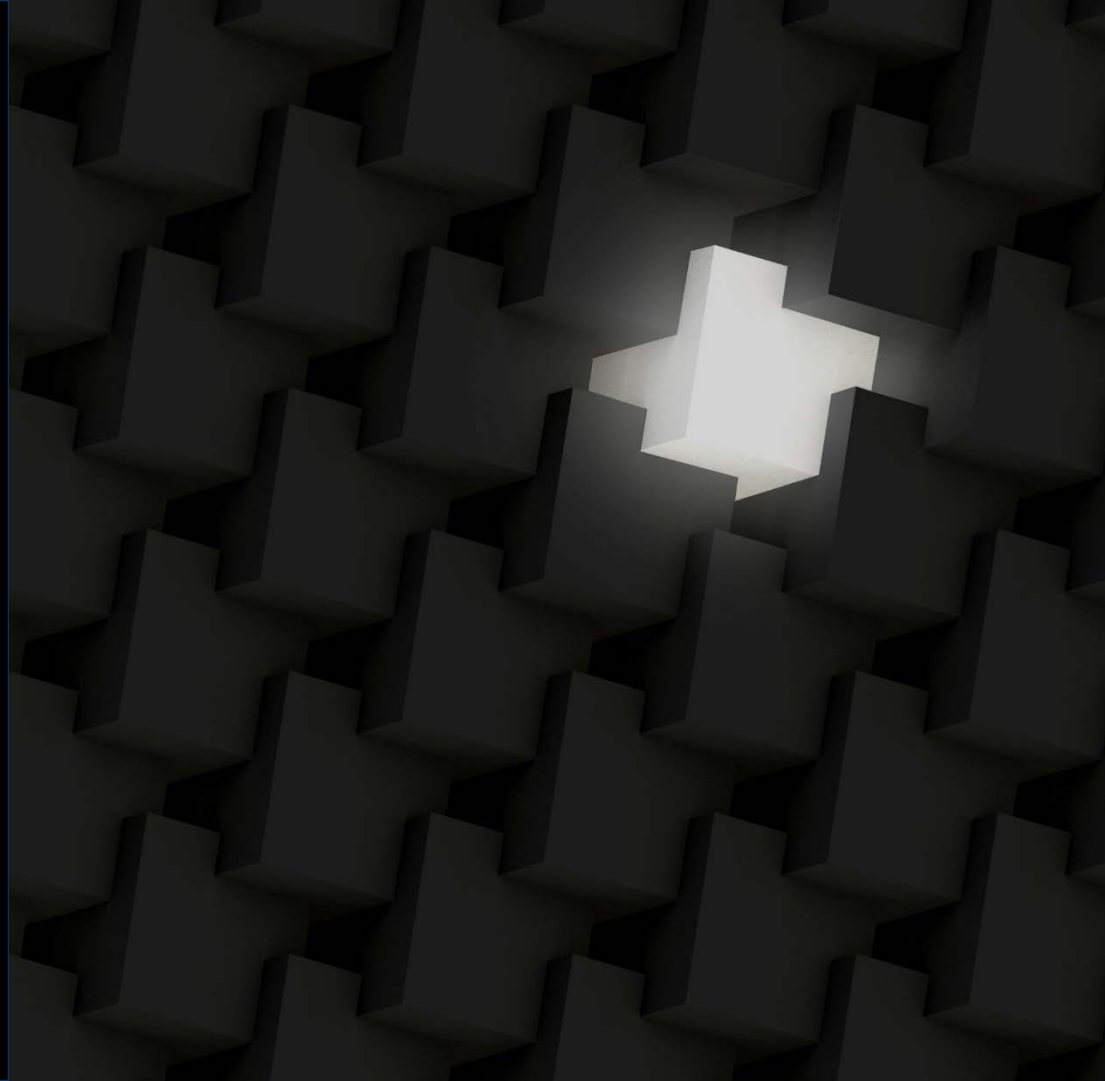
Carnegie Mellon University
Software Engineering Institute

RESEARCH REVIEW 2020

Static Analysis Classification: Line-Funded Research FY16-20

Presenter: Dr. Lori Flynn (PI)

FY20 Team: Ebonie McNeil, Matt Sisk, David Svoboda, Hasan Yasar, Joseph Yankel, David Shepard, and Shane Ficorilli



