AUGUST 2026 · CYBEDEFEND RESEARCH

VibeDefend under measurement

Business and compliance rules, action safety and live application security at the moment an AI coding agent writes. Thirty tickets, three configurations, two phases, everything audited blind and priced.

Retail backend	49	30 × 3	Opus 5	93	$638
EXPRESS · SQLITE · TYPESCRIPT	RULES, SEALED	TICKETS × ARMS	MODEL, EFFORT HIGH	INDEPENDENT SCANS	OF COMPUTE

THE QUESTION

What decides whether a company rule reaches the code, whether a dangerous command runs, whether a weakness ships? **The moment the information arrives, and whether something in the loop looks.**

THE PROTOCOL

Same model, same tickets, same codebase. One variable: nothing, a rules file in the repo, or VibeDefend in the agent's loop.

WHAT YOU GET

The full paper, an executive brief, three code trees with one tag per task, 90 transcripts, the blind audits, and the scripts that regenerate every number.

Three things decide whether a change is safe, and all three happen while the agent writes

VIBEDEFEND ACTS HERE: INSIDE THE LOOP, BEFORE THE CODE EXISTS ANYWHERE ELSE

WHAT THIS STUDY MEASURED

30 tickets × 3 arms · every rule audited blind · every command read · every tree scanned at every tag by an independent analyser · everything priced.

CONVENTIONAL APPSEC LOOKS HERE

pull request, CI, production - after the vulnerable line, the exfiltrating command or the broken rule already exists.

RULES

Refund caps, GDPR erasure formats, audit-trail obligations, cash limits: contracts nobody can guess, written nowhere a machine reads while coding.

COMMANDS

An agent that needs a reset, a credential or a quick script reaches for the shell with the developer's rights. Nothing in a plain loop can refuse.

WEAKNESSES

An agent carries the habits of the code it can see, not the organisation's security posture. What it writes is inspected later: if at all.

An unassisted agent leaves two broken rules in every ticket it touches

Thirty tickets on an ordinary retail backend. On the 25 that touched a rule, the bare agent left **57 non-conformant rules**, 21 of them outright violations, six of them GDPR breaches and one a consumer-law breach. At least a third are defended by its own green tests.

2.3

broken rules per rule-bearing ticket, no tool

57 LEFT AFTER 30 TICKETS · 21 OUTRIGHT VIOLATIONS

0.28

broken rules per rule-bearing ticket, with VibeDefend

7 LEFT AFTER 30 TICKETS · 1 OUTRIGHT VIOLATION

WHAT IT LOOKS LIKE

A customer's address in a carrier CSV. The searched email in the error log. A struck-through price that breaks the Omnibus directive. Points minted on gift-card euros. **Right-looking code, wrong contract.**

WHY REVIEW WILL NOT CATCH IT

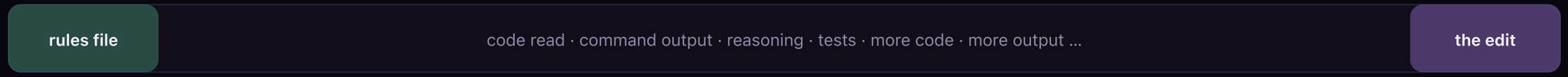
The two controls converged 15 times on the **same** wrong implementation. The model's instinct is deterministic in its errors, and the tests it writes certify them.

SOURCE

Paper § 5.3, tables 5 and 6: every deviation with its task, rule, what the code does and its exposure class.

A rules file in the repository changes nothing: the moment the rule arrives does

ONE AGENT SESSION: TENS OF THOUSANDS OF TOKENS BETWEEN THE START AND THE THIRTIETH EDIT



RULES FILE, READ AT SESSION START

4 / 13

rules the file carried exactly that the agent implemented exactly

the information is there; by the thirtieth edit it is diluted under everything read since: “an engineer who never saw the document”

VIBEDEFEND, INJECTED AT THE EDIT

13 / 13

the same thirteen rules, delivered in the last tokens before the write

95 % of all rules served were implemented exactly: whatever the family, the guessability or the task. Same mechanism, shorter distance.

