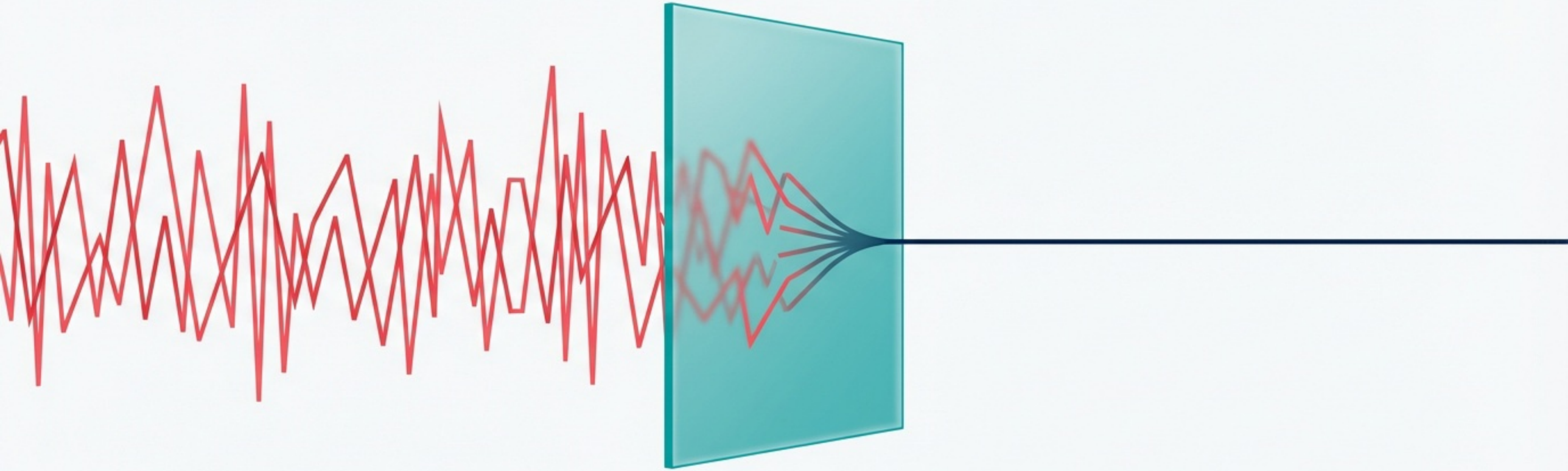
Why Developers Ignore SAST — And How to Fix It

Moving from ‘Finding More’ to ‘Finding Better’

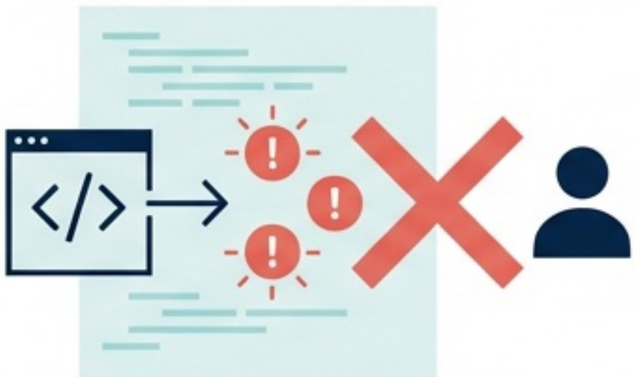


A diagnostic guide to the Trust Crisis in application security.

Security Tools Are Facing a Crisis of Credibility

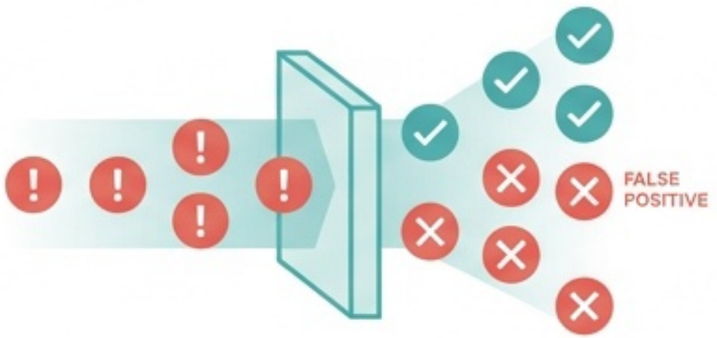
73%

Developers who routinely ignore SAST findings.