Document Markings

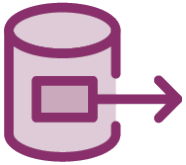
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM20-0859

Static Analysis Classification: Line-Funded Research FY16-20

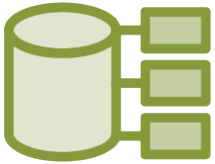
- In the last five years of line-funded research projects I've led, several techniques and tools were developed.
- Each project built on tools and techniques from the previous project.
- For more details on my FY20 project, attend my Friday Research Review presentation.
- At the conclusion of this presentation, I include ideas for combining SA classification and automated code repair.

Highlighted FY16-20 Research Focus Areas

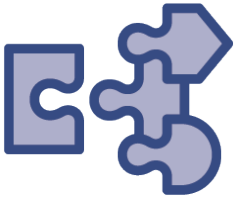


Improve classifier precision & recall

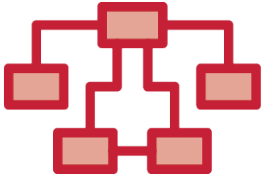
Data quality



Wide variety of labeled data



Enable classifier use via modular architecture



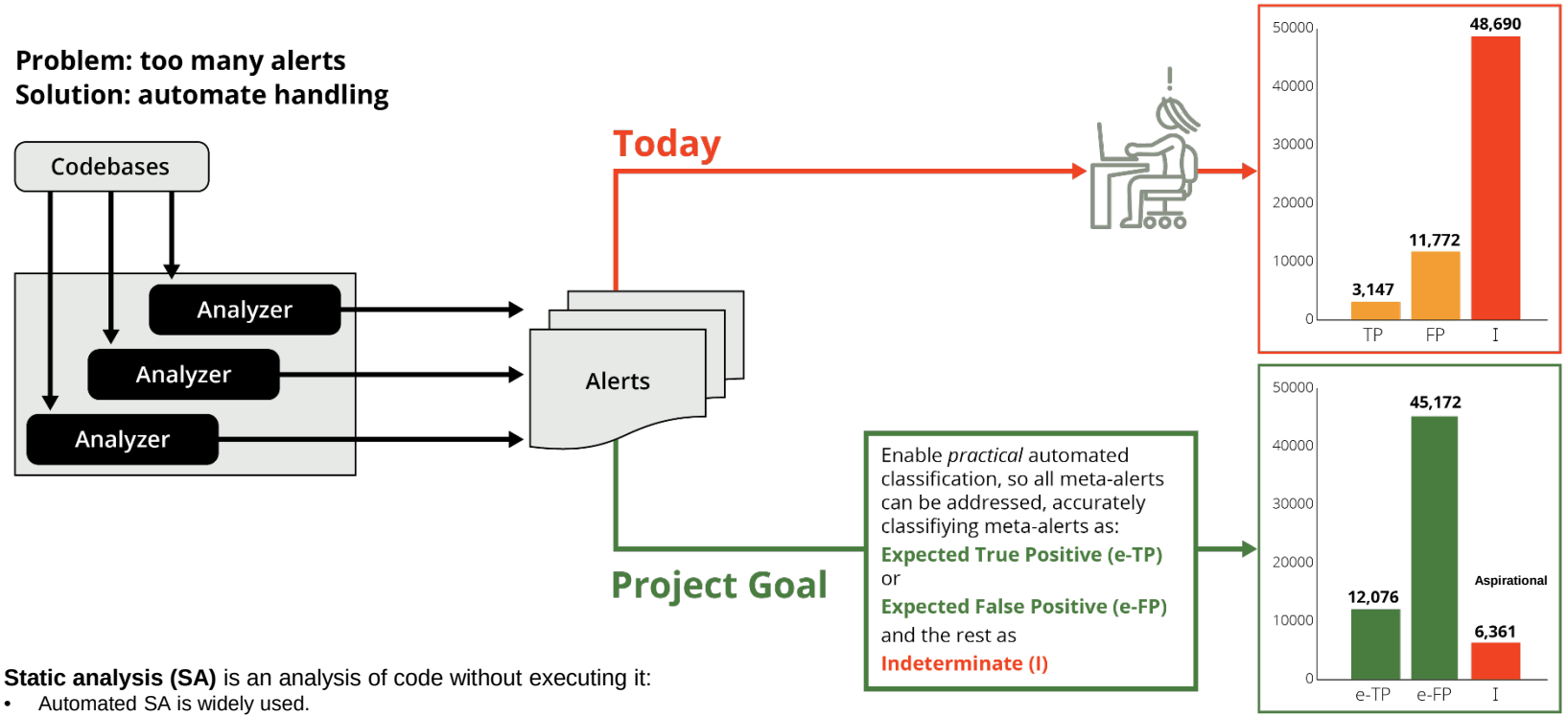
Enable classifier use in CI systems

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

Overview

Definitions: An *alert* is an SA warning (with a checker ID, line #, filepath, message), an *alertCondition* is an alert mapped to a code flaw taxonomy item (e.g., CWE-190), and a *meta-alert* is mapped to by the set of all alertConditions that differ only by checker ID. We do adjudication and classification at the meta-alert level.

Problem: too many alerts
Solution: automate handling



Static analysis (SA) is an analysis of code without executing it:

- Automated SA is widely used.
