FINDING BUGS IS EASY, PATCHING ISN'T!

Build Your Own Remediation Pipeline



Mohankumar Vengatachalam

Naveen Mahavishnu

2026 CORNCON CYBERSECURITY CONFERENCE

Who we are



Naveen Mahavishnu

CTO, Aira Security

- 10+ years in AppSec and AI security
- Five patents in Agent Security
- OWASP MCP contributor



Mohan Kumar

CEO, Aira Security

- 15+ years in AppSec and InfraSec
- Earned Bug bounties in Google, Netflix and Spotify
- Won NorthSec, the largest in-person CTF

Contents

- Discovery Scaled. Remediation Didn’t
- SPAR
- Demo
  - Scan
  - Prioritize
  - Auto-fix
  - Rewrite
  - Payoff
- What to build first

Discovery Scaled. Remediation Didn’t!

Anthropic's Project Glasswing

Dashboard of Open-source Vulnerabilities

22K+

FLAGGED BY MYTHOS

all severities

→

1,596

REPORTED TO MAINTAINERS

7% of flagged

→

97

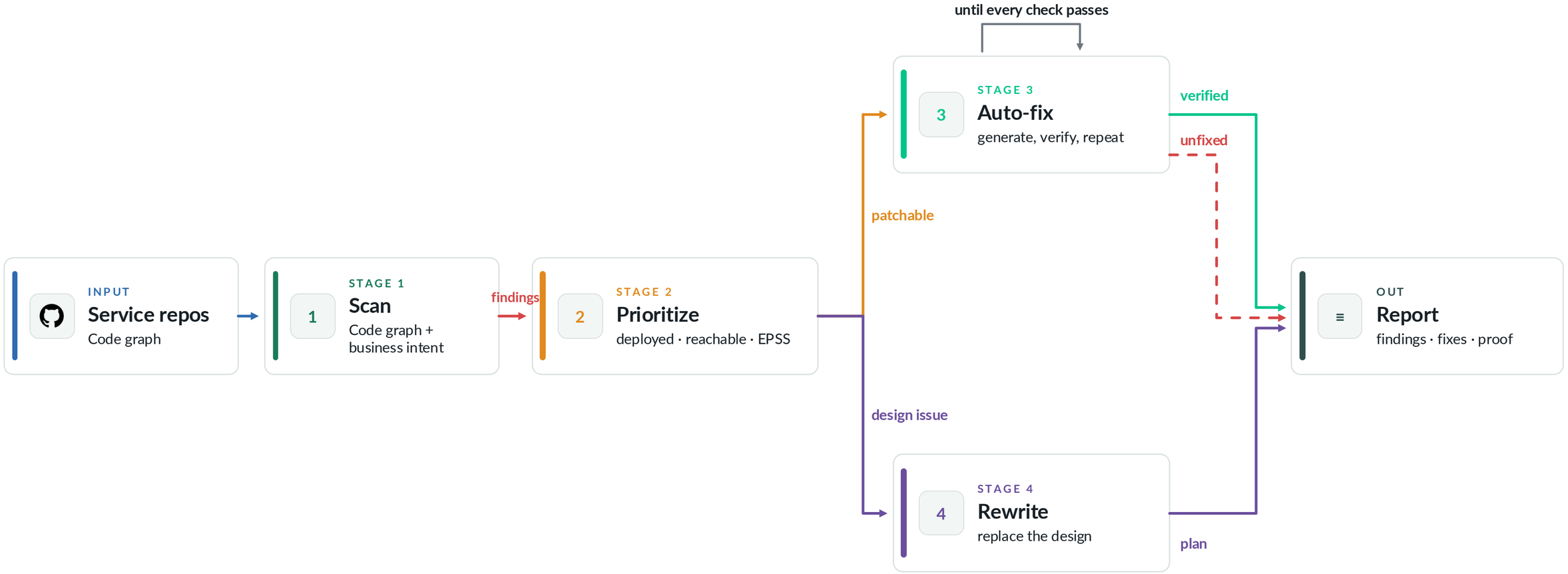
PATCHED

0.4% of flagged · 6% of reported

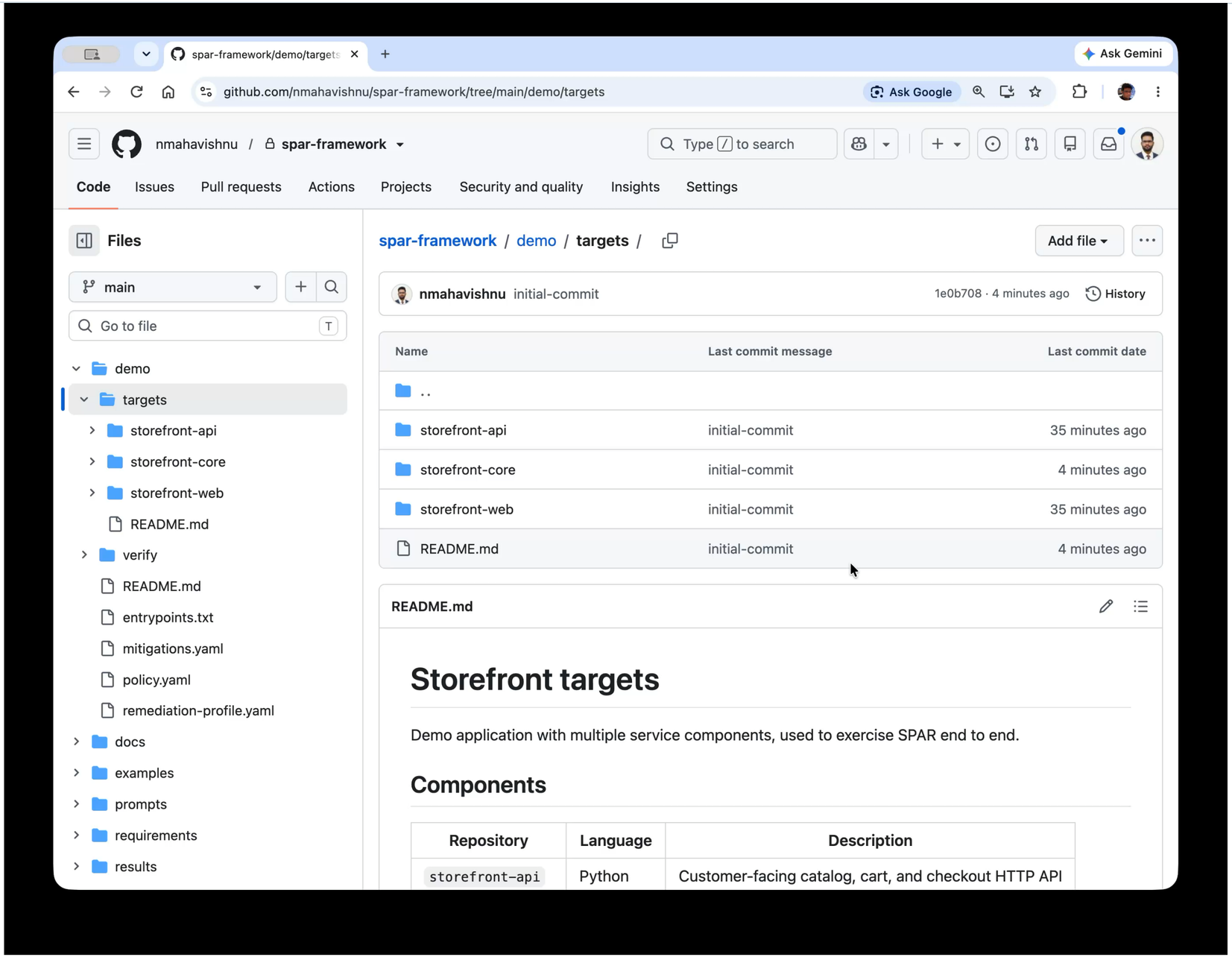
Maintainers have asked Anthropic to **slow down.**

SPAR, end to end

Every finding gets an explicit outcome



Demo


spar-framework/demo/targets
Ask Gemini
github.com/nmahavishnu/spar-framework/tree/main/demo/targets
Ask Google
nmahavishnu / spar-framework
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security and quality
Insights
Settings
Files
main
Go to file T
demo
targets
storefront-api