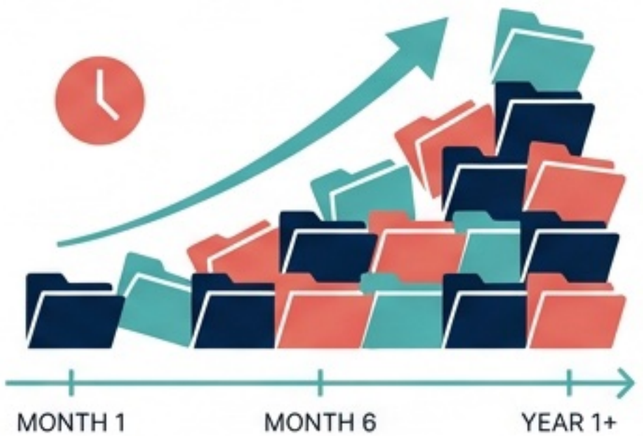
40%

Average False Positive rate in standard deployments.



10k+

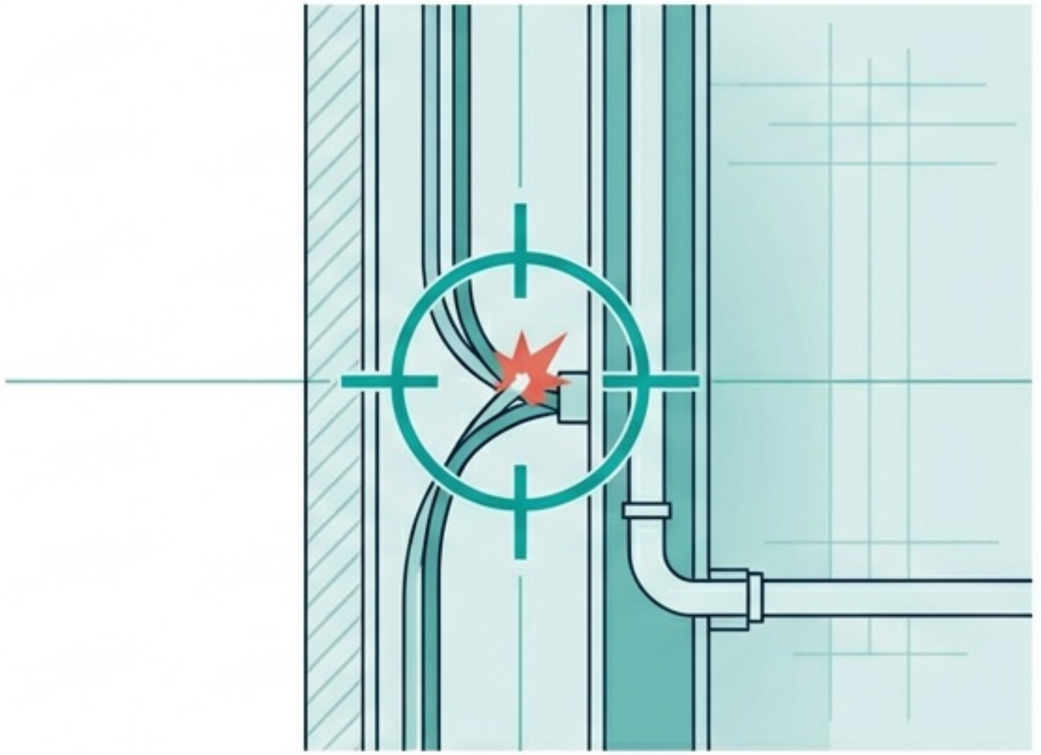
Open findings in the average organization's backlog.



Security teams currently spend 60% of triage time just verifying if findings are real.
When tools cry "wolf" 40% of the time, they get muted.

The Goal Isn’t to Detect More. It’s to Detect Better.

