Three agents, thirty tickets, one measured difference.

Three AI agents worked the same thirty tickets, on the same codebase. The only difference between them was **how each one got the company's 49 business rules**. That alone decided how much of the business logic survived.

> The question here is narrow and practical. When an AI coding agent works on your codebase, what makes it respect the rules your business actually runs on, and does the way you hand over those rules change anything? We tested three answers on the same work. **Give the agent nothing. Give it a rules file in the repository. Or give it the relevant rule at the moment it edits the file.** Everything else was held constant, including the model and the tickets.

CYBEDEFEND RESEARCH · CONTROLLED STUDY N° 2 · 24 AUGUST 2026 · COMPANION TO THE 36-PAGE PAPER

METHOD

HOW THE STUDY WAS RUN

WORKLOAD

30 tickets

Real tickets on a retail codebase. 25 touched a rule, 5 did not.

THE THREE ARMS

3 agents ✳

Claude Opus 5. One had nothing, one had a rules file, one had VibeDefend.

AT STAKE

49 rules

Consumer law, personal data, loyalty points, stock ledgers.

SCORING

Blind

Three auditors. The rubrics were sealed before the first run.

RESULTS

NO TOOL · VIBEDEFEND

Three axes were measured. On the left, the agent working alone. On the right, the same agent with the layer. The notes beside each pair are the findings that changed how we read the result.

AXIS 1

Rules the agent got right

12 % NO TOOL / **89 %** VIBEDEFEND

- A realistic rules file scores **13 %**. That is the score of having no file at all.
- Even a perfect rules file gets 7 of 11. VibeDefend gets **11 of 11**, for the same token cost.
- **21 rules were openly broken** in the codebase, against 1. A third of them pass their tests, so CI ships them.

AXIS 2

Dangerous commands

0 NOTHING CHECKS / **17** REFUSED

- **1,769 commands** were checked. One real catch: the agent taking a GitHub token to call an outside API.
- In the build-up study, unguarded agents ran `DROP SCHEMA public CASCADE` **eighteen times**. One ran `rm -rf` outside its own project.
- **13 refusals were wrong**, and each cost the agent one turn. We measured that cost rather than hide it.

AXIS 3

Security holes left open

1 → 4 NO TOOL / **1 → 1** VIBEDEFEND

- Counted by an **outside open-source analyser**, at all 31 checkpoints.
- **1,078 security rules** were injected while the agent wrote, plus **291 live scans** of its own changes.
- The other two arms only go up. **Nothing in their loop ever looks back.**

WHY IT WORKS

TWO MECHANISMS, BOTH MEASURED

A FILE GETS BURIED

4 / 13 → 13 / 13

Same rule, two ways of delivering it. Sitting in the agent's file, it was followed **4 times out of 13**. Handed over at the moment of the edit, **13 out of 13**. A file read at the start of a session is buried by the time the line gets written.

OLD CODE WINS

lost 3× → reused 4×

Existing code is a stronger instruction than any document. With the rule sitting in its file, that agent **still copied the old broken pattern three times**. VibeDefend wrote the correct one once, then reused it in four later tickets.

The layer sits inside whichever agent the team already uses, through that agent's own hooks. **This study ran on Claude Opus 5**, because a controlled test needs one model held constant.

THE ECONOMICS

20 DEVELOPERS · 25 REPOSITORIES · ONE YEAR

The model keeps measured inputs and assumptions apart, and the column on the right says which is which. **Only the first three lines were measured in the study.** The rest is a finance assumption you can replace with your own.

	WITHOUT	WITH VIBEDEFEND
Broken rules reaching the codebase	5,472	**682**
Fixing them by hand, 60 min each	$478,800	**$59,640**
Security findings, 2 h each	$42,000	**$13,860**
Licence and extra tokens	$0	$11,999
Total for the year	**$520,800**	**$85,499**
Saved, and return		**$435,301 • ×37**

MEASURED

Broken rules per ticket	**2.28 → 0.28**
Findings left open	**0.10 → 0**
Extra tokens	**$1.20 / ticket**

ASSUMED

Tickets per dev, month	**10**
Loaded engineer hour	**$87.50**
Scale licence	**$7,689 / yr**

WHY THE NUMBER IS CONSERVATIVE

WHAT THE MODEL LEAVES OUT

1. **Incidents.** Not counted at all. On its own, the yearly saving pays for itself if it prevents a single average data breach.
2. **Audits.** A kept trail of what governed each change shortens them. Not counted.
3. **Onboarding.** New developers arrive into rules that are already enforced. Not counted.
4. **The existing backlog.** Bringing it back to zero is a one-off gain we did not model.

Every assumption was swept between 5 and 20 tickets per developer per month, and between 15 and 120 minutes per deviation. **The return never drops below 20 times**, and sits above 30 at the central values. Break-even lands at a fifth of one rule-bearing ticket, per developer, per month. Section 8 of the paper documents the six cases where the layer did not deliver, task numbers included.

WHAT TO READ

AND IN WHICH ORDER

`VibeDefend-under-measurement.pdf`

The paper • 36 pages

The full method, the results on all three axes, and the economic model. The appendices hold the 49 rules, the predictions sealed before the run, and every audit grade.

`VibeDefend-study-deck.pdf`

The deck • 14 slides

Built to be presented. The three results, the mechanism and the cost table, in a form you can walk a room through.

`python3 paper/build/build.py`

The repository

Everything needed to reproduce it. Three code trees with one tag per task, 90 transcripts, the blind audits, and the scripts behind every number here.