Turning the dial on SAST: Reducing False Positives with Call Graph–Driven LLM Reasoning

Iman Sharafaldin and Vrushal Nedungadi | Forward Security

2026-06-01

Mission Briefing

- A signal every security team keeps receiving
  (and keeps learning to ignore)
- A tool built to help us
  (but buried beneath its own noise)
- Thousands of findings across a project
  (and only a handful that truly matter)
- Mission: Find a way to finally make sense of the mess
  (without building a 20-person triage team)



FWDSEC»

The Best of Us

- Before the mission,
  there were people who believed
  static analysis had already solved the problem.
- The scanners were trusted.
- Every finding mattered.
- And among them,
  there was one man they trusted more than anyone else.



FWDSEC»

FACT: Whatever SAST Reports Is CRITICAL

(— Dwight Schrute, Director of Security)

- DWIGHT never wants to miss a vulnerability.
- His philosophy was simple: if SAST reported it, it had to be real.
- At a security summit, he boldly declared:
  - "FACT: Whatever SAST Reports Is Critical."
- When I suggested there could be false positives, Dwight stared at me and asked:
  - "WHO ARE YOU?"



Dwight K Schrute,
Director of Security, Dunder Mifflin

FWDSEC»

WHO WE ARE?
Dr. Brand and Cooper to the rescue



Iman Sharafaldin
Application and Cloud Security Lead



Vrushal Nedungadi
Application Security Analyst

FWDSEC»

The Blight
The origins of SAST and why it is worth fighting for



- The codebases grew.
- The attack surface grew.
- The findings grew.
- Human review could no longer keep pace.
- So we built SAST.
- And for a while... **it worked**.

FWDSEC»

The Blight spreads – Who clears the mess?

- Hundreds of alerts piling up
- Developers may spend more time triaging than coding
- Real vulnerabilities get lost in the noise
- Alerts feel like “shouting in a crowded room”
- Manual cleanup is slow, error-prone, and exhausting



FWDSEC»

The Endurance

What we bring to the table

- A Call Graph–driven understanding of your codebase
  So findings aren't just matched to files they're matched to actual execution paths.

- Smarter SARIF interpretation
  We extract vulnerable snippets, enrich them with context,
  and map them to the right nodes in the graph.

- A two-stage LLM reasoning engine
  Using CWE-specific guidance,
  it evaluates whether the finding really behaves like a vulnerability.

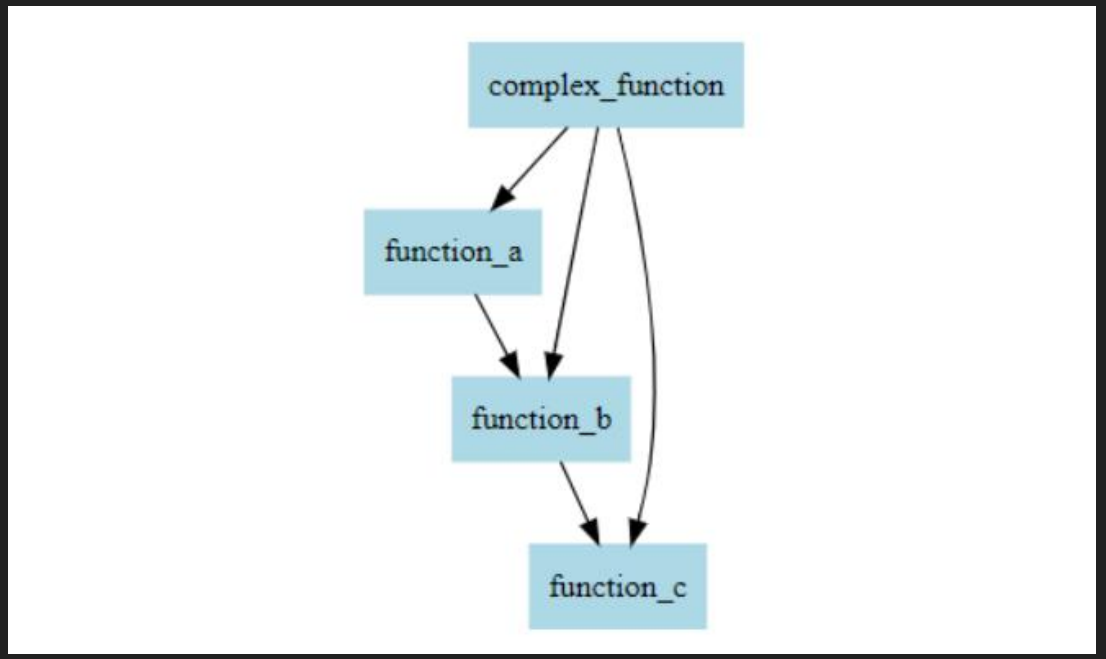
- End-to-end automation that cleans the noise
  From parsing → mapping → reasoning → reporting.