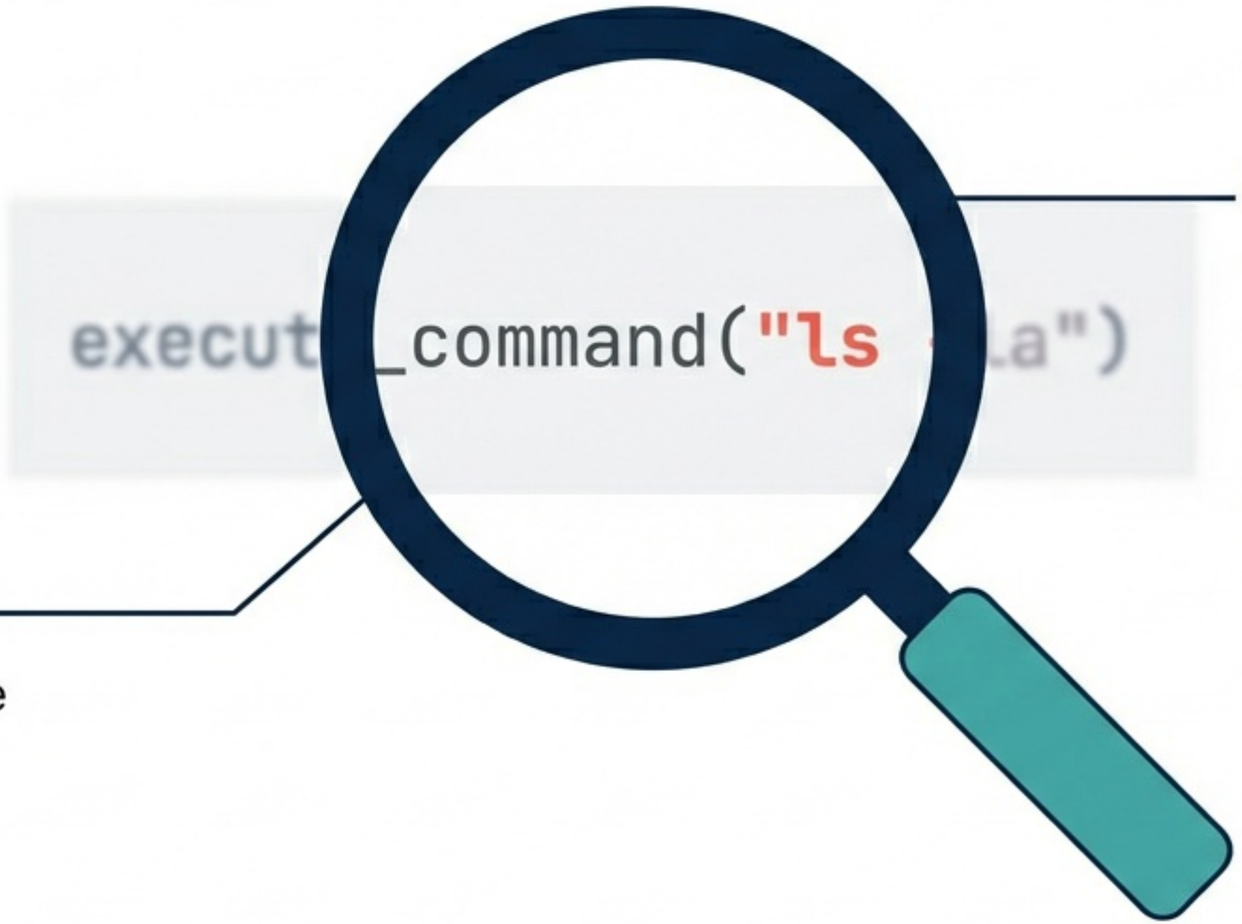
The Status Quo: Annoying. Disconnected.



The Goal: Actionable. Trusted.

A tool that flags safe code effectively trains developers to ignore dangerous code.