- It is a normal part of testing by DoD and commercial organizations.

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

RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

Five Years in Two Slides

FY16-19 Static Analysis Meta-Alert Classification Research

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 $$, data, experts)
- Novel approach: **develop an extensible architecture with a novel test-suite data method**
- Result: **wider use of classifiers (less $$, 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 Static Analysis Meta-Alert Classification Research

FY20 (of a two-year project, FY20-21)

- Issue addressed: It takes too much time to adjudicate (i.e., audit) static analysis meta-alerts during continuous integration (CI).
- Novel approach: During CI builds, use **classifiers** with **precise cascading** and **CI/CD features**.
- Results
  - Design for CI-SCAIFE system integration
  - SCAIFE System v1 release (classifier defined, run, and results can be viewed from [G]UI module)
  - Defined cascading API
  - Less-precise cascading using the API
  - Test results for less-precise cascading
  - Significant progress on CI-SCAIFE system integration development
  - Deployment and testing by DoD collaborators (multiple rounds)
  - A published RC_Data open dataset for improved classifier research
  - APIs, technical manuals, and SCALe public publication
- FY21 plan: a precise cascading algorithm, improved classifiers, full integration

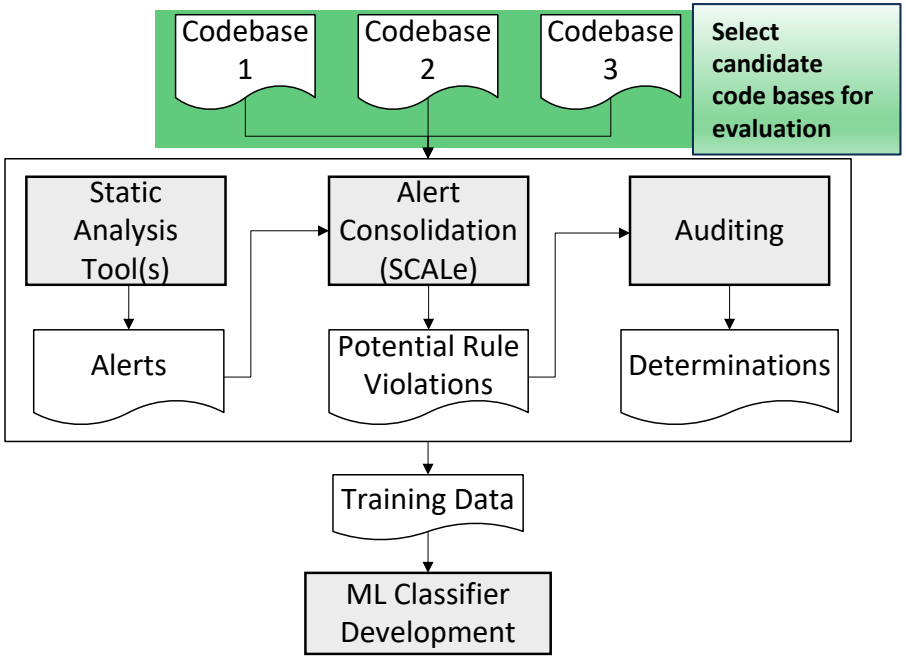
Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

