$ cppcheck happy_coder.cpp
Checking happy_coder.cpp ...
[happy_coder.cpp:5]: (error) Array 'a[4]' accessed at index 6, which is out of bounds.
Individual experts build standard in the open 1/2

OASIS Static Analysis Results Interchange Format (SARIF) TC

Defining a standard output format for static analysis tools

David Keaton, dmk@dmk.com, Chair
Luke Cartey, Chair
Stefan Hagen, stefan@hagen.link, Secretary

Table of Contents

- Announcements
- Overview
- Subcommittees
- TC Liaisons
- TC Tools and Approved Publications
- Technical Work Produced by the Committee
Announcements


The first meeting of the OASIS SARIF Technical Committee was held via teleconference on September 06, 2017. David Keaton (Individual) and Luke Cartey (Semmle) were elected TC Co-Chairs.

Participation in the OASIS SARIF TC is open to all interested parties. Contact join@oasis-open.org for more information.
Real people and really open!

OASIS SARIF TC: Repository for development of the draft standard, where requests for modification should be made via Github Issues. https://github.com/oasis-tcs/sarif-spec

- 50 commits
- 4 branches
- 0 releases
- 3 contributors

Latest commit 2f95b28 3 days ago

- Documents: Propose language for the result.rank property.
- meetings: Meeting minutes draft of #6 from 2017-NOV-08.
- CONTRIBUTING.md: create static boilerplate CONTRIBUTING file
- LICENSE.md: create static boilerplate LICENSE file
- README.md: update with link to Issues
- Workflow.md: Fix #67: Clarify approval process in Workflow.md
Purpose (from contributor slides)

Make developers more productive by enabling them to interact with results from multiple analysis tools in a uniform way.

- Enable uniform viewing experiences (e.g., IDE integrations)
- Enable uniform storage in and retrieval from a back end
  - “result management systems”
Other applications (from contributor slides)

• Dynamic analysis tools
  • code flow support with location.kind

• Web scanning tools
  • analysis targets expressed as URLs
Design goals (from contributor slides)

- Comprehensively capture range of data produced by commonly used static analysis tools.

- Be a useful format for analysis tools to emit directly,
  - and also an effective interchange format into which the output of any analysis tool can be converted.

- Be suitable for use in a variety of scenarios related to analysis result management,
  - and be extensible for use in new scenarios.

- Reduce cost & complexity of aggregating the results of various analysis tools into common workflows.

- Capture information useful assessing project compliance with
  - corporate policy or conformance to certification standards.

- Adopt a widely used serialization format that can be parsed by readily available tools.

- Represent analysis results for all kinds of programming artifacts
  - including source code and object code.

- Represent logical construct against which a result is produced
  - such as a function, class, or namespace.

- Represent physical location at which a result is produced
  - including problems detected in nested files
    - ... such as a source file within a compressed container
History (from contributor slides)

- 2013: Originated in Microsoft’s security organization
  - to unify results produced by several security-related static analysis tools.

- Developed “in the open”:
  https://github.com/sarif-standard/sarif-spec
  - Open issues in the repo are reported in OASIS
    - for resolution by the SARIF TC –
      - including concepts from related formats such as:
        - SATE and SWAMP/SCARF.

- Supported by latest Microsoft C#/VB/C++ compilers, a variety of publicly available Microsoft tools, as well as tools from .....
Features (from contributor slides)

- Multiple runs per file
- Tool information
- Run/invocation information
- Rich description of “results” including:
  - “Physical” and “logical” locations
  - Multiple locations per result
  - Code flows
  - Stacks
  - Fixes
- Rule metadata
- File metadata
  - Hashes
  - MIME type
  - Embedded contents
- Support for both text and binary files
- Support for “nested” files
- Tool notifications (e.g. capture tool console output)
- Support for “baselining"
Example:

```json
{
  "version": "1.0.0",
  "runs": [
    {
      "tool": {
        "name": "CodeScanner",
        "semanticVersion": "2.1.0"
      },
      "files": {
        "file:///user/builder/work/src/collections/list.cpp": {
          "mimeType": "text/x-c"
        }
      },
      "results": [
        {
          "ruleId": "C2001",
          "message": "Variable \"count\" was used without being initialized.\",
          "locations": [
            {
              "analysisTarget": {
                "uri": "file:///user/builder/work/src/collections/list.cpp",
                "region": {
                  "startLine": 15
                }
              },
              "fullyQualifiedLogicalName": "collections::list:add"
            }
          ],
          "rules": {
            "C2001": {
              "id": "C2001",
              "fullDescription": "A variable was used without being initialized. This can result in runtime errors such as null reference exceptions."
            }
          }
        }
      ]
    }
  ]
}
```
One sample free OSS “Integrator” – SWAMP

SWAMP Funding and Support
The SWAMP is funded by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency, Cyber Security Division (DHS S&T/HSARPA/CSD)
Thanks.