

## Research Review 2021

# 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](mailto:permission@sei.cmu.edu).

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM21-0835

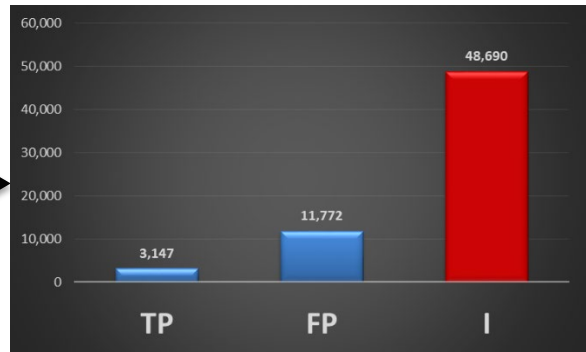
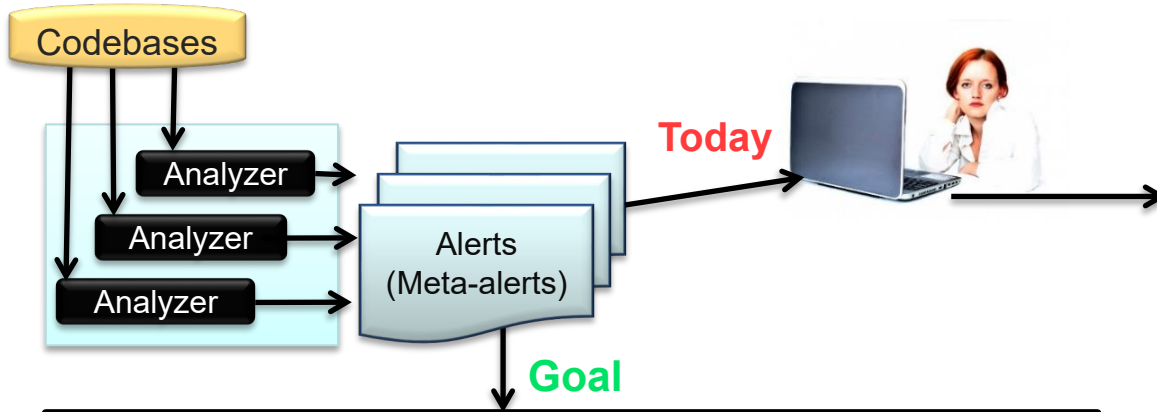
# Rapid Handling of Static Analysis Results During CI

**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.

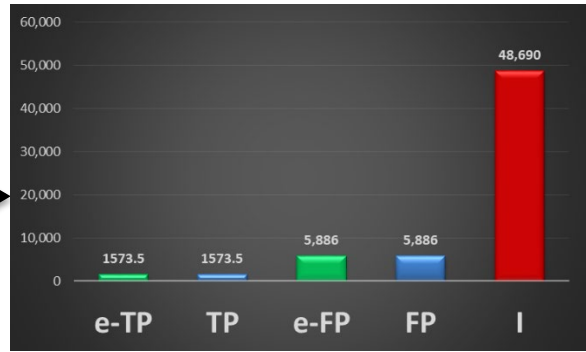
# SCAIFE's Automated Classification System



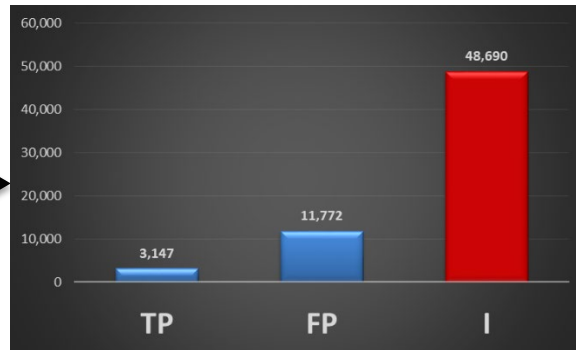
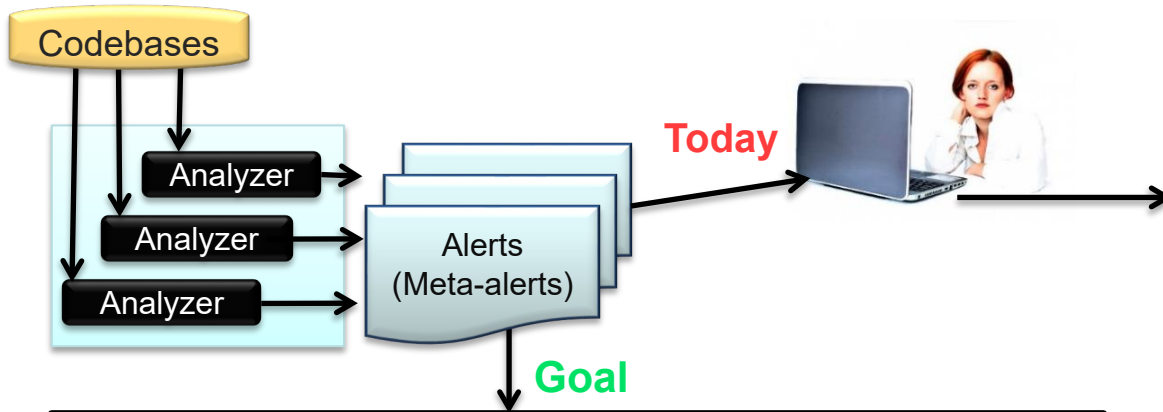
A meta-alert is an SA result for a particular line number, filepath, and code flaw condition (e.g., CWE-190).