Why Traditional SAST Fails: The ‘Ctrl+F’ Problem



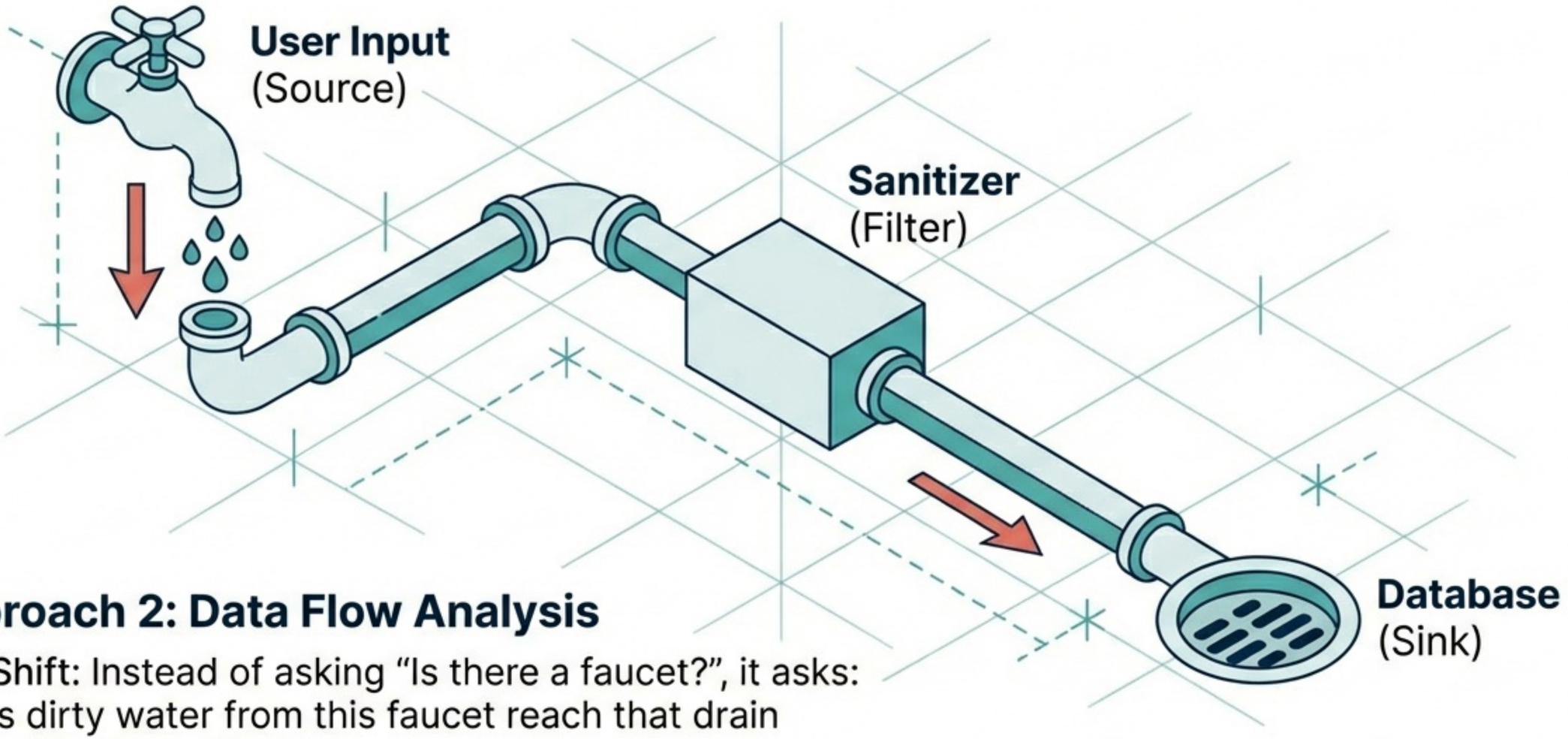
Approach 1: Pattern Matching

Does this code *look* like a vulnerability?

The Flaw

It lacks context. It sees the pattern but misses the safety logic. It doesn't know where the input came from or if it was sanitized.