FWDSEC»

The Wormhole called Call Graph

What the damn hell is a Call Graph



- A call graph is a map of function calls in your code.
- Each node represents a function/method, each edge represents a call from one function to another.
- Purpose: understand how code flows, which is essential for analyzing vulnerabilities.

FWDSEC»

So… how do we build one?



Parse the codebase with Tree-sitter

Identify method/function definitions, calls, and import statements.

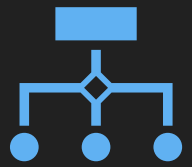
Imports help infer types and fully resolve methods across modules.



Map caller-callee relationships

For each node (function/method), determine which functions it calls and which functions call it.

Maintain lists of callers and callees for each node.



Build the graph

Nodes = functions/methods

Edges = calls between functions (caller → callee)

Result: a complete call graph representing the flow of the code.

FWDSEC»

How is it useful?

Enables contextual analysis → Identify caller and callee chains → Reduces false positives

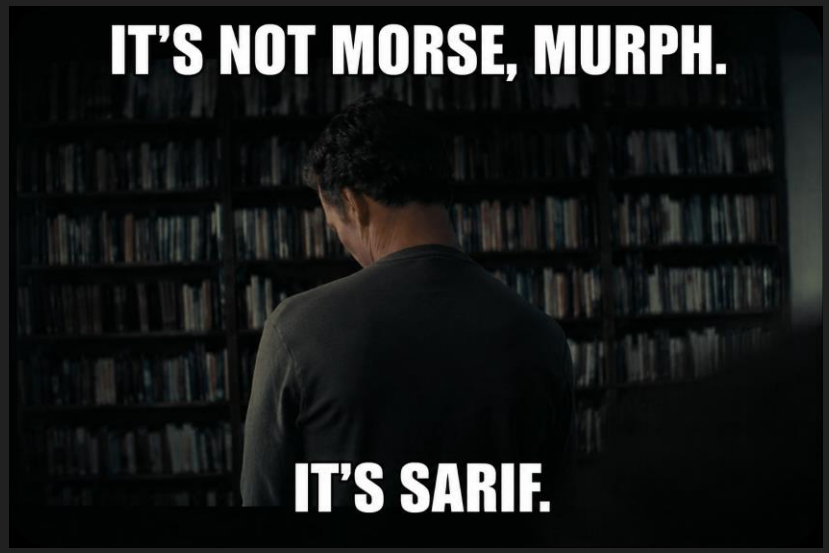
FWDSEC»

The co-ordinates from the previous missions

What a SARIF is and WHY SARIF?

The transmission protocol of the mission -

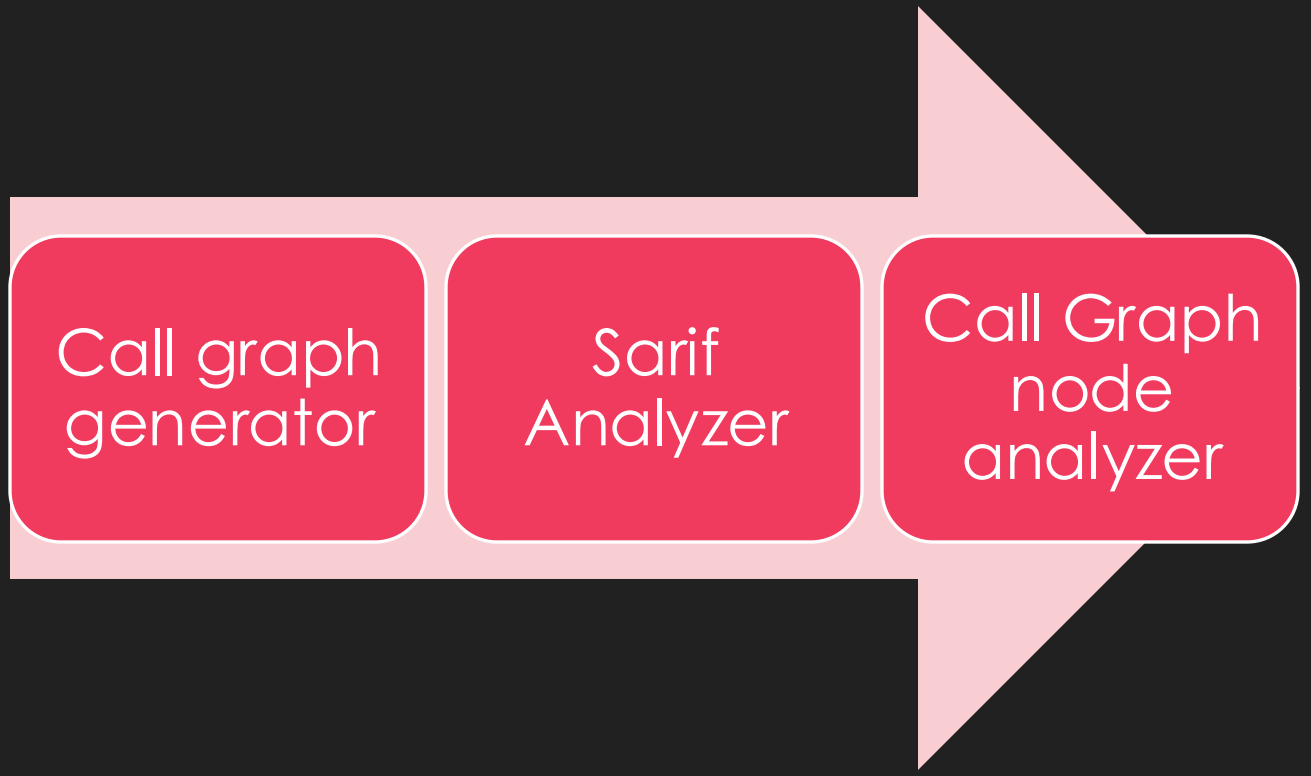
- SARIF is the standard format used by SAST tools to report findings.
- It captures the signal: locations, snippets, rules, severity, and metadata.
- It allows scanners, platforms, and tools to speak a common language.
- For Aletheia, SARIF provides the coordinates: the CWE and code snippet that anchor us in the call graph.
- The signal tells us where to look; context tells us what it means.



