Static Analysis Classification, SCAIFE, and Your Work

Jan 24, 2023

Lori Flynn, PhD

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM23-0048

Static Analysis Classification, SCAIFE, and Your Work

Source Code Analysis Integrated Framework Environment (**SCAIFE**)

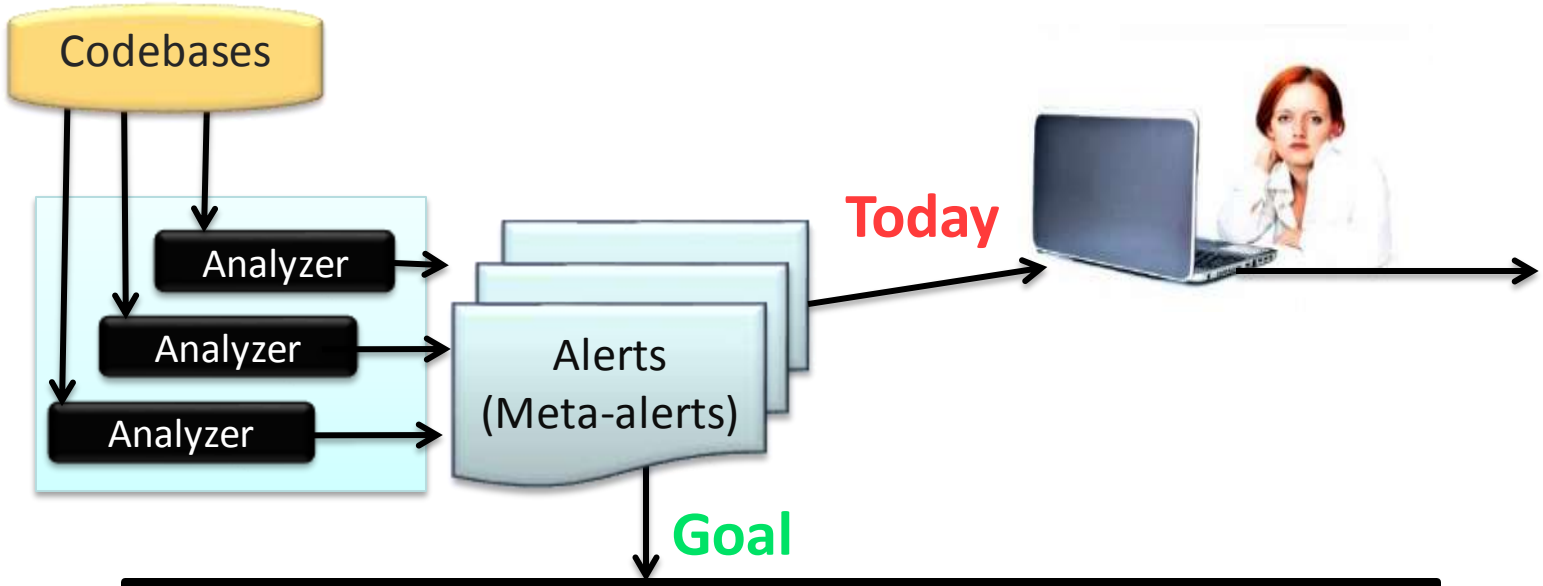
Today, we'll talk about:

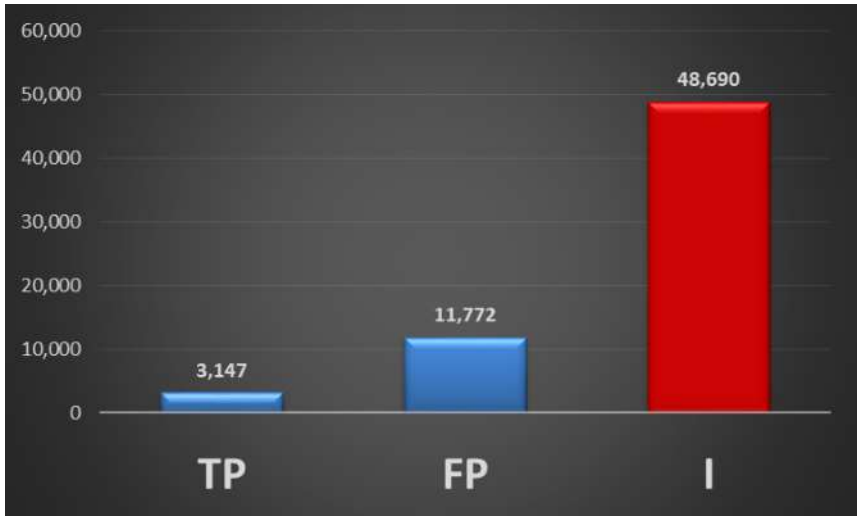
- Static analysis classification
- Features in SCAIFE that you might want to use / take / improve upon
  - ➢ Includes a demo
- The issues you are facing, that we might be able to provide resources for

SCAIFE
classification system

Problem: too many static analysis alerts
Solution: automate handling

A **meta-alert** is a static analysis result for a particular line number, filepath, and code flaw condition (e.g., CWE-190).





CI-optional systems that
precisely and with high recall, classify at least as many manually-adjudicated meta-alerts as:

Expected True Positive (e-TP) or
Expected False Positive (e-FP),
and
the rest as Indeterminate (I)

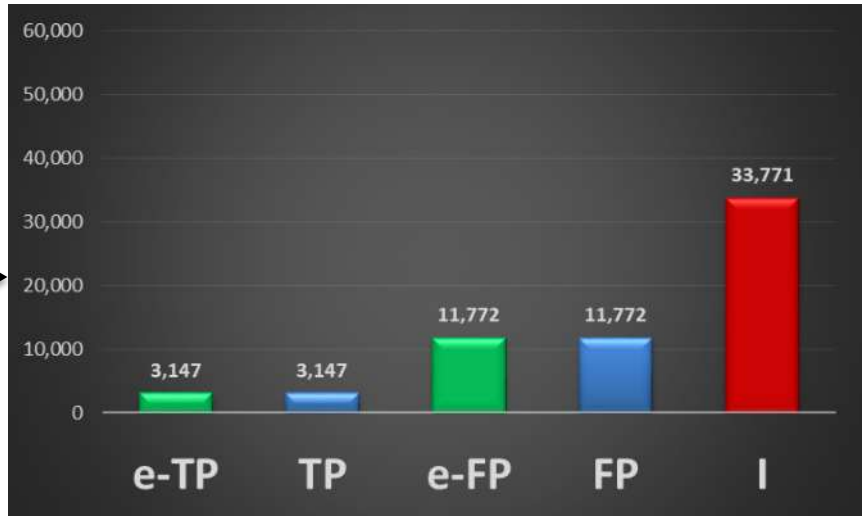


Image of woman and laptop from http://www.publicdomainpictures.net/view-image.php?image=47526&picture=woman-and-laptop "Woman And Laptop"

Research Roadmap: SA Classification

SA classification research: Existing topic for decades. For example:

- 85% accuracy in [Ruthruff]: FindBugs, Logistic Regression, adaptive using code-fix decisions
- 62% precision for top 50 alerts, using locality, flaw type, code version number [Williams]
- 89% precision for *jdom*. Machine learning using Weka tool [Heckman B]
- 9% improvement in cross-project defect prediction, using semantic features [Wang]
- Adaptive algorithms
  - [Heckman] ARM: 81% true positive alerts after investigating only 20% of alerts (vs. avg. of 50 random orderings found 22% after investigating 20%)
  - [Kremenec] Feedback-Rank: 2-8x improvement of performance ratio over random