SCAIFE classifies meta-alerts as

- Expected True Positive (e-TP)
- Expected False Positive (e-FP)
- Indeterminate (I)



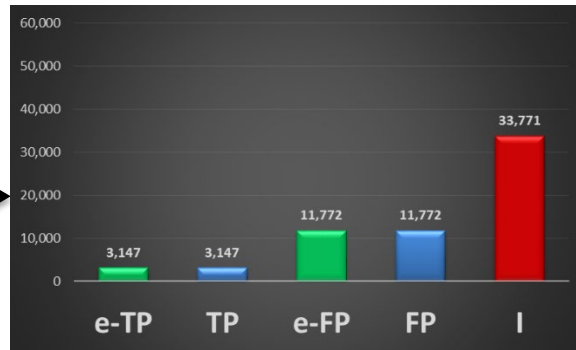
# SCAIFE's Automated Classification System



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.

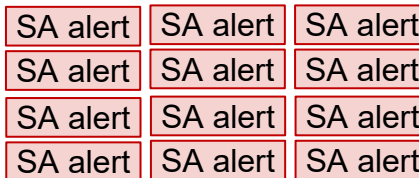
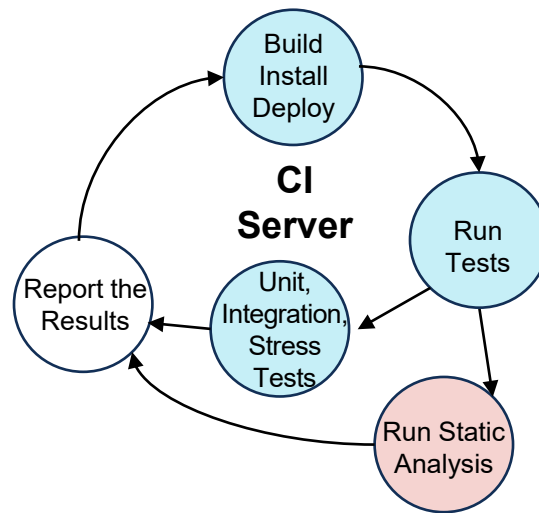
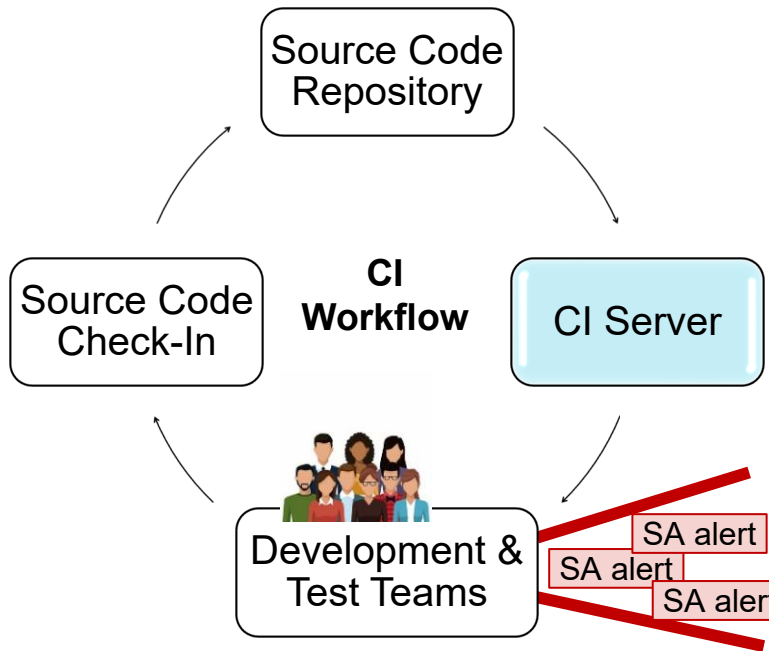
# SCAIFE's Classification System