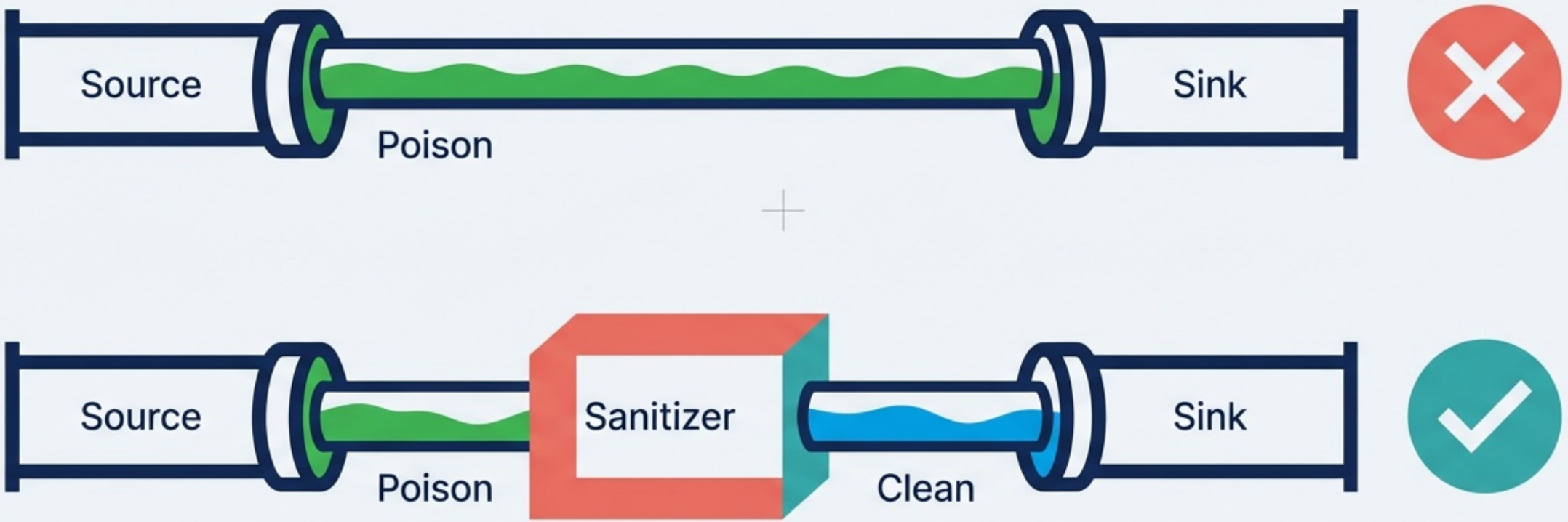
The Solution: Treating Code Like Plumbing



