

SYNERGY DEVPARTNER CONFERENCE 2026

Static Code Analysis: From Output You Can Actually Read to Onboarding You Can Actually Do

Oliver Chu, Software Engineer

The Problem We’re Trying to Solve

1. Even if it compiles, security issues can still exist in code
2. Some vulnerabilities are difficult to detect through testing alone
3. Manual code review becomes harder as applications grow
4. Automated detections are useful for AI-written code



How CAIT can help you

01

Find security defects, especially untrusted input reaching command execution or file-system operations

02

Detect unsafe or problematic use of DBL types, field widths, argument directions, and memory overlays

03

Resolve ordinary and dynamic DBL calls (XSUBR, XADDR) well enough for such analysis

04

Report findings, source locations, and dataflow paths in SARIF

Why Teams Use CAIT

SYNERGY DEVPARTNER CONFERENCE 2026

Catch hard-to-see issues before they reach production

01

FIND RISKS EARLIER

Detect potential vulnerabilities in source code without running the application.

Before release

02

SEE CROSS-PROJECT FLOWS

Follow values through DBL routines, calls, and referenced projects.

Beyond one file

03

ACT ON CLEAR EVIDENCE

Export locations, rule information and dataflow steps in standard SARIF.

Developer-friendly

CAIT supplements existing development, testing, and review processes.

How CAIT Tracks Risky Data

SYNERGY 2026 DEVPARTNER CONFERENCE

From an input operation to a sensitive action

1 INPUT

Untrusted data enters

xfParameter

→

2 DATAFLOW

Value moves through code

Assignments + calls

→

3 SINK

Potentially unsafe use

SPAWN / SHELL / OPEN

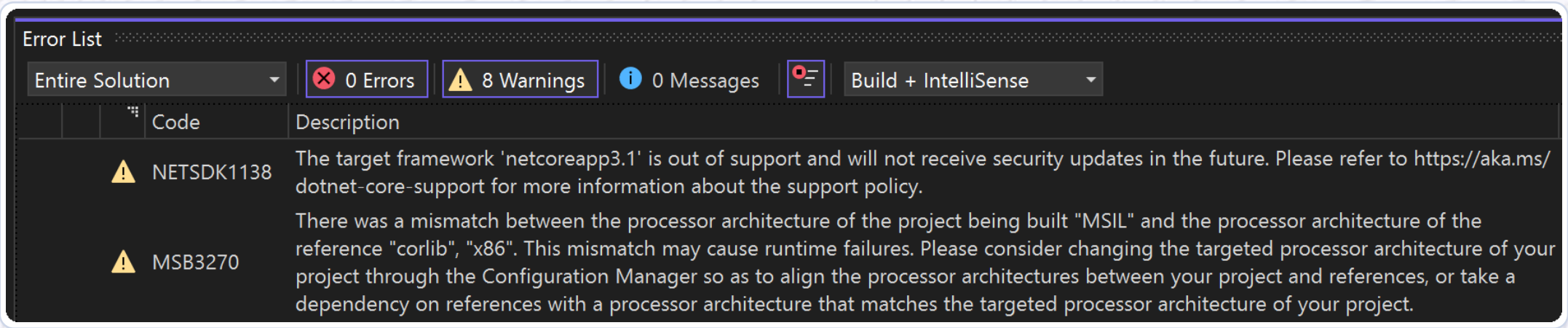
The finding helps a developer see why an input might reach a sensitive operation.

Illustrative flow. CAIT reports source-to-sink steps when the analysis can reconstruct them.

Detect Type Confusion



Different from Compiler Warnings



- CAIT detects issues not covered by compiler errors and warnings
- We derived a robust type system for you
- Type confusion and type safety scenarios
- Alpha to decimal assumptions

What CAIT Checks Today

SYNERGY DEVPARTNER CONFERENCE 2026

Security and type-safety issues relevant to Synergy DBL

CWE-78	**Command injection**	Untrusted data may reach command execution *SPAWN · SPAWN_REDIRECT · SHELL*
CWE-22	**Path traversal**	Untrusted data may influence a file path *OPEN*
DBL	**Type safety**	Potential truncation, incompatible types, or overlapping storage *Field widths · Overlays · Argument directions*

CAIT identifies potential problems for review; the current rule set is not a complete security audit.