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

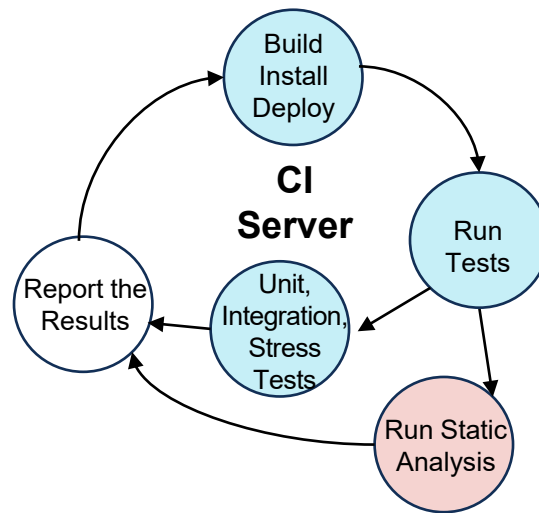
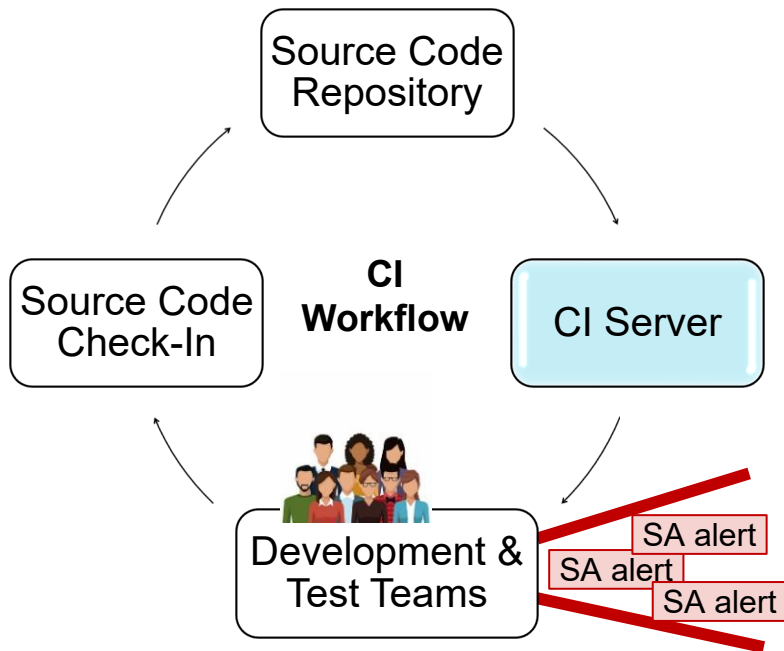
- with different user interfaces and SA tools
  - SARIF static analysis format
  - SCARF format (DHS SWAMP)
  - Various tools and versions, with a standard method for adding new tools
- Docker containers usable in a wide variety of systems

# Rapid Handling of Static Analysis Meta-Alerts During CI



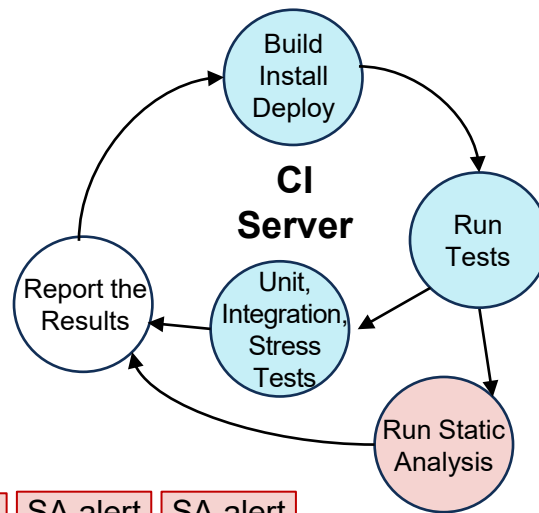
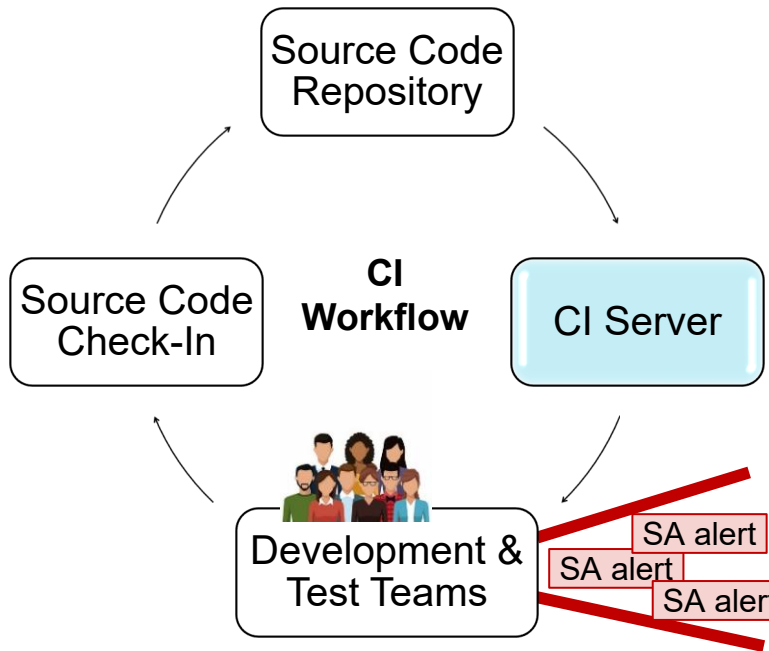