[Baishakhi] Ray, Baishakhi, et al. "On the naturalness of buggy code." ICSE, 2016.
[Heckman] Heckman, Sarah Smith. "Adaptively ranking alerts generated from automated static analysis.“, Crossroads, 2007.
[Heckman B] S. Heckman, L. Williams, On establishing a benchmark for evaluating static analysis alert prioritization and classification techniques, Empirical Software Engineering and Measurement, 2008, pp. 41–50.
[Hoole] A. Hoole et al. "Improving vulnerability detection measurement" ICSE, 2016.
[Kim] S. Kim, M.D. Ernst, Prioritizing warning categories by analyzing software history, International Workshop on Mining Software Repositories, 2007, p. 27.
[Kremenec] T. Kremenek, K. Ashcraft, J. Yang, D. Engler, Correlation exploitation in error ranking, FSE, 2004, pp.83–93.
[Wang] Wang, Song, Taiyue Liu, and Lin Tan. "Automatically learning semantic features for defect prediction." ICSE, 2016.
[Williams] C. Williams et al., Automatic mining of source code repositories to improve bug finding techniques, Trans. SE 2005.
[Ruthruff] J. Ruthruff et al. "Predicting accurate and actionable static analysis warnings: an experimental approach." ICSE, 2008.
[Zhang] Zhang, Feng, et al. "Cross-project defect prediction using a connectivity-based unsupervised classifier." ICSE, 2016.

Research Roadmap: SA Classification

Current SCAIFE prototype: features and capabilities from multiple SA classification research projects at SEI

Take from SCAIFE: Organizations could tailor a plugin (e.g., CodeDX, SonarQube, etc.) for classifiers, features, other SA frameworks, other system designs and test with datasets.

FY16-19 SA Classification Research Detail

FY16

- Issue addressed: classifier accuracy
- Novel approach: use **multiple static analysis tools as features**
- Result: increased accuracy

FY17

- Issues addressed: **data quality**, **too little labeled data** for accurate classifiers for some conditions (e.g., CWEs, coding rules)
- Novel approach: **audit rules+lexicon; use test suites to automate the production of labeled** (true/false) **meta-alert data* for many conditions**
- Result: high precision for more conditions

FY18-19

- Issue addressed: **little use of automated meta-alert classifier technology** (requires $$, data, experts)
- Novel approach: **develop an extensible architecture with a novel test-suite data method**
- Result: enabled **wider use of classifiers (with less $$, data, experts)** with an extensible architecture, API, software to instantiate architecture, and adaptive heuristic research

* By the end of FY18, ~38K new labeled (T/F) meta-alerts from eight SA tools on the Juliet test suite (vs. ~7K from CERT audit archives over 10 years)

Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

FY20-21 SA Classification Research Detail

- Issue addressed: It takes too much time to adjudicate static analysis results during continuous integration (CI).
- Novel approach: Develop modular CI-enabled design using SA **classifiers** and different **cascading,** enable performance experiments
- Results (highlights)
  - Designed, implemented, and tested CI-SCAIFE system integration
  - Both diff-based and precise cascading developed, tested
  - Enhanced performance metrics collection and auto-setup of experiments
  - Some experiment results and collaborator testing
  - Multiple SCAIFE System, SCAIFE API, and SCAIFE UI Module (SCALe) releases

Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

SCAIFE features you may want to use

- **Static analysis automated classification**: make predictions about confidence a warning is true or false
- **Auto-label test suite codebases using NIST SARD manifests**: generate more labeled static analysis data quickly, for a wide variety of code flaws
- **Gather classification prediction performance data at key points and periodically**. This uses labeled data, making predictions on holdout labeled data. (the existing adjudicated static analysis results, manual and test suite)
- Depending on the performance, **use the classification predictions to order and/or filter not-yet-adjudicated static analysis results**
- **Fusion (meta-alerts)**
- **Cascading adjudications**
- **CI integration**
- **Prioritization formulas**

Static analysis training you might be interested in

You might be interested in **static analysis training** we've developed (or possibly tailoring training to your needs)**:**

- Using rules and lexicon for consistent adjudication of static analysis results
- CERT rules and CWE
- Using SCAIFE
- Using SCALe

Online SCAIFE/SCALe training available with a CAC card at https://moodle.cyberforce.site/mod/page/view.php?id=49389

- VM-based hands-on guided demos of some functionality

SCAIFE Static Analysis Classifiers Detail

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

Labeled static analysis meta-alerts 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.