Installing CAIT



1. .NET 10 is required on Windows or Linux

2. Machines with more RAM are preferable, 64 GB is a good baseline

Prepare the CAIT Environment

SYNERGY DEVPARTNER CONFERENCE 2026

What you need before your first scan

CAIT TOOLS

- To install CAIT, run the MSI installer for CAIT on our Downloads page.
- Install the required .NET 10 runtime/SDK for caitsln.

YOUR BUILD ENVIRONMENT

- Source files, .sln/.synproj, and all referenced inputs.
- MSBuild/Synergy targets and the relevant compiler setup.
- Accessible includes and synergy.ini settings. DBPs should be made by the build.

Larger solutions may need substantial RAM; 64 GB is a practical planning baseline, not a strict minimum.

Onboarding



1. Solution scans use MSBuild
2. Visual Studio can manage traditional Synergy DBL projects for solution analysis
3. Analyze DBL files through their solution with CAIT solution
4. A successful compilation yields the most complete analysis results

Running CAIT on Your Solution

Scan an entire solution

COMPLETE SOLUTION

Follow calls across projects

```
cait solution MyApp.sln
```

Best when routines and dependencies span projects

For cross-project dataflow, use full-solution analysis with the right build configuration.

Set Up Your Solution Scan

SYNERGY DEVPARTNER CONFERENCE 2026

Match the same project settings your build already uses

1

Project evaluation

Run CAIT against the .sln file. Select Configuration / Platform if needed.

--configuration Release

2

Solution discovery

CAIT uses caitsln and MSBuild. Keep caitsln and CAIT in the same directory.

caitsln + CAIT together

3

Synergy environment

Use dblvars and the Synergy bin folder. Ensure synergy.ini and needed paths resolve.

dblvars · Synergy bin

Use the same Synergy build environment that successfully builds your solution.

Academic Basis

SYNERGY DEVPARTNER CONFERENCE 2026

ABSTRACT INTERPRETATION : A UNIFIED LATTICE MODEL FOR STATIC ANALYSIS OF PROGRAMS BY CONSTRUCTION OR APPROXIMATION OF FIXPOINTS

Patrick Cousot* and Radhia Cousot**

Laboratoire d'Informatique, U.S.M.G., BP. 53
38041 Grenoble cedex, France

What We Had Before

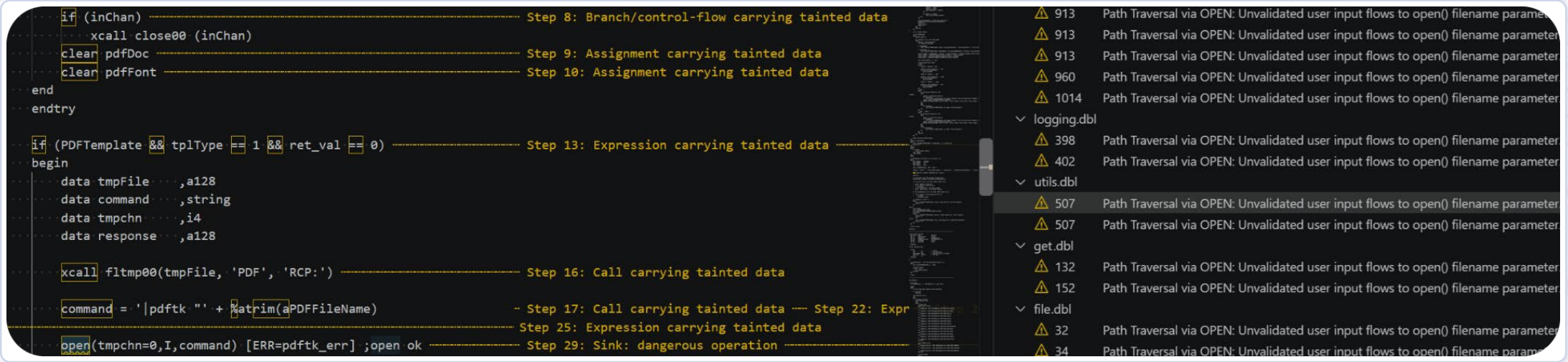
```
                |                                        ^^^^^^^^
            16 |     freturn 0
            17 | endfunction

        [Step 2 of 8]
        System.Console.ReadLine()@System.String callsite was passed into
            STRING_IDENTITY.parm which is parameter #1
            Current routine: DSPIMG_FUNCTION
            In file: C:\customerprojects\unx2\WelfareAppV10\aoss\GTIMG4.DBL

            13 | endcommon
            14 | proc
            15 |     fld = %STRING_IDENTITY(Console.ReadLine())
                |             ^^^^^^^^^^^^^^^          ^^^^^^^^
            16 |     freturn 0
            17 | endfunction

        [Step 3 of 8]
        STRING_IDENTITY.parm was assigned to STRING_IDENTITY.temp
            Current routine: STRING_IDENTITY
            In file: C:\customerprojects\unx2\WelfareAppV10\aoss\GTIMG4.DBL
```