# Rapid Handling of Static Analysis Meta-Alerts During CI

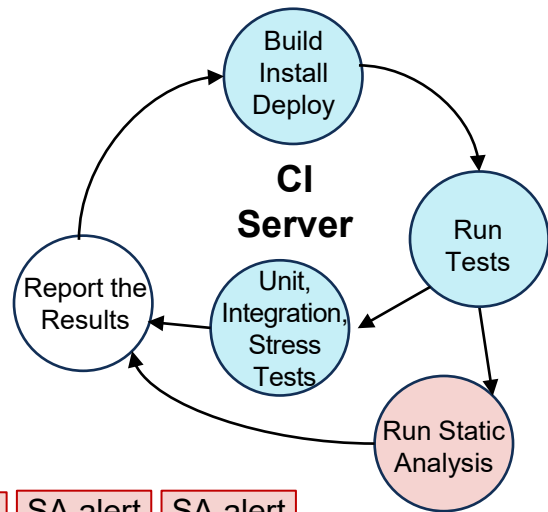
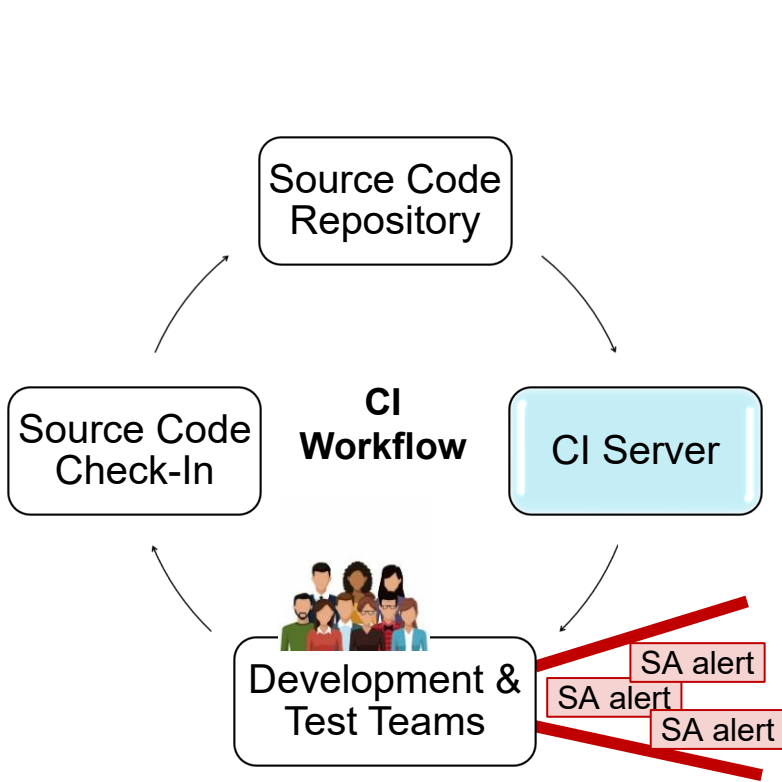


SA alert	SA alert	SA alert	SA alert	SA alert	SA alert
SA alert	SA alert	SA alert	SA alert	SA alert	SA alert
SA alert	SA alert	SA alert	SA alert	SA alert	SA alert
SA alert	SA alert	SA alert	SA alert	SA alert	SA alert