Built-in options:

- Classifiers: XGBoost, Random Forest, LightGBM
- Active Learning (adaptive heuristics): Similarities, K-Nearest Neighbors, and Label Propagation
- Hyper-parameter optimization: Bayesian Optimization

SCAIFE Classification System

Designed to be used in a wide variety of systems, with many other tools

Full SCAIFE system includes all 5 modules

Modular system designed to work with different user interfaces and static analysis tools

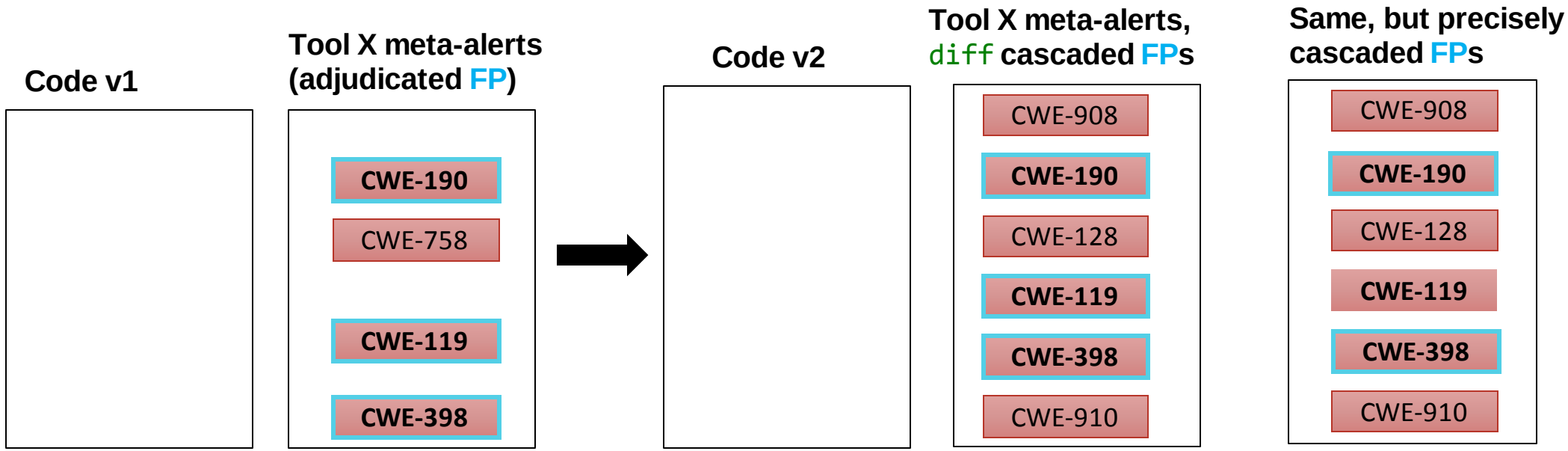
- SARIF static analysis format
- SCARF format (DHS SWAMP)
- Various tools and versions, with standard method for adding new tools

Use SCAIFE for a single code version or a codebase in a CI system

- CI system: updates to code and static analysis for the new code version

SCAIFE: 2 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 code 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.



