Rapid Adjudication of Static Analysis Results During CI:

Reporting for the SEI Research Review

November 2021
Dr. Lori Flynn (PI)
Copyright 2021 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM21-0835
Problem: It takes too much time to adjudicate static analysis (SA) results during continuous integration (CI).

Solution: Increase the number of SA results addressed automatically, by enabling optional CI integration with 2 types of automated adjudication transfers between code versions and use of classifiers in an automatically updating system.

Benefits: Develop more secure software in less time at lower cost, for a wide variety of SA tools and development systems, including modern CI systems.
A meta-alert is an SA result for a particular line number, filepath, and code flaw condition (e.g., CWE-190).

SCAIFE classifies meta-alerts as
- Expected True Positive (e-TP)
- Expected False Positive (e-FP)
- Indeterminate (I)
A meta-alert is an SA result for a particular line number, filepath, and code flaw condition (e.g., CWE-190).

SCAIFE classifies meta-alerts as
- Expected True Positive (e-TP)
- Expected False Positive (e-FP)
- Indeterminate (I)
SCAIFE Static Analysis Classification

Designed for use by machine learning novices, with settings that can be tweaked by experts

Labeled SA meta-alerts are used to create classifiers

- Manually adjudicated meta-alerts (true positive, false positive)
- Test suites (e.g., Juliet): SCAIFE automatically adjudicates meta-alerts
- User chooses labeled data sets, classifier, active learning, and other options

Modular ability to add different types of classifiers, active learning, and hyper-parameter optimization methods.
SCAIFE’s Classification System

Use SCAIFE to adjudicate and classify results for a single code version at a time, in a CI-optional system with automated updates to code and SA

Modular system designed to work with a wide variety of systems and tools
  • with different user interfaces and SA tools
    - SARIF static analysis format
    - SCARF format (DHS SWAMP)
    - Various tools and versions, with a standard method for adding new tools
  • Docker containers usable in a wide variety of systems
Rapid Handling of Static Analysis Meta-Alerts During CI

Source Code Repository

CI Workflow

CI Server

Source Code Check-In

Development & Test Teams

SA alert

CI Server

Build Install Deploy

Run Tests

Run Static Analysis

Report the Results

Unit, Integration, Stress Tests

SA alert

SA alert

SA alert

SA alert

SA alert
Rapid Handling of Static Analysis Meta-Alerts During CI

Source Code Repository

CI Server

CI Workflow

Development & Test Teams

CI Server

Build Install Deploy

Run Tests

Report the Results

CI Server

Run Static Analysis

Unit, Integration Stress Tests

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert

SA alert SA alert SA alert SA alert SA alert SA alert SA alert SA alert
Rapid Handling of Static Analysis Meta-Alerts During CI
Rapid Handling of Static Analysis Meta-Alerts During CI

Source Code Repository

CI Workflow

CI Server

Development & Test Teams

Source Code Check-In

Build Install Deploy

Report the Results

Run Static Analysis

Unit, Integration, Stress Tests

Run Tests
Rapid Handling of Static Analysis Meta-Alerts During CI
We Developed Two Types of Meta-Alert Adjudication Cascading

- For code versions 1 and 2, can a manual adjudication (e.g., true, false) for a meta-alert from v1 be applied to a meta-alert for v2?
- Imprecise cascading happens on a per-file analysis and uses regular expression and/or line numbers.
- Precise cascading means analysis across a whole program using control flow, data flow, and type flow.
Goal: Enable **practical** automated classification, so all meta-alerts can be addressed
FY21 Results and New Capability

• **Adjudication cascading**
  - Diff-based cascading integrated in SCAIFE
  - Generated diff-based cascading test data
  - Collaboration with Dr. Le’s team from Iowa State University:
    • Precise cascerader development
    • Generated test data, compared precise and diff-based cascading