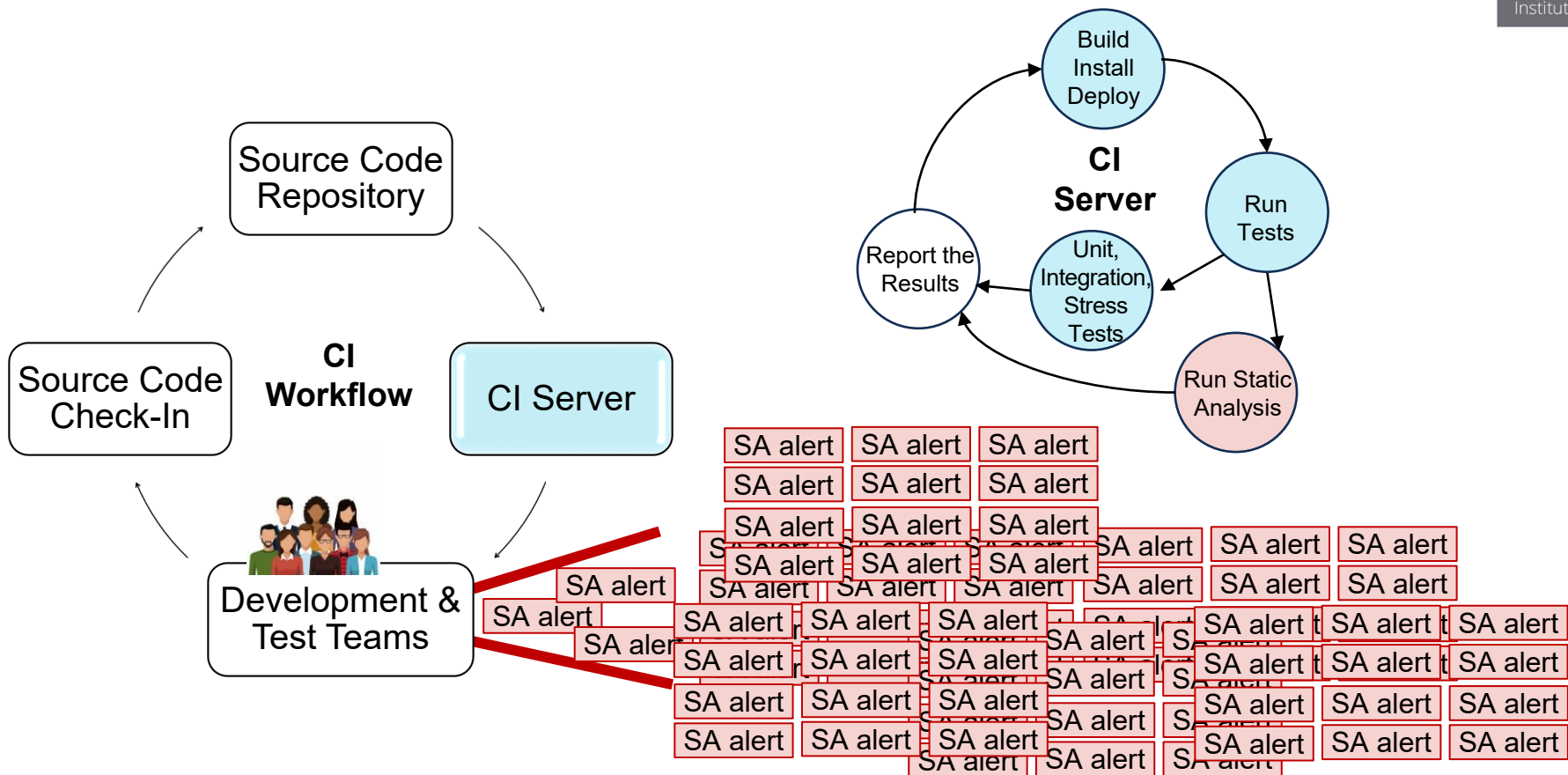
# Rapid Handling of Static Analysis Meta-Alerts During CI