What Is SARIF?



Open VS Code, install the Microsoft SARIF Viewer Extension, and open a SARIF file produced by CAIT.

Find Your CAIT Reports

SYNERGY 2026 DEVPARTNER CONFERENCE

CAIT writes machine-readable SARIF 2.1.0 results

Where the report goes

Solution: per project

```
<OutputDb>\<ProjectName>.sarif
```

Solution: combined

```
--combined-output reports\all.sarif
```

What a finding contains

- **Rule & severity**
  Why it was flagged
- **Source location**
  Where to inspect code
- **Code flow**
  How data reached the sink
- **CWE & metadata**
  Category and uncertainty

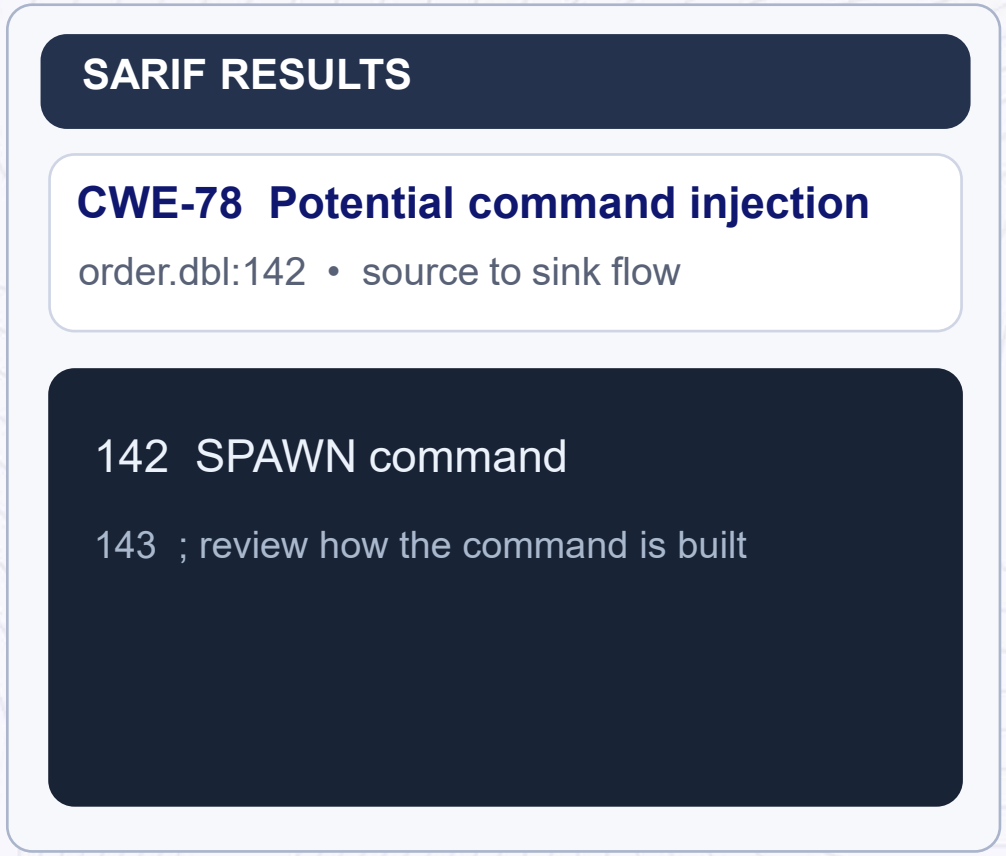
A project with no findings may not have a per-project SARIF file.