• **SCAIFE v3 release**
  - Enhanced performance metrics collection
    • Experiment mode, auto-setup experiments with configuration files + datasets, metrics collection, auto-end, and data export
    • Metrics include classifier precision and recall; counts of adjudicated vs. high-confidence predicted; latencies; CPU, bandwidth, and memory use
  - Java test suites now fully usable by SCAIFE
  - CI integration and CI demo (also in FY21 SCAIFE v2 release)
FY21 Results and New Capability

- **SCAIFE release test results and analysis**
  - SEI CI experts did CI demo and verified SCAIFE updates with their CI server and Git repository
  - External collaborators verified CI demo and performance experiment functionality in most systems
  - SEI and collaborator performance data generated: latency, classifier precision, and recall for particular datasets
  - Analysis: Latency of classifier creation, precision, and recall are topics for future system design enhancements

- **GitHub publications of SCAIFE API and SCAIFE UI module (SCALE) code**
  - [https://github.com/cmu-sei/SCAIFE-API](https://github.com/cmu-sei/SCAIFE-API)
  - [https://github.com/cmu-sei/SCALe/tree/scaife-scale](https://github.com/cmu-sei/SCALe/tree/scaife-scale)
Want to Become a Tester?

Involves

1. Start up SCAIFE (docker-compose command on Linux or Mac machine)

Start SCAIFE
Want to Become a Tester?

Involves

1. Start up SCAIFe (docker-compose command on Linux or Mac machine)
2. Select experiment from list (rest auto-fills)
Want to Become a Tester?

Involves

1. Start up SCAIfFe (docker-compose command on Linux or Mac machine)
2. Select experiment from list (rest auto-fills)
3. Adjudicate meta-alerts, working from top of list (list reorders after adjudications)
Want to Become a Tester?

Involves

1. Start up SCAIFe (docker-compose command on Linux or Mac machine)
2. Select experiment from list (rest auto-fills)
3. Adjudicate meta-alerts, working from top of list (list reorders after adjudications)
4. Send exported 3 files back if possible, or provide qualitative feedback on testing

Experiment End: Exports 3 Files

datahub_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_Random_Forest_and_K-Nearest_Neighbors.json
scale_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_Random_Forest_and_K-Nearest_Neighbors.json
stats_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_Random_Forest_and_K-Nearest_Neighbors_2021-09-03_22:58:46_324347.json
Benefits and Impact of Rapid Handling of SA Results During CI

• Automated SA handling – adjudicate more SA results in less time
• Integrated with CI – keep your code secure, build by build; fix earlier and cheaper
• Develop more secure software in less time at lower cost
Questions?

Contact: Dr. Lori Flynn
Senior Software Security Researcher
info@sei.cmu.edu

Dr. Lori Flynn (PI)   Joseph Yankel  Matt Sisk  Tyler Brooks  David Svoboda  Dustin Updyke
Jeffrey Mellon  Ebonie McNeil  Rhonda Brown  Lyndsi Hughes  Joseph Sible  Wei-ren Murray
Tester  Tester  Tester
Research Review 2021

Project Concept: Integrated SCAIFE-ACR
Project Concept: Integrated Static Analysis Classification and Automated Code Repair for CI

**Problem:** DoD organizations that develop code or analyze code security need to make code more secure, with as little costly manual effort as possible. Automated code repair (ACR) tools can fix some code flaws, and automated SA classifiers can save manual work adjudicating SA results, but they may not work well together as-is.

**Solution:** A system that can modularly incorporate a wide variety of SA classifiers and ACRs that increases the percent of high-severity SA results addressed automatically after applying ACR, designed for CI. “Automatically” means “automatically repaired” or “automatically classified with confidence 70% or greater” (provided that the classifier has a precision and recall 70% or greater).

**Approach:** Building on what we’ve learned and the tools from the previous 6 years of Line-funded projects (SEI ACR and SCAIFE), we will develop a modularly integrated SCAIFE-ACR system for CI, then use it to measure impact of SEI ACR on the percent of high-severity SA results addressed automatically. We test it with open-source code and collaborators use it in their systems on their code. Also, novel use of ACR fix data to improve classifier predictions, and measure effectiveness with classifier precision and recall comparisons.