# Rapid Handling of Static Analysis Meta-Alerts During CI

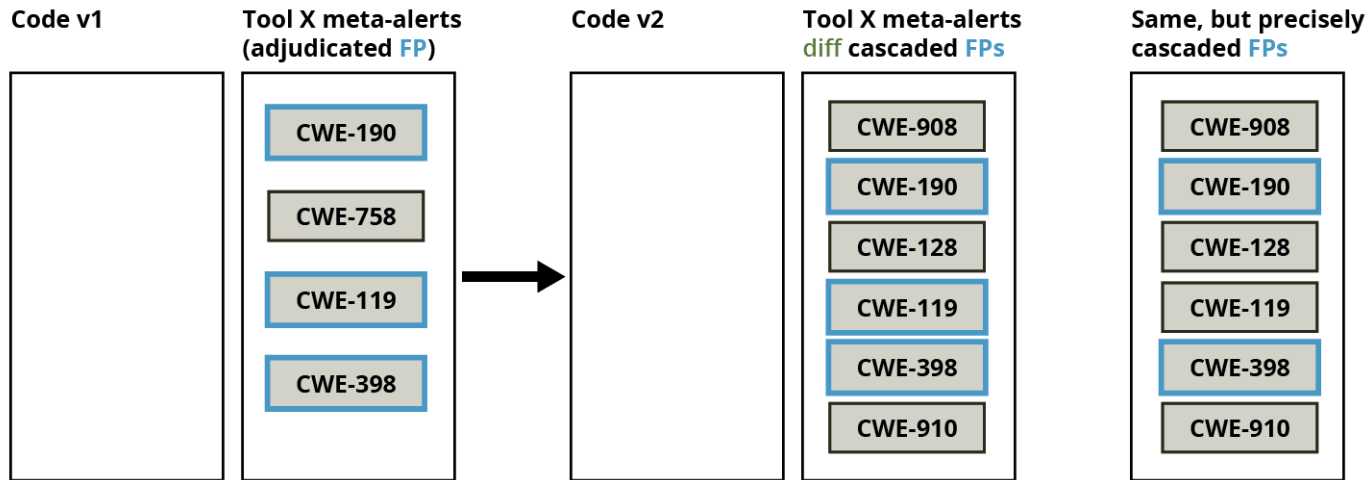


# Rapid Handling of Static Analysis Meta-Alerts During CI



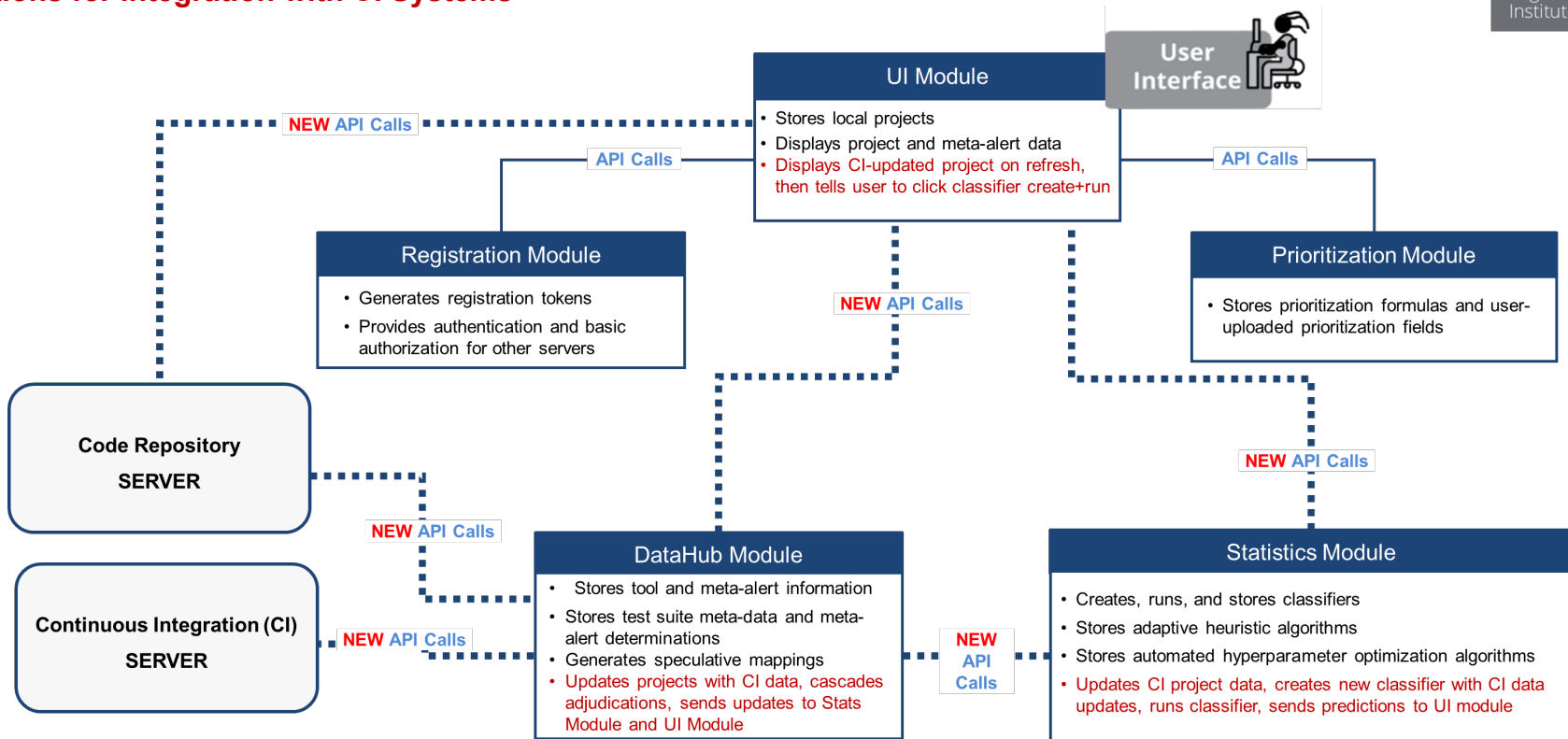
# We Developed Two Types of Meta-Alert Adjudication Cascading

- 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.



