PART OF THE KAI AUTONOMOUS DEFENSE PLATFORM

Autonomous Exposure Remediation

The fix authored. The fix executed.

Most tools stop at the ticket, which is exactly where the delay starts. Kai writes the change itself, and where a maintenance window blocks the patch, specifies the configuration change to the controls you already run instead.

16%	of organizations close a critical vulnerability within three days
57%	run less than half of their remediation workflows through any automation
32%	already permit automated remediation actions without human approval

Reported by 500 chief information security officers at enterprises above $500 million in revenue.

01 · THE PROBLEM

A ticket is a request, not a change

Patch orchestration has been solved for a decade, provided somebody writes the plan: which version, what goes first, what depends on it, and whether the host comes back up.

So the finding goes out as a ticket, to a team that rebuilds the plan from scratch, in a window it has to find, for an asset it may not own. That is where the weeks go.

02 · THE SOLUTION

The change, authored and staged for its owner

Kai turns each validated exposure into the specific change and routes it to the accountable owner, inside the boundary you set.

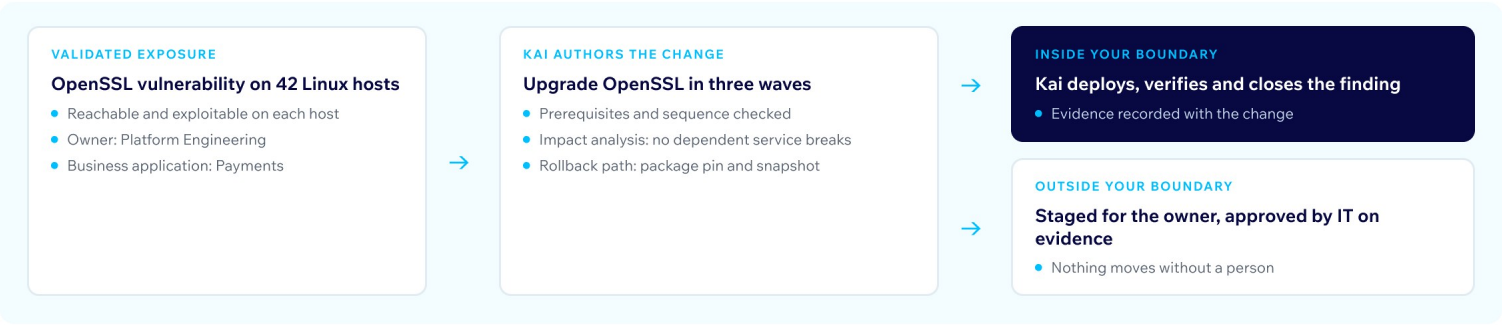
03 · USE CASES

Automated remediation
Inside your boundary Kai writes, validates and deploys the change. Outside it, the change is staged with owner, sequencing and rollback.
Exposures close without adding headcount.

Predictive remediation impact analysis
Kai predicts whether a fix will break dependent systems before it runs.
IT approves a staged change with a rollback path, on evidence.

04 · HOW KAI DOES IT

One validated exposure, two paths to closed



How a change moves in Kai: authored once, deployed inside your boundary or staged for the owner outside it. Sample data.

05 · OUTCOMES

Inside your boundary
you decide where Kai may act, and the boundary is enforced rather than advisory

Rollback recorded
for every change Kai executes, alongside the evidence that justified it

One record
so the change resolves against the same asset and owner as everything else

What you can trust, and what you can hand your CISO

Every change carries its evidence
The rollback path and the proof that justified it are recorded with the change, not in somebody's memory.

Every removal carries a reason
A finding cut from the queue carries a written reason you can defend in an audit.

The boundary is enforced
You set where Kai may act, per environment, asset class and change type. Outside it, nothing moves without a person.

Nothing deployed to start
Agentless and read only against the sources you already run. SOC 2 Type II. Nothing is replaced.

06 · IN PRACTICE

The only handoff is the change

Security owns the workflow from the first exposure to the planned change. IT and Engineering own the change itself, staged for a person or executed by Kai inside the boundary they set.

Autonomous Exposure Remediation

CVE-2024-6387

From validated finding to landed change

VALIDATED FINDING

Remote code execution

openssh-server 9.6p1 (Ubuntu)

- Exploitable: yes
- Reachable: yes
- Owner: Infrastructure team

CHANGE AUTHORED BY KAI

Upgrade 9.6p1 → 9.8p1

Order: drain · apt upgrade · restart sshd
Prerequisite: package mirror reachable
Blast radius: 14 hosts, labeled
Rollback: snapshot + prior package
Window: 08 Jun, 01:00 to 03:00

YOUR ACTION BOUNDARY

INSIDE
Kai executes, audit trail

OUTSIDE
Staged, owner named

THE SAME FINDING, TWO PATHS



WHEN THE PATCH IS BLOCKED

Kai specifies the compensating change to the controls you already run and confirms a detection covers the gap until the change lands.

> Kai is not an incremental improvement. For the first time, I am looking at something that does not ask my team to do more. It asks them to do less and get better outcomes. That is a completely different conversation.
>
> SECURITY LEADER, KAI CUSTOMER

07 · WHAT KAI DOES

What Kai does that an orchestrator cannot

01 Author only what is proven and owned.
Changes are written only for findings validated on the asset and resolved to an owner.

02 Write the plan, not the request.
Target version, ordering, prerequisites and rollback per asset. What orchestrators need.

03 Label the blast radius.
Patch reliability, rollback and dependent services stated up front, never assumed safe.

04 Shape to the owner's cadence.
Fitted to the window the receiving team operates, where weeks are usually lost.

05 Enforce the boundary.
Inside it Kai executes and records the trail. Outside it, the change waits for a person.

06 Cover the interval.
When the patch is blocked, Kai specifies the compensating control and keeps a detection on.

Pick a finding. We will write the change.

Give us one open critical in your environment. We will come back with the specific fix, its sequencing and prerequisites, the owner it routes to, and the window it fits.

Nothing runs outside your boundary. Rollback path recorded. Your orchestrator stays. kai.security

Get a demo