Approach 2: Data Flow Analysis

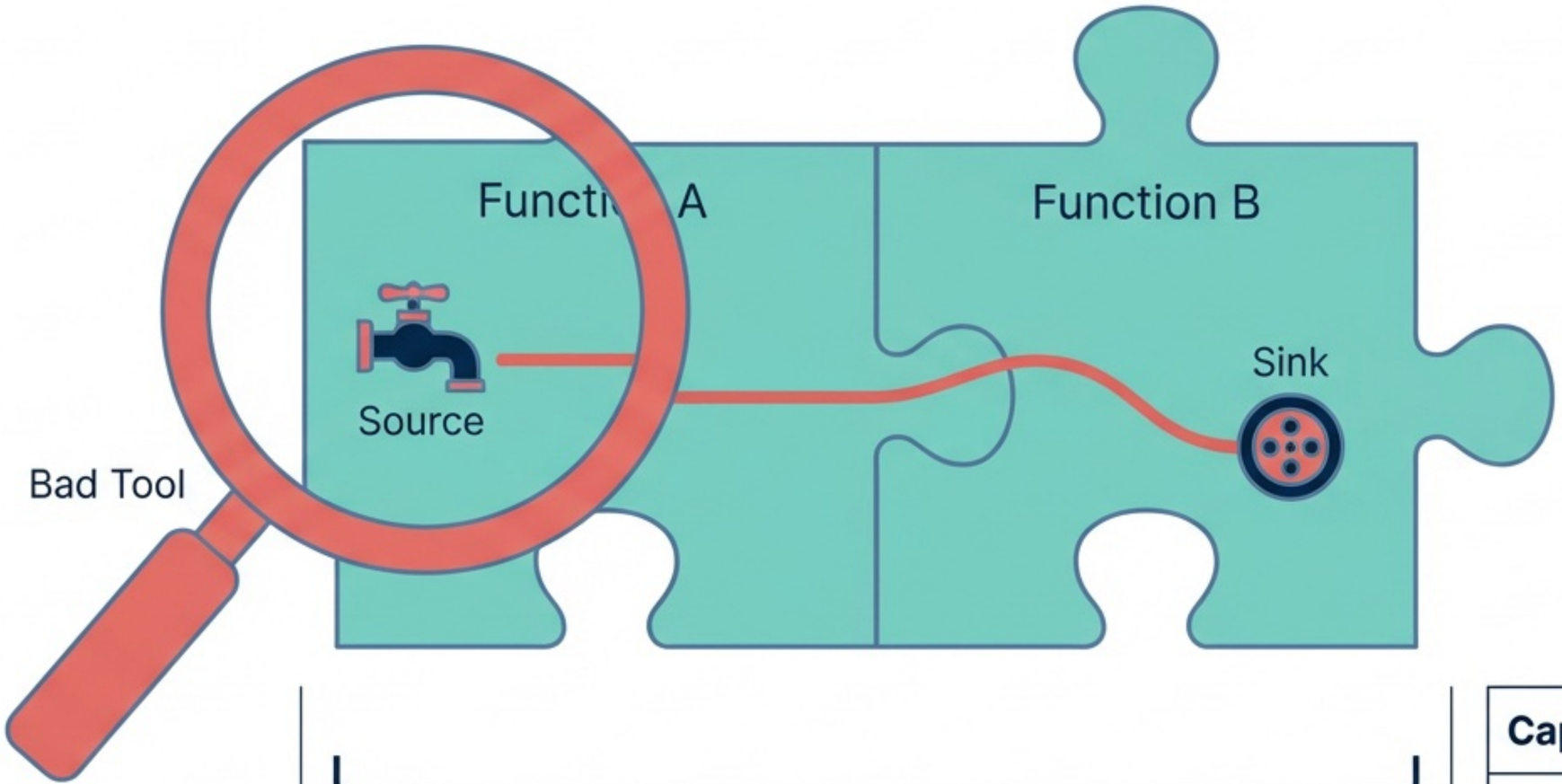
The Shift: Instead of asking “Is there a faucet?”, it asks: “Does dirty water from this faucet reach that drain without hitting a filter?”

Concept 1: Taint Analysis (Following the Poison)



If the tool doesn’t understand the antidote, it flags the cure as the disease.

Concept 2: Vulnerabilities Don’t Respect Function Boundaries



Capability Check:	
Basic Matchers:	✗
CodeQL/Advanced Taint:	✓

Concept 3: Context is King

Not all sanitizers are created equal. A smart tool knows that an HTML shield doesn’t stop a SQL sword.