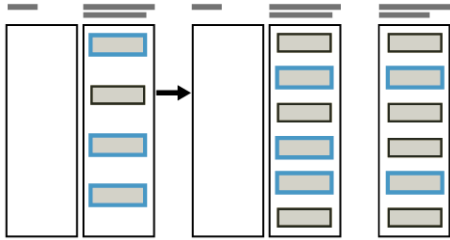
# SCAIFE Architecture

## Modifications for Integration with CI Systems



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 cascader 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)

## [scaife.online.3.0.0.tar.gz](#)

### Auto-exported experiment metrics files

```
datahub_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with  
scale_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_R  
stats_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_R
```

# FY21 Results and New Capability

## Notes about SCAIFE testing

Here are the instructions that were provided:

Testing prep:

In order to test CI integration with SCAIFE you will need to be able to

```
datahub_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_scale_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_R_stats_(Experiment_2021-09-03T22:56:08Z)_dos2unix_with_R
```

Options:

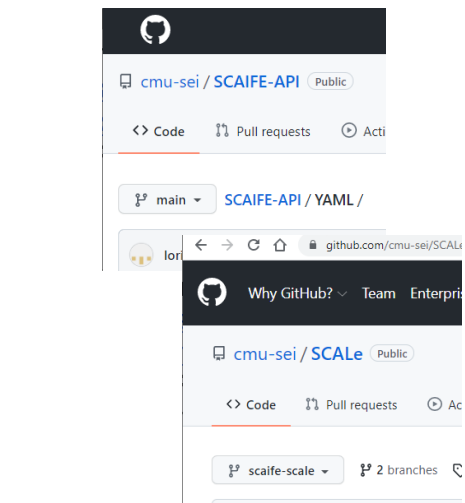
## • SCAIFE release test results and analysis

- 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/SCALE/tree/scaife-scale>



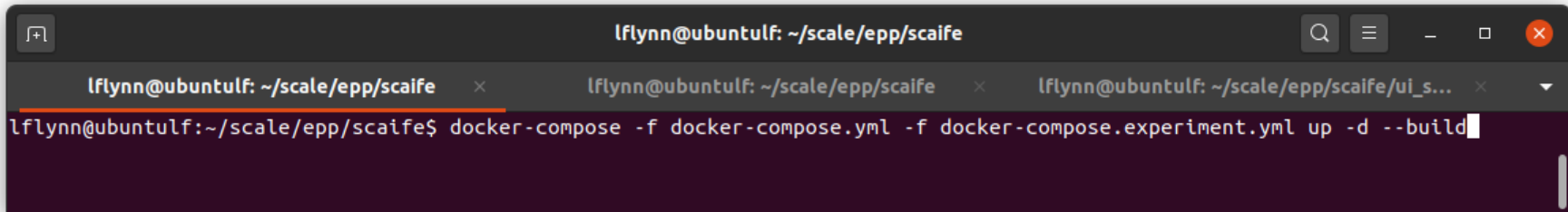


# Want to Become a Tester?

## Involves

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

## Start SCAIFE

A terminal window with a dark background and light text. The title bar shows 'lflynn@ubuntulf: ~/scale/epp/scaife'. There are three tabs open: 'lflynn@ubuntulf: ~/scale/epp/scaife', 'lflynn@ubuntulf: ~/scale/epp/scaife', and 'lflynn@ubuntulf: ~/scale/epp/scaife/ui\_s...'. The main terminal area shows the command 'lflynn@ubuntulf:~/scale/epp/scaife\$ docker-compose -f docker-compose.yml -f docker-compose.experiment.yml up -d --build' with a cursor at the end.

```
lflynn@ubuntulf:~/scale/epp/scaife$ docker-compose -f docker-compose.yml -f docker-compose.experiment.yml up -d --build
```

# 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)

## Select Experiment in Drop-Down

SCAIFe Analysis Tool SCAIFe at CERT Definitions Help Copyright [SCAIFe Logout: scaleUser](#) [Disconnect from SCAIFe](#)

### New Experiment

#### ADJUDICATOR INFORMATION

Name

Organization

Coding Experience  
   
 Years Coding

Adjudication Experience  
   
 Years Adjudicating

Experiment Type Meta Alert Order

#### EXPERIMENT CONFIGURATION

Select Experiment:

Classifier Training Additional Project

Classifier Type

Adaptive Heuristic Type

Meta-Alert Filtering

Meta-Alert Priority Scheme

Hyper-Parameterization Type

Fused

[Create Experiment](#)

# 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)

## Adjudicate Meta-alerts (Do Top of List)

Project: dos2unix/rosecheckers project

Order by: Path ASC, Line ASC Fused View: On

New Alert + Condition

All IDs: Verdict: Unknown Previous: Path: Line: Checker: All Checkers Tool: All Tools Condition: All Taxonomy: View All Category: All Shuffle Seed: Filter Clear Filters

Showing 1 to 10 of 218 | Meta-alerts per page: 10 Go

Select all 218 Meta-alerts: SCAIFe Mode: Connected Run Classifier: automation: Random Forest 0 Cursify Manual Verdicts: 20 Predicted Verdicts: 0