Review Results in VS Code

SYNERGY 2026 DEVPARTNER CONFERENCE

A practical review workflow

1. **Open the relevant source folder**
   Make the project files available to VS Code.
2. **Install SARIF Viewer**
   Use the Microsoft SARIF Viewer extension.
3. **Open the .sarif file**
   Browse results and select a finding.
4. **Navigate to the source**
   Inspect the highlighted sink and code-flow steps.



If navigation fails, check that source files exist at the paths/URIs contained in the SARIF report.

Turn a Finding Into a Fix

SYNERGY DEVPARTNER CONFERENCE 2026

Review the evidence, then decide what needs to change

1

OPEN THE FINDING

Check rule, severity and source location.

2

FOLLOW THE PATH

Trace the input through operations and calls.

3

VERIFY THE RISK

Look for validation or sanitization in context.

4

FIX & RESCAN

Remediate the code and rerun the analysis.

Validate the finding before fixing.

The code owner validates exploitability and confirms the fix in the context of the application.

Automate CAIT in CI/CD

SYNERGY DEVPARTNER CONFERENCE 2026

Scan code during your normal build pipeline

Source	Build agent	CAIT	SARIF	Review
Commit or PR	Synergy setup	Analyze .sln	Publish artifact	Triage findings

```
cait solution MyApp.sln --combined-output artifacts\cait.sarif
```

- Ensure the pipeline agent has all the same build inputs, targets, and environment settings.
- Publish SARIF for review; define any severity-based build gate separately.

Useful CAIT Command Options

SYNERGY 2026 DEVPARTNER CONFERENCE

Common switches for a solution scan

`--configuration`

Select a solution configuration, such as Release.

`--platform`

Select a solution platform.

`--project`

Limit the scan to selected projects and dependencies.

`--combined-output`

Write an additional whole-solution SARIF report.

`--include-xfmethod-projects`

Include all projects using xfMethod without listing them individually.

`--json-summary`

Convert the final summary into JSON.

Match --configuration and --platform to the settings your build already uses.

Troubleshooting Your First Scan

If a run fails or a report looks incomplete

CAIT cannot load the solution	If you type msbuild at the command line for your solution, does it build? Also check .NET 10, caitsln, and Synergy targets.
Compiler cannot resolve files	Check synergy.ini, includes, DBPs, logicals, and variables.
No expected SARIF file	Check OutputDb and whether a project actually had findings.
SARIF cannot open source	Make sure files exist and URIs/paths resolve in the workspace.

Diagnostic aids: --verbose and --log-file

Your First CAIT Scan

SYNERGY DEVPARTNER CONFERENCE 2026

A practical three-step starting point

1 PREPARE

Confirm CAIT/caitsln, .NET 10, MSBuild, and your Synergy environment.

2 SCAN

Run CAIT against a representative solution.

3 REVIEW

Open the SARIF in VS Code, trace a finding, and plan remediation.

```
cait solution MyApp.sln --combined-output cait-results.sarif
```

Next: Run CAIT on a real project, review findings, and repeat the scan.

Use the same solution scan workflow manually or in your CI/CD pipeline.

Live Demo

Image Credits

SYNERGY 2026 DEVPARTNER CONFERENCE

1. **MICROSOFT .NET LOGO**

   https://en.wikipedia.org/wiki/File:Microsoft_.NET_logo.svg

2. **VISUAL STUDIO 2026 ICON**

   https://en.wikipedia.org/wiki/File:Visual_Studio_Icon_2026.svg

3. **COUSOT & COUSOT (1977) PAPER**

   https://cs.nyu.edu/~pcousot/publications.www/CousotCousot-POPL-77-ACM-p238--252-1977.pdf