Safe

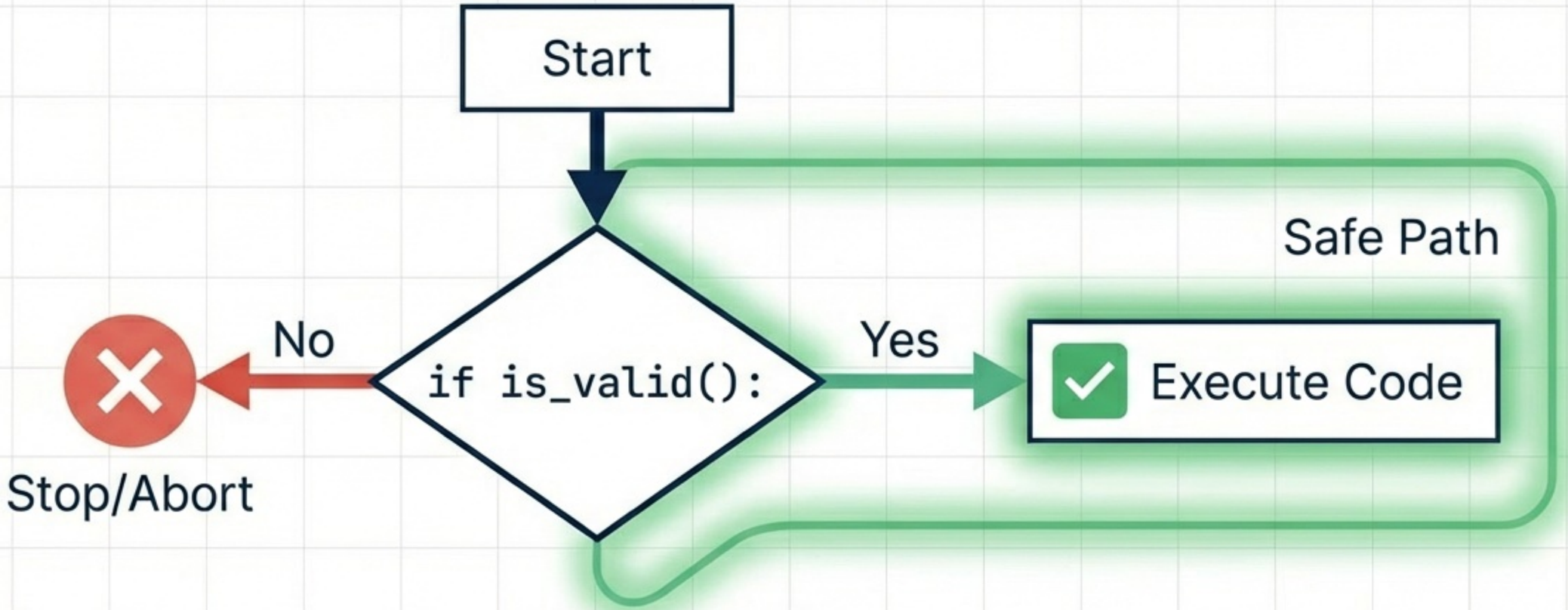
Vulnerable

Concept 4: Precision Matters (Scalpel vs. Hammer)

Without field sensitivity, one dirty drop spoils the whole bucket.

Concept 5: Respecting the Guard Rails

Path Sensitivity means the tool understands that dangerous code inside a validation block is safe. It trusts the developer's logic.



Fix the Workflow, Not Just the Code

The Wall of Shame (Ignored)

The Actionable Diff (Fixed)

Differential (PR) scanning is the #1 driver of adoption. It shifts focus from ‘technical debt’ to ‘stopping the bleeding’.

The Scorecard: 8 Questions to Ask Your Vendor

1. Taint Analysis: Do you trace data or match patterns?

- ✓ We trace data flow across the application.
- ✗ **Red Flag:** We match known vulnerability patterns

2. Cross-Function: Do you track across boundaries?

- ✓ We follow data through function calls and modules.
- ✗ **Red Flag:** We analyze files independently

3. Context: Do you understand sanitizers?

- ✓ We analyze validation logic and sanitization functions.
- ✗ **Red Flag:** We have a list of safe function names

4. PR Scanning: Can I scan just the diff?

- ✓ We provide differential scanning for pull requests.
- ✗ **Red Flag:** We only do full scans

5. Custom Rules: Can I write them in YAML?

- ✓ Custom rules are easily defined in YAML.
- ✗ **Red Flag:** Contact professional services

6. Pathing: Do you show source-to-sink traces?

- ✓ We visualize the complete execution path.
- ✗ **Red Flag:** Finding at Line 42