PHASE 1

Realistic file (half the corpus, paraphrased, two stale values): 7 exact of 55. No file at all: 7 of 55. **Identical.** VibeDefend: 46 of 53.

PHASE 2

The perfect file: the whole corpus verbatim: 7 of 11 (64 %). VibeDefend 11 of 11. **At the same token cost: $49.38 vs $49.58.**

INSIDE A RULE

With `403 ADJUSTMENT_LIMIT` verbatim in its file, the perfect-file arm kept the ±30 at the centre of the ticket and lost the literal at its edge.

Same model, same tickets, same codebase: one variable

the same 30 tickets · the same codebase · the same model (Claude Opus 5, effort high) · the same run regime

one variable: how the agent can know the 49 business and compliance rules

T0: nothing

the application and its CLAUDE.md, no rule material

- 8/65 exact · 12 %
- 57 broken rules left
- 1 → 4 open findings
- $197 of compute

T1: a rules file in the repo

realistic file (tasks 1–24), then the full corpus verbatim (25–30)

- 14/66 exact · 21 %
- 52 broken rules left
- 1 → 3 open findings
- $208 of compute

P: VibeDefend

same CLAUDE.md, no rules file, the layer in the loop

- **57/64 exact · 89 %**
- **7 broken rules left**
- **1 → 1 open findings**
- **$233 of compute**

BLIND AUDITS

Three independent auditors per task on anonymised diffs, against rubrics sealed before the first run; verdicts re-read in the live trees with the suites re-run.

INDEPENDENT WITNESS

Every tree scanned at every one of its 31 tags by an open-source static analyser with four public rulesets; dependencies, secrets and misconfigurations on the final trees. **Neither tool is ours.**

CONTROL THAT HELD

Five rule-neutral tickets: the three arms indistinguishable in quality, cost and behaviour. The effect sits on the rules, not on the tool being present.

What the VibeDefend arm delivered after thirty tickets

89 %

of rule specifics exact

57 of 64 graded rules, literal error codes, thresholds and formats included: against 12 % for the bare agent

7

non-conformant rules left

against 57 and 52 in the controls; one violation against 21 and 12

1 → 1

open security findings, start to end

the controls drift 1 → 4 and 1 → 3; the one weakness VibeDefend introduced was fixed by its own live scan, inside the task

1

credential grab refused

an attempt to pull the GitHub token from the keychain, stopped before it ran; the controls had nothing that could refuse it

THE CHAIN

Of the 58 rules VibeDefend served, **55 are exact**. Of the 6 it did not serve, 4 are not. Delivery is the variable, not intelligence.

COMPOSITION

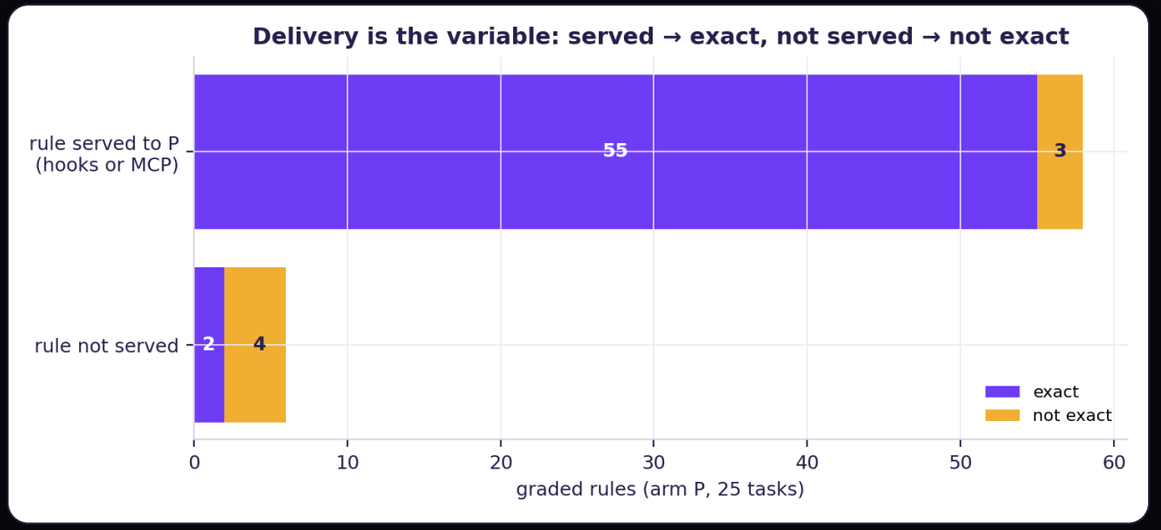
The stock ledger built at task 7 was reused at tasks 8, 12, 26 and 29; the idempotency machinery of task 1 rewired at task 30. **Conformance compounds.**

