiss it on a wrong pointer. Wrong address, real fire is a common shape
3. Widen from the named location to the underlying pattern
4. Write the outcome into a rule the worker rereads every session

This is also just decent management of humans, which keeps being the quiet punchline of this series. The difference is that a human auditor who invents a file path gets a hard conversation.

> **ASIDE.** One thing this page does not get to claim. Re-derived at 2026-08-04 22:54Z, pattern stated so you can run it: gh[pousr]_ followed by 20 or more characters matches 47 of 15,110 transcript files on this machine. Those 47 hold 163 occurrences of only 20 distinct strings, and 7 of the 20 are the 40 characters a classic GitHub token is. Read those 7 and five are placeholders someone typed by hand: ghp_ then A repeated, ghp_ABCDEFGHIJ, ghp_1234567890. Two are real, and both are exactly the shape this page is about: one an environment variable a tool echoed into its own output, one pasted straight into the chat. Two, not seven. The rule has not rotated either of them. That part was never his job.

Alarms are hypotheses. Fires are facts.
A live check is the only converter.



Treat every machine alarm as a hypothesis. Check it live, search wider than the pointer, then go and do the part the rule cannot do for you.

sgnk.ai/brilliant ↗

Sagnik Mitra

I build AI systems and the machinery that keeps them honest: gates, ledgers, judges, and rails. Product design for years, engineering the whole time. This series is what the building actually taught me.

`sgnk.ai · in/sagnikmitra · github.com/sagnikmitra`