storefront-core
storefront-web
README.md
verify
README.md
entrypoints.txt
mitigations.yaml
policy.yaml
remediation-profile.yaml
docs
examples
prompts
requirements
results
spar-framework / demo / targets /
Add file
nmahavishnu initial-commit
1e0b708 · 4 minutes ago
History
Name
Last commit message
Last commit date
..
storefront-api initial-commit 35 minutes ago
storefront-core initial-commit 4 minutes ago
storefront-web initial-commit 35 minutes ago
README.md initial-commit 4 minutes ago
README.md
Storefront targets
Demo application with multiple service components, used to exercise SPAR end to end.
Components
Repository
Language
Description
storefront-api
Python
Customer-facing catalog, cart, and checkout HTTP API

Scan: the findings pile

78 findings

```
==> Service details
 SERVICE DETAILS
 Target           /Users/naveen/spar-framework/demo
 Components       3
 storefront-api   Customer-facing catalog, cart, and checkout HTTP API
 storefront-web   Browser storefront and account UI
 storefront-core  Shared pricing, inventory, and order-domain library


==> Scanning Storefront
 SCAN COMPLETE
 Duration  41.2s
 Findings  78
 Severity  critical 8  ·  high 31  ·  medium 28  ·  low 11
 OSV       live · 214 queries
 LLM       openai/gpt-5.6
```

One Application. Three service components. Now multiply by every application you own.