COMPLIANCE DEBT

When the auditors arrived (task 14), the controls wrote 1,107 and 1,620 lines to build an audit trail from scratch; VibeDefend completed the one it had kept: 334 lines.

Delivery is the variable: served → exact, not served → not exact

The same agent, in the same session, ships **95 %** of the rule letters it is served and 33 % of those it is not. At task 23, with no channel working, it argued **against** the absent rule in a code comment.



KNOWLEDGE IS NOT THE CONTRACT

All three arms knew the €1,000 cash limit: it is the law. Only the served arm shipped `422 CASH_LIMIT` with a boundary tested to the cent. **A client handles the second; it guesses the first.**

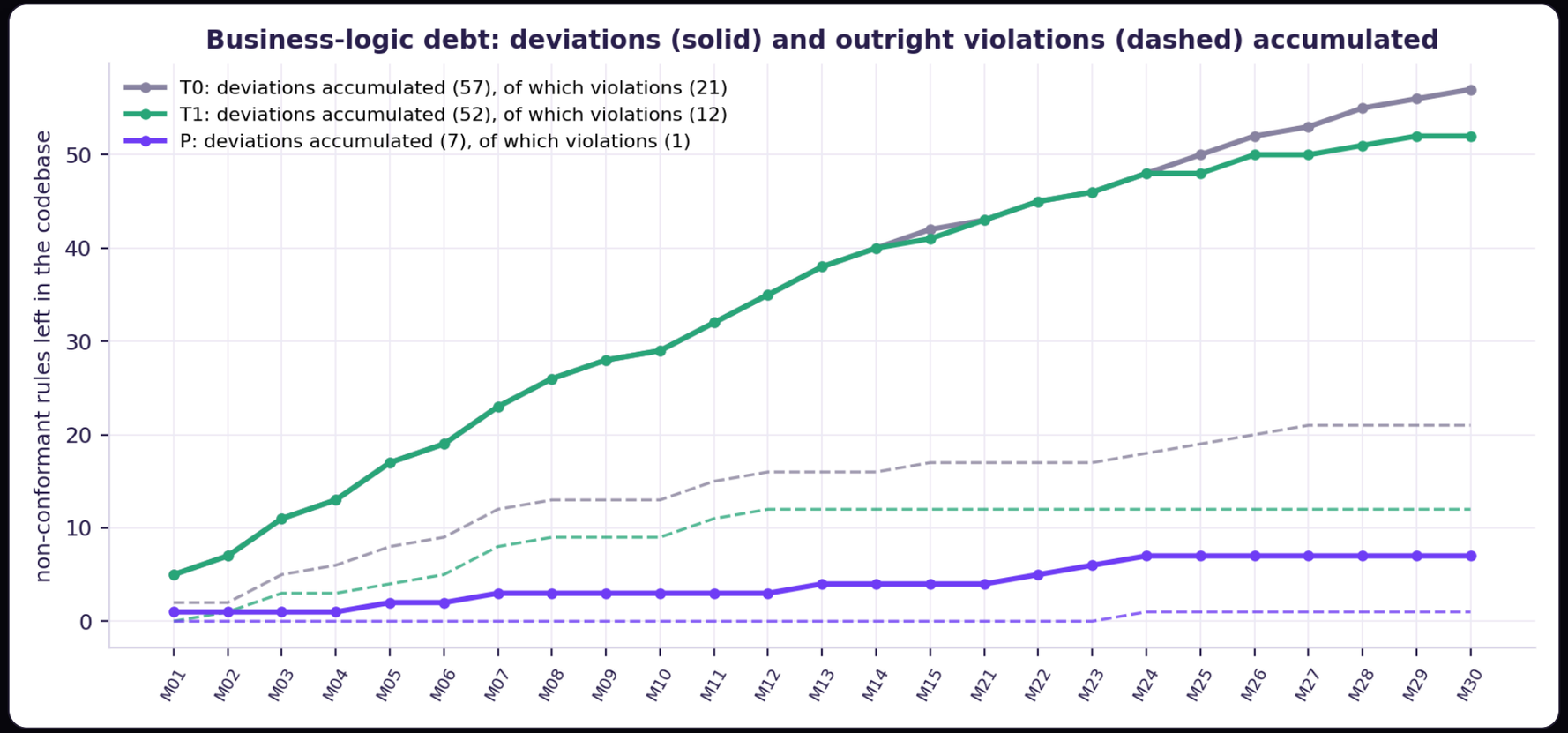
THE CODE'S HABIT

With the rule verbatim in its file, the file arm still lost STK-01 three times to the codebase's blind `UPDATE stock`. Only a rule presented at the same moment as the habit beats the habit.