FWDSEC»

Loading the Endurance – step by step

- We first build a call graph by parsing the codebase into methods, calls, and relationships.
- The SARIF analyzer then reads each finding and extracts the vulnerable code snippet and location.
- We map that SARIF location onto the call graph, finding the exact method node where the snippet lives.
- From that node, we generate its caller and callee chains, giving full execution context around the finding.



FWDSEC»

But how do we identify the vulnerable node from just the snippet?

- Pattern matching
- Signature matching
- Last-resort fallback
- Heuristics & scoring
- Content matching



FWDSEC»

Meet TARS, we call ours ALETHEIA

- Ground the finding in reality: parse SARIF, map nodes, and recover execution context.
- Enrich the signal with caller/callee chains, snippets, metadata, and CWE knowledge.
- LLM-1 investigates the finding using CWE-specific reasoning.
- LLM-2 audits the investigation and challenges unsupported conclusions.
- Consensus produces the final verdict, updated SARIF, and analyst-ready reports.



FWDSEC»

How does Aletheia know a finding is false positive?



- We do not let it guess; each finding is grounded in SARIF, call graph context, and CWE-specific guidance.
- The CWE prompt defines what matters: sources, sinks, sanitizers, and required control-flow conditions.
- The enriched context shows whether those conditions actually occur in the code path.
- LLM-1 reasons over the evidence; LLM-2 challenges the reasoning.
- A finding is marked false positive only when the expected CWE conditions do not hold.

FWDSEC»

You mean CVE? What is CWE? Why CWE?

- CVE = specific incident,

  a real vulnerability in a particular product/version.

- CWE = weakness pattern, the general flaw type SAST tools detect, not product-specific issues.

- LLM uses CWE rules (sources, sinks, sanitizers, flow conditions) to judge whether the reported issue actually fits the pattern or is a false positive.



FWDSEC»

What after the detection?



- Each finding gets a final verdict: true positive or false positive
- An updated SARIF
- A structured JSON report: includes full traces, analysis notes
- An analyst-friendly Excel report

FWDSEC»

The Water Planet

Time moves differently when context is missing

Time moves differently when context is missing -

- Analysts reconstruct context manually, one path at a time.
- Vulnerabilities rarely exist in isolation.
- Call graphs collapse the distance between a finding and its context.
- The wormhole was never the visualization. It was the shortcut through the codebase itself.



FWDSEC»

The full data from the mission

Category	Total Cases	Correctly Identified	Incorrectly Identified	Accuracy
True Vulnerabilities (TP set)	943	**908**	35	**96.3%**
False Positives (FP set)	580	**423**	157	**72.9%**

FWDSEC»

Not Every Planet Was Habitable

CWE-level results

CWE	Weakness	Accuracy (Sonnet 4.5 6 months ago)	Accuracy (Opus 4.8 now)
CWE-501	Trust Boundary	**100%**	84.4%
CWE-79	XSS	**~90%**	~80%
CWE-22	Path Traversal	~93%	~93%
CWE-89	SQL Injection	~88%	**~94%**
CWE-90	LDAP Injection	~75%	**~90%**
CWE-78	Command Injection	~50%	**~75%**
CWE-643	XPath Injection	~25%	**~75%**