Prioritize: what matters

Is it applicable?

- Dev-only or Prod
- Wrong platform - Windows only bug on Linux

Is it reachable?

- Direct & Transitive dep → Vulnerable function is in use or not

Is exploit known or likely?

How bad if it lands?

- Severity, what the service holds

→ 44 Not Applicable · 31 Actionable · 3 Manual Review

Auto-fix: what the harness let through

31 actionable findings



harness-verified
every check passed

flagged for review
generated, not verified

for rewrite
no upstream fix

Rewrite: when a patch isn't enough

The same weakness, new line number.

Fix each finding

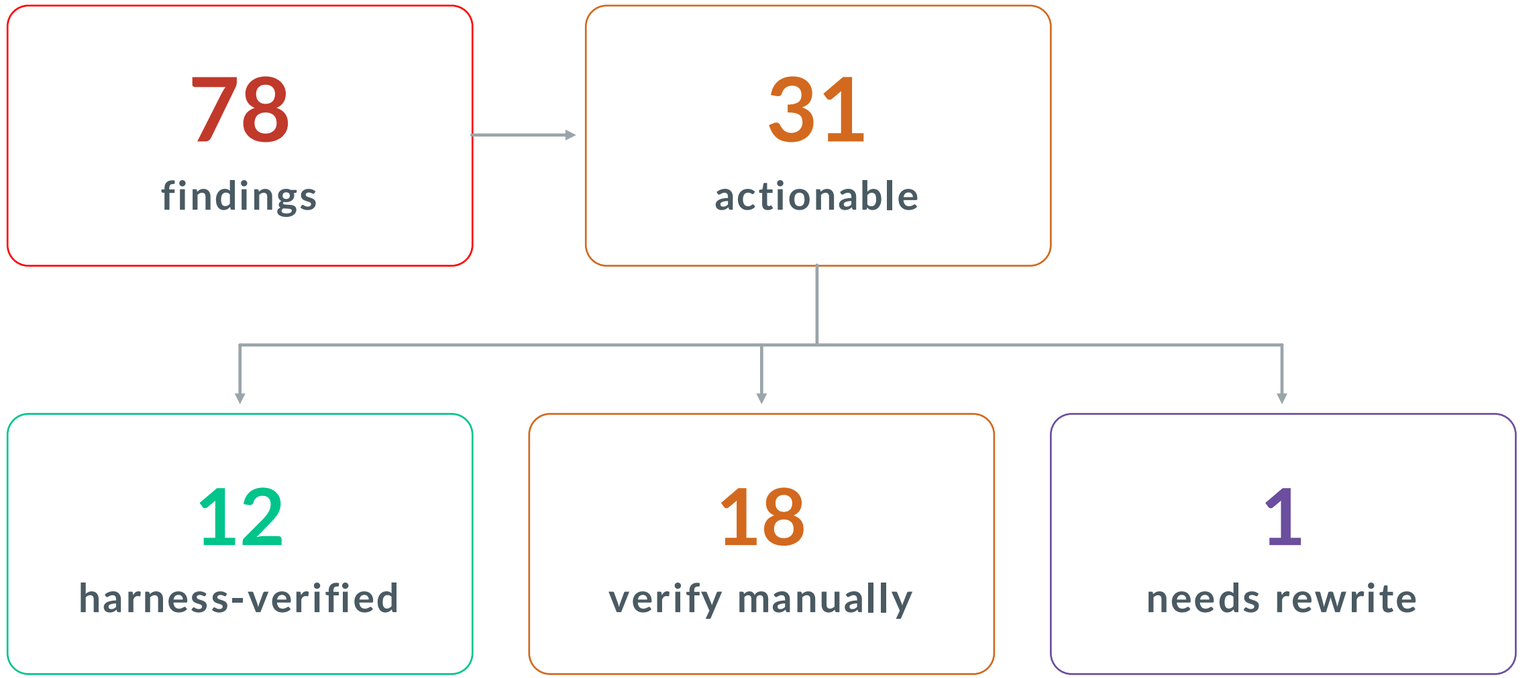
Every time it comes back.

→

Replace the component

Remove the pattern that keeps producing it.

Payoff



Still one application. Three services. Compound it across every application you own.

Build the checks before the fixes

- Normalize every finding into one format
- Drop findings that never reach production
- Write down your own policy
- Build the rejection log
- Build the harness
- Generate patches last

Questions

Connect with Us