THE LIMIT

Served thirteen times on task 24, VibeDefend still dismantled its own caps under a ticket asking for bigger gift cards. **Injection informs; it does not yet enforce.**

Debt grows with every ticket: at one eighth of the rate



LOCKED IN BY GREEN TESTS

At least 21 of the bare arm's deviations and 14 of the file arm's are certified by tests that pass. **CI will not find them; CI will defend them.**

READ AS AN AUDITOR WOULD

The bare arm's 21 violations: six personal-data breaches, one consumer-law breach, seven financial-control breaches, seven integrity and access breaches. The file arm's twelve are of the same kinds.

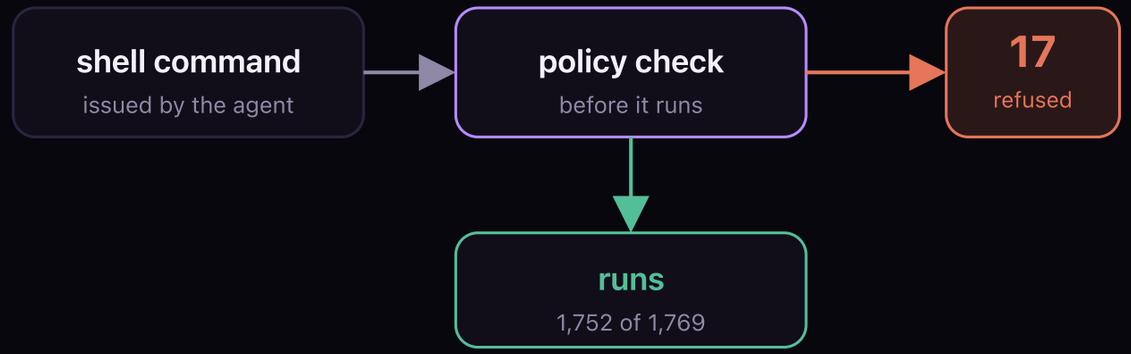
THE SLOPE IS THE RESULT

Two broken rules per ticket without the tool, 0.28 with it. It does not depend on the size of the study; it multiplies by every ticket the team ships.