FWDSEC»

Brand New Worlds

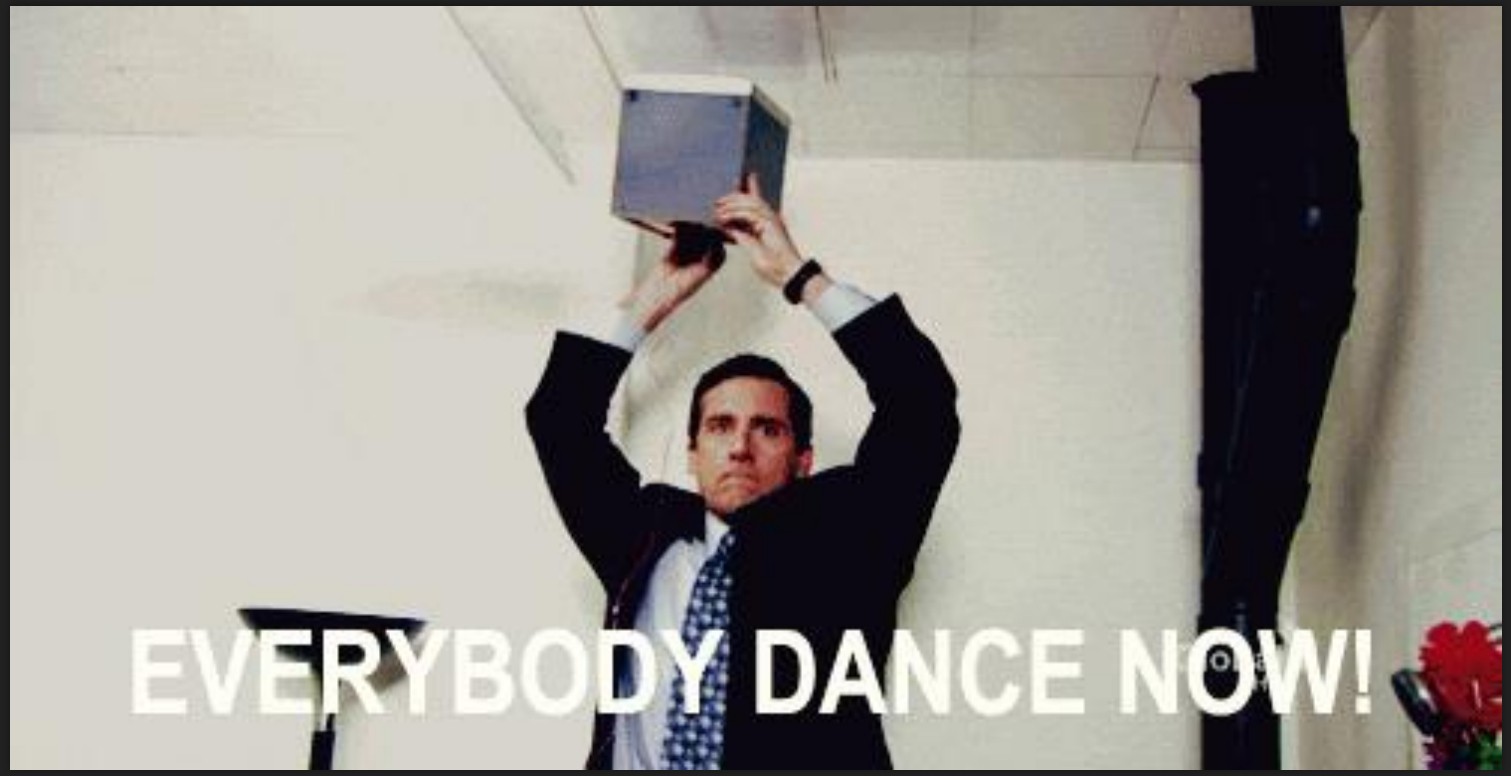
- Aletheia was never meant to end the mission.
- Only to make the journey survivable.
- More CWEs remain.
  More languages.
  More complex execution paths.
  More uncertainty.
- The search continues.



FWDSEC»

You've reached the maximum Token Limit

To chat more on Aletheia, please subscribe! JK!



FWDSEC»

Connect with us

Connect with us on LinkedIn →

Or on mail →

i.sharafaldin@fwdsec.com

v.nedungadi@fwdsec.com

Visit https://forwardsecurity.com/ to know more!

Iman's LinkedIn

Vrushal's LinkedIn

FWDSEC»

S.T.A.Y

- I received a morse code message a few days ago
- It read: -.. .-- .. --. .... - .-- .- ... .-- .-. --- -. --.
- I asked Claude to decipher it for me, and it did.
- It says DWIGHT WAS WRONG.
- Probably it was me from the future ☺

FWDSEC»