SCAIFE Architecture

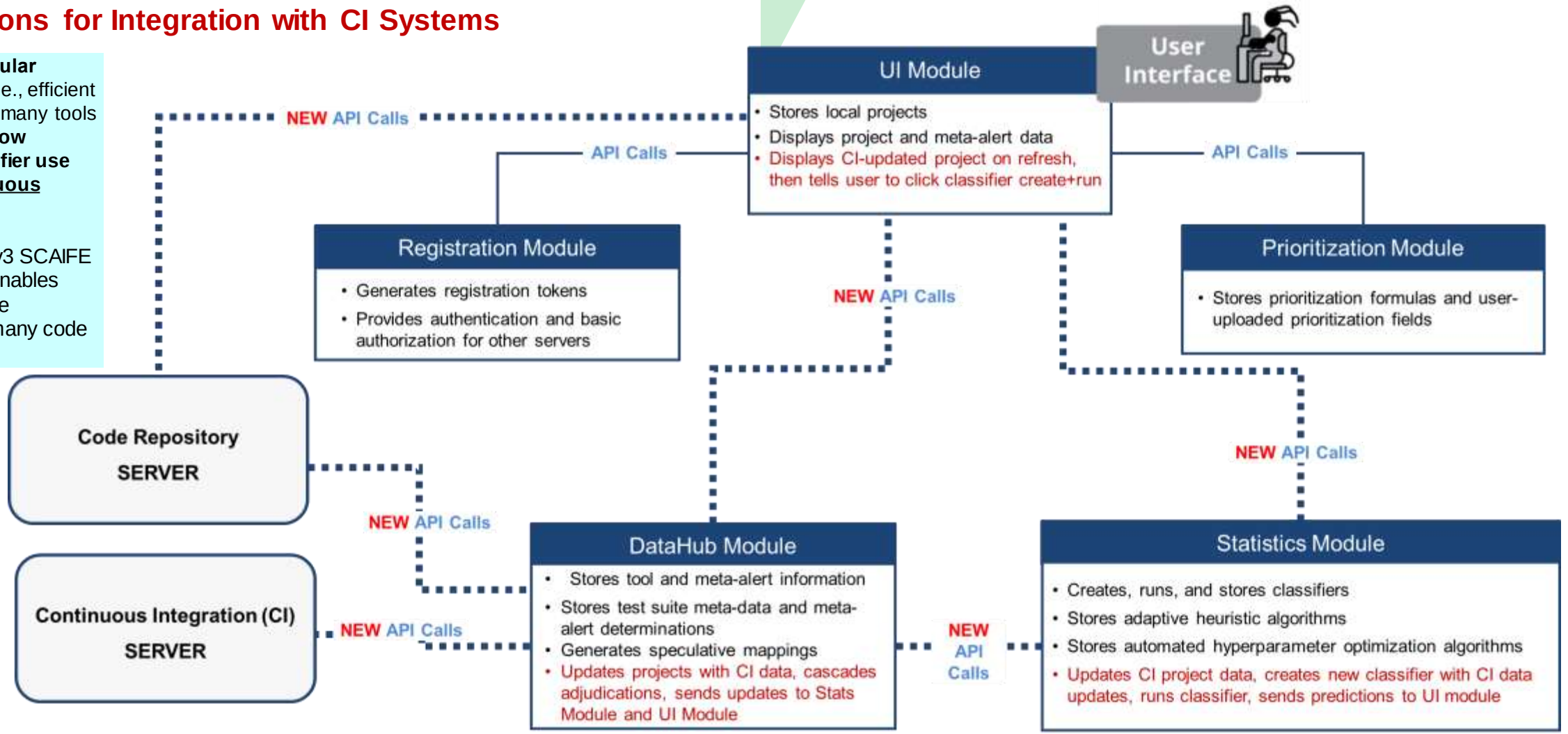
Any static analysis tool can instantiate APIs to become a UI Module. For example

- SEI SCALe
- DHS SWAMP
- CCDC C5ISR SwAT
- Other aggregator tools
- Single static analysis tools

Modifications for Integration with CI Systems

SCAIFE's modular architecture (i.e., efficient integration with many tools and systems) **now enables classifier use <u>during continuous integration.</u>**