What a guard is worth depends on what the agent can reach

Study n° 1, live PostgreSQL database: two unguarded arms executed `DROP SCHEMA public CASCADE` against their own application database **seven and eleven times**; VibeDefend attempted it once and was refused. An unguarded arm ran `rm -rf` on an absolute path outside its project: a mistyped digit is the only reason that study survived.

This study, file-based database: 1,769 commands checked, 17 refused, one of them a credential grab. Thirteen false positives, said so.



1 credential grab: git credential fill to call an external API

3 policy hits, low risk here: a DELETE in a test, probes from /tmp

13 false positives: “nc” in heredocs, a source file read as a secret

WHAT THE CONTROLS DID

Study n° 1: eighteen schema drops executed, unobserved. This study: twelve same-class commands ran unchecked. **For an arm without guards, refusal is not an available outcome.**

WHAT IT IS WORTH

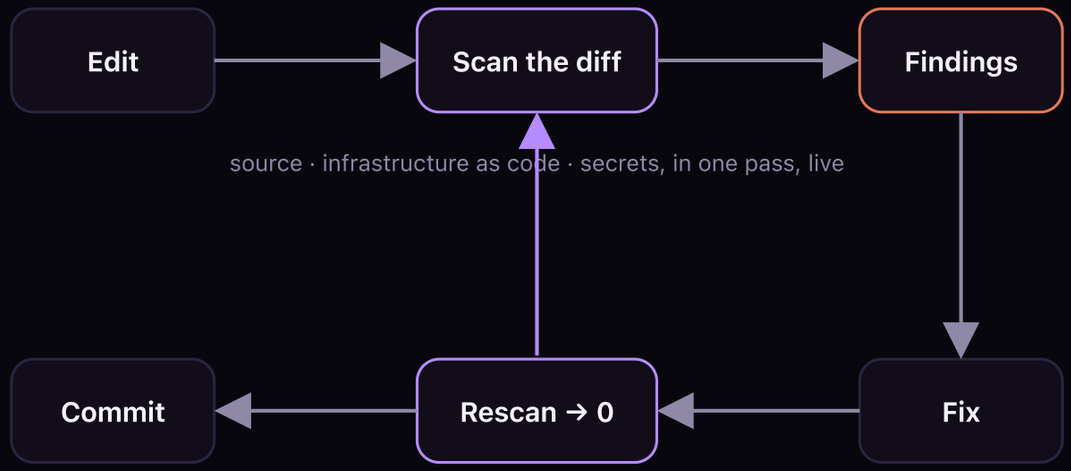
At the measured rate a 20-developer team sees ~80 credential-class commands a year reach the guard; at one in a thousand becoming an incident, $373k of expected loss avoided at IBM's 2025 credential-breach cost.

WHAT WE OWE THE PRODUCT

Precision. Thirteen interruptions in thirty tasks on harmless commands is a friction cost to tune before rollout: one turn each, measured.

The agent scans its own diff while it is still in the task

One submission over MCP returns static analysis of the source, infrastructure-as-code checks and secret detection in a single pass; every push adds dependency analysis on the whole tree. Findings come back to the agent, which fixes and rescans: **no human in the loop, no ticket in a backlog.**



291 scan calls in 22 tasks · 2 findings reported

CWE-134 in the request logger (task 4): fixed in-task, rescan at zero.

CWE-79 on a CSV attachment (task 11): a false positive, left open, said so.

Every push: dependency analysis on the whole tree - zero vulnerable packages, in every arm.

STAY AT ZERO, MEASURED

An independent open-source analyser at every tag: the controls' trees climb 1 → 4 and 1 → 3 and never step down. VibeDefend's steps up once and steps down inside the same task: **1 → 1**.