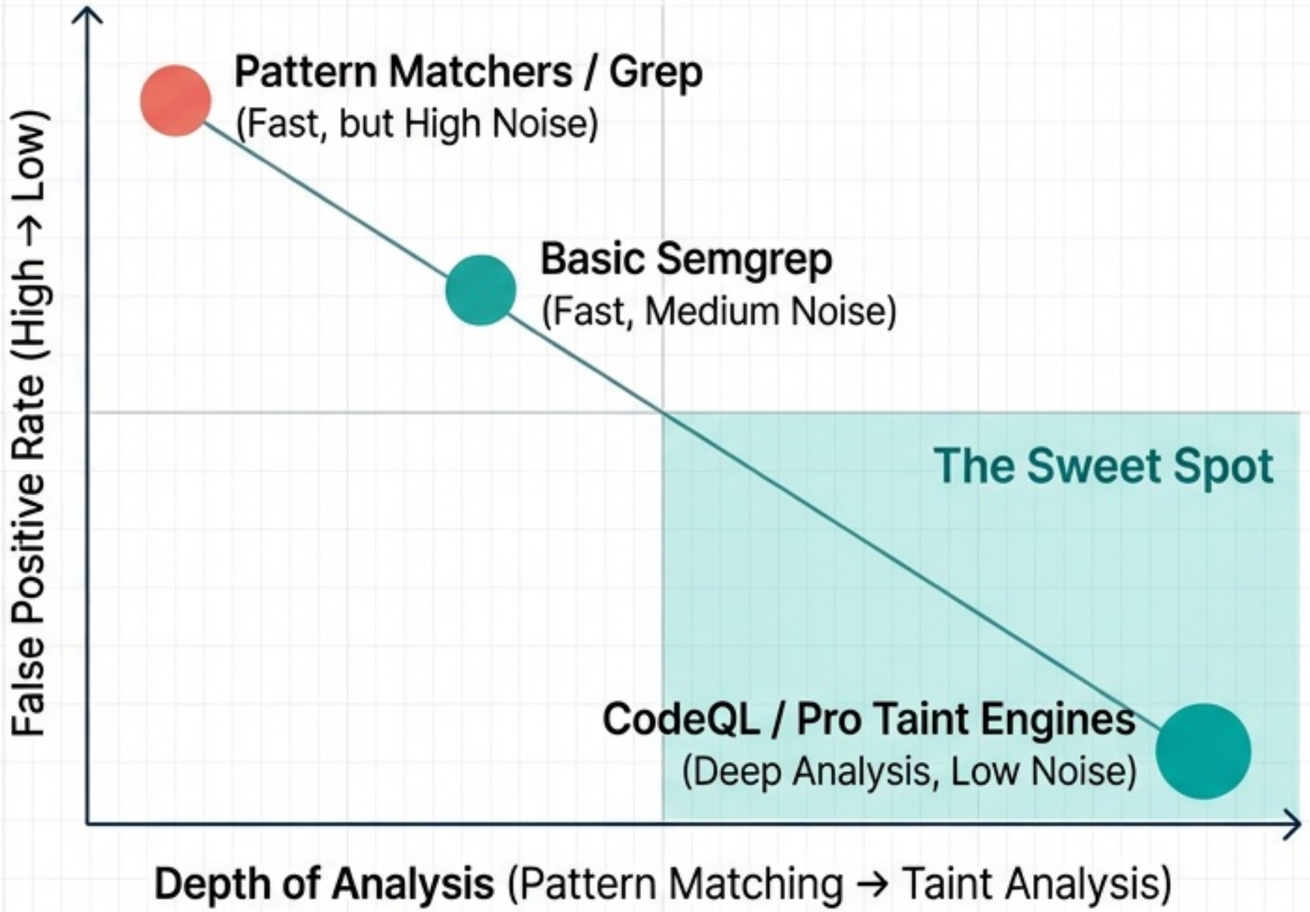
7. Deduplication: Do you group by root cause?

- ✓ We group similar findings by their root cause.
- ✗ **Red Flag:** We report every instance

8. Field Sensitivity: Do you track individual fields?

- ✓ We track data flow at the individual field level.
- ✗ **Red Flag:** Blank stare

The Landscape: Speed vs. Depth



Fast is only good if it’s true. Deep analysis pays for itself in reduced triage time.

The Monday Morning Checklist

1. **Enable PR Scanning:** Stop the bleeding on new code.
2. **Triage Top 50:** Ignore the 10,000 backlog. Fix the critical exploitable paths first.
3. **Write 3 Custom Rules:** Cover your specific internal frameworks.
4. **Measure False Positives:** If developers mark >20% as “Not Applicable,” replace the tool.
5. **Visualize the Path:** Ensure developers see the source-to-sink trace.

“Stop measuring your SAST tool by how many vulnerabilities it finds.

Start measuring it by how many vulnerabilities developers actually fix.”

Resources: OWASP SAST Comparison • CodeQL Docs • Semgrep Docs