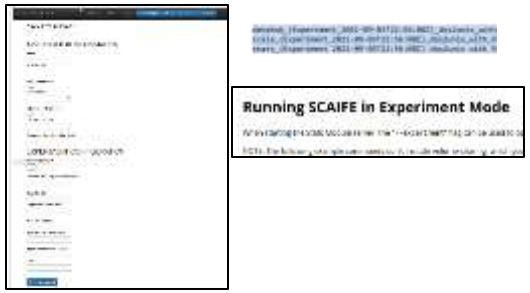
The OpenAPI v3 SCAIFE API definition enables automated code generation in many code languages

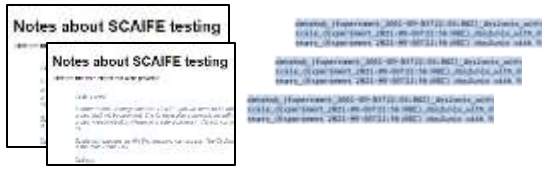


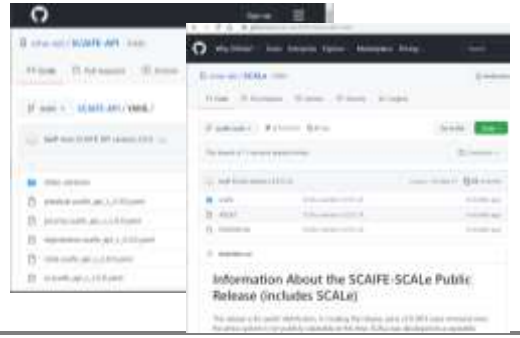
Goal: Enable **practical** automated classification, so all meta-alerts can be addressed

FY21 Select Artifacts

scaife.online.3.0.0.tar.gz







- **Paper** "Test suites as a source of training data for static analysis alert classifiers" by Lori Flynn, William Snavely, and Zachary Kurtz to ICSE-associated Conference on Automation of Software Test (AST) 2021 https://conf.researchr.org/home/icse-2021/ast-2021 and video https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=737855
- **SCAIFE v3 release**
  - Contains much-enhanced performance metrics collection:
    - Experiment mode: auto-setup experiments with configuration files + datasets, collect metrics, auto-end, and export data
    - Metrics include (among others): classifier precision and recall, counts of adjudicated vs. high-confidence predicted, and key step latencies, CPU use (max, avg), bandwidth use (max, avg), memory use (max, avg)
  - Java test suites now fully usable by SCAIFE
- **SCAIFE release test results and analysis:**
  - SEI CI experts did the CI demo, provided feedback (Lyndsi Hughes and Joe Sible)
  - External collaborator testing and feedback
- **GitHub publications of SCAIFE API** https://github.com/cmu-sei/SCAIFE-API
- **GitHub publications of SCAIFE UI module** (SCALe) **code** at https://github.com/cmu-sei/SCALe/tree/scaife-scale

Your Thoughts on How Any of This Might Help Your Org

1. What are your related issues?
2. Which of those issues are top priorities?
3. Would you be interested in using tools, methods, code, or APIs that I talked about? Which ones?
4. What type of modifications to the current SCAIFE system would be most helpful?
5. Would you be willing to share info about your use of any of this? (That would help me a lot!)
   - High-level feedback fine
     - E.g., which artifact you used, if it helped you find and fix code flaws
   - Detailed feedback even better.
     - E.g., “We are using it as part of DevSecOps workflow by 200 developers and compared to the same effort the previous year without it, it resulted in 25% more found and fixed code flaws that are top-25 CWE.”

Invitation to Test

We invite you to test and/or extend SCAIFE:

- Full SCAIFE system release limited to DoD and DoD contractors (Distribution D)
  - If interested, contact lflynn@cert.org
- Also online hands-on demo version available with CAC

Deployment and testing support by SCAIFE:

- release system Docker-containerized, with configuration files (ports, URLs, names) to ease integration in variety of systems
- comes with documentation
- hands-on demos and tutorials, for quick start
- Development support includes set of CI tests

Thanks + Contact Info

Thank you for listening!

Questions?

Lori Flynn, PhD
lflynn@sei.cmu.edu

Carnegie Mellon University
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612