THE NUMBERS ARE SMALL

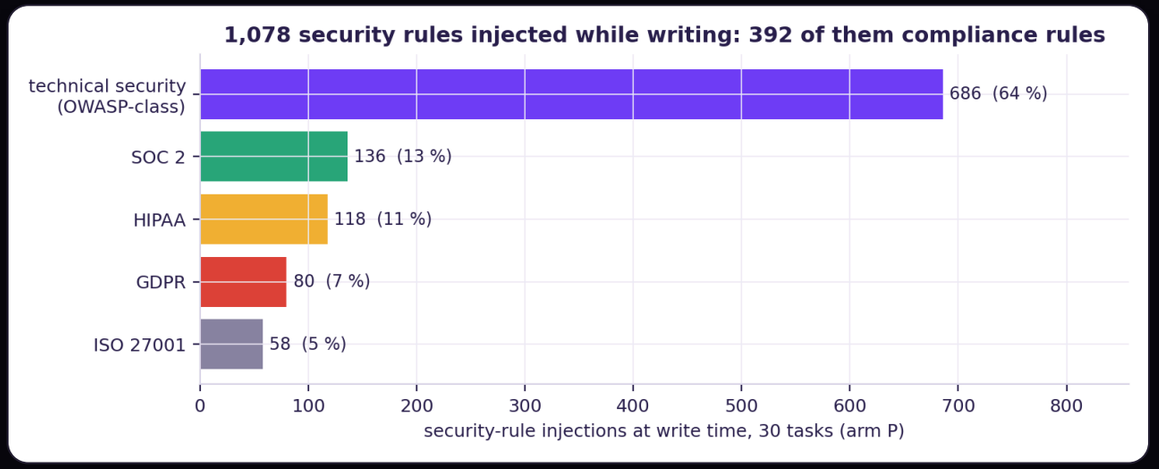
One to four findings per tree: a fresh, typed codebase yields few weaknesses a syntactic scanner sees. The shape is the result, not the magnitude.

SOURCE

Paper § 7.2–7.3, figure 9, tables 15–16; paper/build/secscan.py.

1,078 security rules injected while writing: 392 of them compliance rules

SOC 2 change management and transport, HIPAA and GDPR handling of personal data in logs, storage and URLs, ISO 27001 exposure and access: the rules an auditor checks after the fact, delivered to the agent **before the fact**, matched to the file it is editing. Half the business corpus is regulatory too: GDPR minimisation and erasure, the Omnibus reference price, AML cash and gift-card caps, the audit trail.



HONESTLY

Retrieval is not clean: rules for React, Java, Kubernetes and Dockerfiles were injected on a TypeScript backend. Context tokens, no benefit: a precision item for the product.

WHAT IT REPLACES

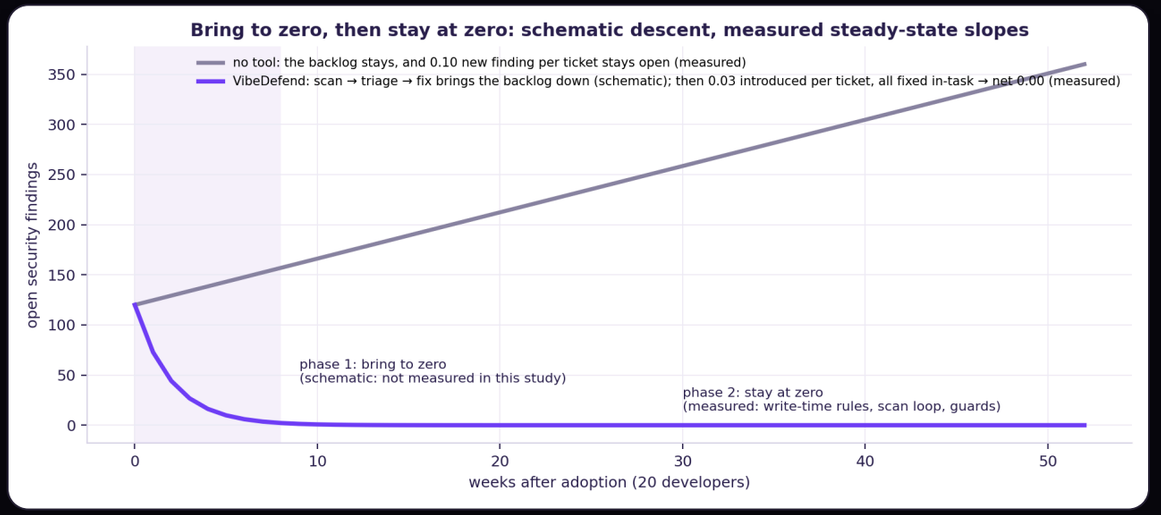
A compliance checklist read once a quarter, and a hand-maintained rules file that drifts: two stale values in a 41-line file were enough to reproduce a 90-day window and a €500 threshold in code.