1. **High-severity SA results:** warnings for top 25 dangerous CWE and CERT coding rules with severity level 3
2. **SCAIFE:** modular SA classification system with GUI developed by L. Flynn’s SEI projects
3. **ACR:** uses a semantic representation of code to fix code flaws
4. **SEI ACR:** ACR tooling for memory safety in C developed by W. Klieber’s SEI projects
Auto-Labeled ACR Fixes

1. An ACR fix is made to the code (was version 1, now v2 in new branch)

<table>
<thead>
<tr>
<th>ACR possible?</th>
<th>Priority</th>
<th>Auto-Repair?</th>
<th>Manual Adjudication</th>
<th>Classifier Confidence True (%)</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>8890</td>
<td>YES</td>
<td></td>
<td></td>
<td>CWE-190</td>
</tr>
<tr>
<td>yes</td>
<td>8889</td>
<td>NO</td>
<td></td>
<td></td>
<td>INT31-C</td>
</tr>
<tr>
<td>yes</td>
<td>8888</td>
<td>YES</td>
<td></td>
<td></td>
<td>CWE-191</td>
</tr>
<tr>
<td>yes</td>
<td>8887</td>
<td>YES</td>
<td></td>
<td></td>
<td>CWE-79</td>
</tr>
<tr>
<td>yes</td>
<td>8886</td>
<td>YES</td>
<td></td>
<td></td>
<td>CWE-787</td>
</tr>
<tr>
<td>yes</td>
<td>8885</td>
<td>YES</td>
<td></td>
<td></td>
<td>CWE-125</td>
</tr>
</tbody>
</table>

2. The ACR code fixes (v2) are committed to the remote repository. Then, the CI server builds and tests the code, and sends results back to the development+test team and to SCAIFE.

3. SCAIFE fuses SA results into meta-alerts. If last code push is an auto-repair, SCAIFE checks if a meta-alert for the repaired condition re-appears on matched lines* of repaired code. If yes, it auto-labels the meta-alert False if fix marked ‘reliable’ by the ACR. New feature “auto-repair” for classifier. No ACR auto-labels True.

SEI ACR does not prove the code v1 meta-alert was True, but it fixes many memory safety violations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>8885</td>
<td>YES</td>
<td></td>
<td></td>
<td>N/A</td>
<td>N/A</td>
<td>10</td>
<td>dir4/FileX</td>
</tr>
</tbody>
</table>

Code v1, ACR-fixed meta-alert

N/A 7000 PREVIOUS 80 CWE-125 FALSE 10 dir4/FileX

Code v2, same location and condition: meta-alert auto-labeled FALSE

* Lines may be matched using POSIX diff program, possibly enhanced with extra matching information related to the ACR system. E.g., a memory access that previously took 1 line might be expanded by the ACR to take 3 lines in one file plus a new function in another file, and any of those locations would count as a match.

4. The new labeled data (meta-alert auto-labeled false and associated data) is used to improve the SA classifier predictions for all remaining not-yet-adjudicated meta-alerts, using adaptive heuristics and/or occasional classifier retraining.

---

25
Rapid Adjudication of Static Analysis Results During CI
© 2021 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.
Interest in SCAIFE-ACR Integration?

To do the integration work described, we would need more funding. Would you or your org be interested? If so, please contact us!
Invitation to Test

We invite you to test SCAIFE and ACR tools:

• Full SCAIFE system release limited to DoD and DoD contractors (Distribution D)
• Testing does *not* have to include data sharing
• SCAIFE classification performance release needs testers ASAP.
• If interested please contact us: flynn@cert.org and weklieber@cert.org

Deployment and testing support by SCAIFE:

• release system Docker-containerized, with configuration files (ports, URLs, names) to ease integration in wide variety of systems
• comes with documentation, much-extended in last year per collaborator feedback
• hands-on demos and tutorials, for quick start

Deployment and testing support by ACR:

• Coming soon: release system to be Docker-containerized
• Coming soon: hands-on demos and tutorials, for quick start