ID	Flag	Verdict	Supplemental	Notes	Previous	Path	Line	Message	Checker	Tool	Condition	Title	Label	Confidence	Category	AlertCondition Pri	Sev	Lik	Rem	Pri	Lev	CWE_Lik
381 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	265	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	
382 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	267	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	
383 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	268	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	
384 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	270	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	
385 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	276	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	
386 (d)	[ ]	[Unknown]	Edit	0	0	/dos2unix-7.2.2/common.c	277	Exclude user input from format strings	FI030-C	rosecheckers_oss	FI030-C	Exclude user input from format strings	false	0.0	I		3	3	2	18	1	

```

261 printf("-863 use 805 code page 863 (French Canadian)\n");
262 printf("-865 use 805 code page 865 (Nordic)\n");
263 printf("-f convert 8 bit characters to 7 bit space\n");
264 if (!dos2unix(program))
265     printf("-b, --keep-bom keep Byte Order Mark\n");
266 else
267     printf("-h, --keep-bom keep Byte Order Mark (default)\n");
268 printf("-c, --convcode conversion code\n");
269 convcode
270 printf("-f, --force force conversion of Binary files\n");
271 #ifdef GNU_UNICODE
272 #if defined(W32) && !defined(CYGWIN)
273 printf("-gb, --gb18030 convert UTF-16 to GB18030\n");
274 #endif
275 #endif
276 printf("-h, --help display this help text\n");
277 printf("-i, --info[=FLAG] display file information\n");
278 file files to analyze\n);
  
```

# 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](mailto: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**  
Tester



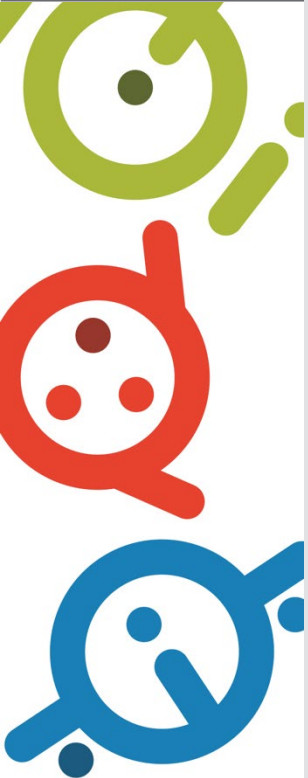
**Joseph Sible**  
Tester



**Wei-ren Murray**  
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<sup>1</sup> 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<sup>4</sup> and SCAIFE<sup>2</sup>), 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

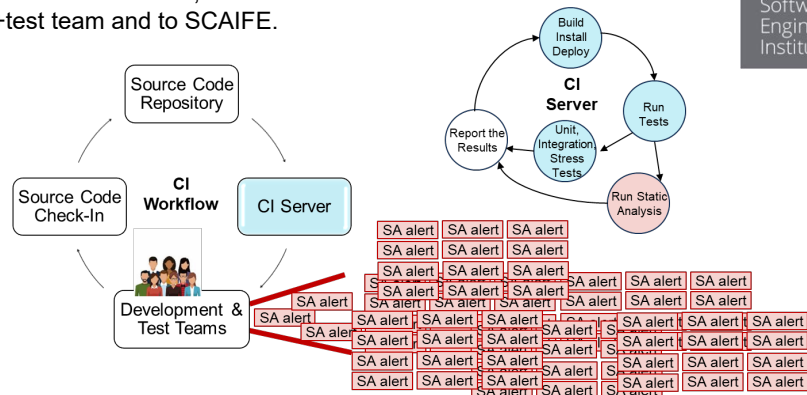


# Auto-Labeled ACR Fixes

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

ACR possible?	Priority	Auto-Repair?	Manual Adjudication	Classifier Confidence True (%)	Condition
yes	8890	YES			CWE-190
yes	8889	NO			INT31-C
yes	8888	YES			CWE-191
yes	8887	YES			CWE-79
yes	8886	YES			CWE-787
yes	8885	YES			CWE-125

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

ACR possible?	Priority	Auto-Repair?	Manual Adjudication	Classifier Confidence True (%)	Condition	Auto-Adjudication?	Line	Filepath
yes	8885	YES		88	CWE-125	N/A	10	dir4/FileX

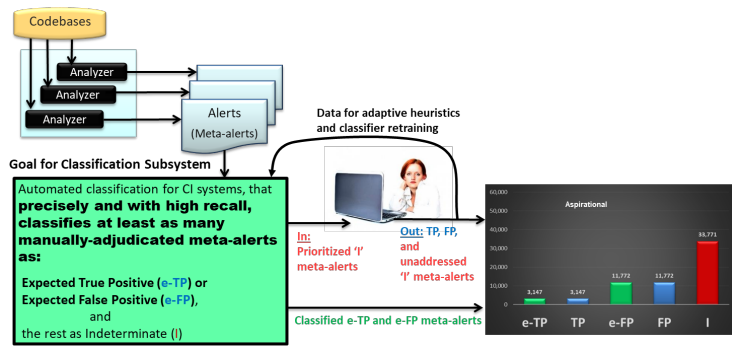
Code v1, ACR-fixed meta-alert

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.



# 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: [lflynn@cert.org](mailto:lflynn@cert.org) and [weklieber@cert.org](mailto: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