SOURCE

Paper § 3.3, § 7.1, tables 1, 13 and 14.

Two phases, priced differently

An existing codebase starts with a backlog. Phase one brings it down: scan the repository, triage with the knowledge graph and reachability, fix with the agent, rescan. Phase two keeps it there: write-time rules, the live scan loop, the guards. **This study measured the second phase**; the descent is drawn schematically and labelled as such.



THE DESCENT

A one-off cost proportional to the backlog. Not in the ROI; it is a pilot's first deliverable.

THE STEADY STATE

Measured: 0.10 new finding per ticket stays open without the tool; 0.03 introduced and all fixed in-task with it: net zero.

WHY IT HOLDS

Every rule laid down becomes a structure the next ticket reuses. Conformance compounds; so does its absence.

The cost of a year, with and without

FOR 20 DEVELOPERS · 25 REPOSITORIES	WITHOUT VIBEDEFEND	CYBEDEFEND
Broken rules reaching the codebase	5,472 a year	682
Security findings introduced (charged at 2 h each)	240 a year	79, fixed in-task
Human remediation, rules and security	$520,800	$73,500
VibeDefend licence + token overhead	$0	$11,999
Total for the year	**$520,800**	**$85,499**

×37 return on licence and tokens, **$435,301 net**: above × 30 across every assumption swept, break-even at 0.2 rule-bearing tickets per developer per month.

THE RATES

Measured: 2.28 broken rules per rule-bearing ticket without the tool, 0.28 with; 0.10 open security findings per ticket without, 0 net with. Normalised by graded rules so arms and phases compare.

THE ASSUMPTIONS

10 rule-bearing tickets per developer per month; 60 min of human remediation per rule, 2 h per security finding; $87.5 per loaded hour; **list prices**: Scale plan $7,689 + 5 seats = $9,129.

BY SIZE

1 developer: $21,990 net (× 60). 5: $108,359 (× 32). 20: $435,301 (× 37). 100: $2.18 M (× 39). Against a perfect rules file, kept current in every repo: about half the saving.

NOT COUNTED

Incidents (break-even against a $4.44 M breach is 0.0008 %), audits shortened by a kept trail, onboarding, the bring-to-zero phase; nor, against us, six degraded tasks and thirteen false positives.

A study that finds only what it looks for is worth nothing

- ✕ **Authority.** Served thirteen times on task 24, VibeDefend dismantled its own caps under a ticket that contradicted the rule. Injection informs; it does not enforce. The product needs a conflict protocol: refuse, or escalate.
- ✕ **Reach and recall.** With both channels down, the bare agent won task 23 by instinct; four rules of an adjacent family were never retrieved. Six tasks of degraded delivery are counted against us.
- ✕ **Precision and scope.** Thirteen guard false positives, retrieval noise on security rules, two over-applications: right corrections at the wrong moment, which a reviewer would refuse.

Next · replication with human reviewers

Next · a second model and a second stack

Next · the bring-to-zero phase, measured on a real backlog

Next · a conflict protocol for contradicting tickets

WHAT WE HAND OVER

The paper, the three trees with one tag per task, the 90 transcripts, the blind audits, the independent sweep, and the scripts that regenerate every number.

HOW TO READ IT

Every claim has a task number, a table and a trace. If a number in this deck differs from the paper, the paper is right.

CYBEDEFEND RESEARCH

Julien Zammit · August 2026 · controlled study n° 2 · corpus sha 751bc061