Data Quality: Audit Lexicon and Rules

Meta-Alert Classifiers and Data Quality



Improve classifier precision & recall

Data quality

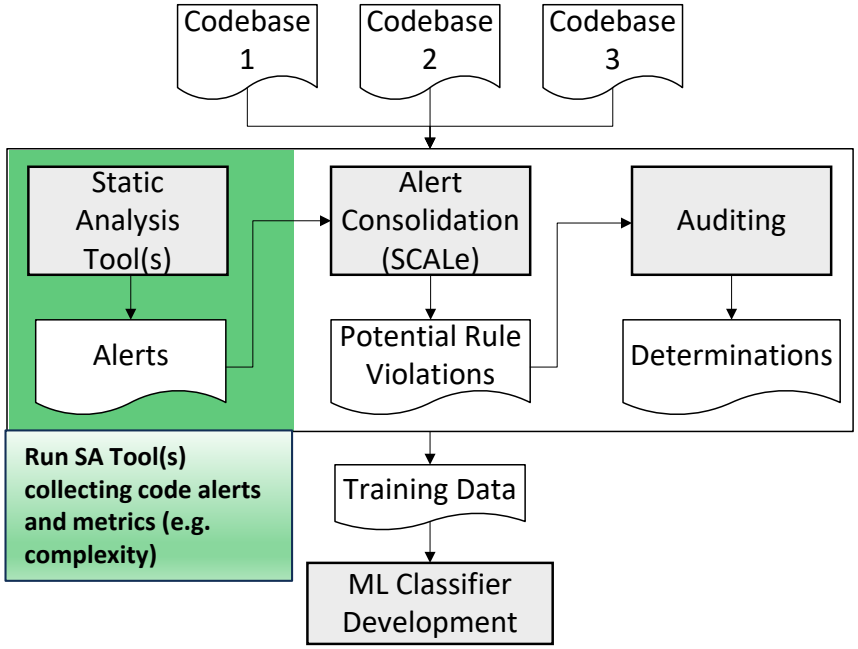
Wide variety of labeled data

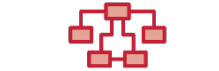
Enable classifier use via modular architecture

Enable classifier use in CI systems

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

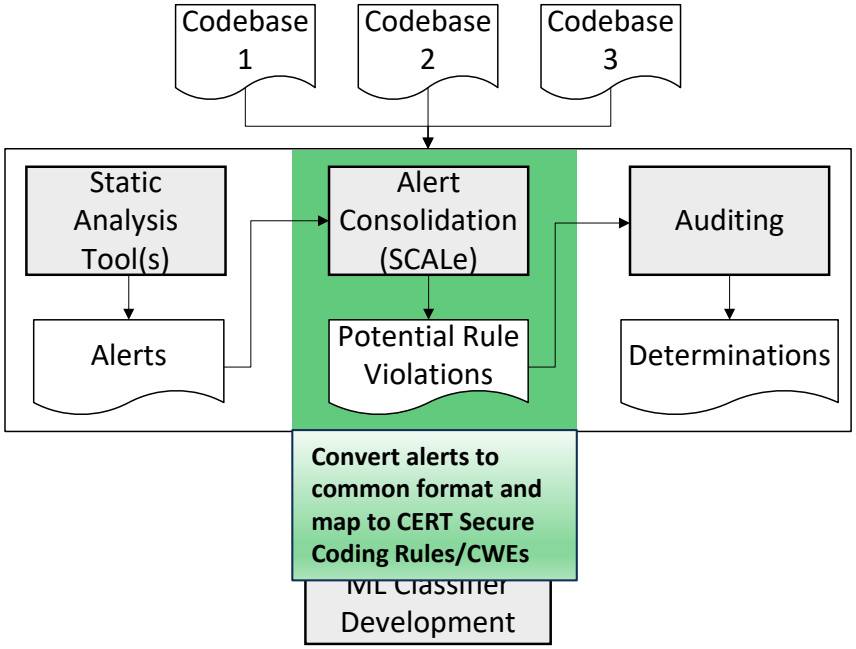
Meta-Alert Classifiers and Data Quality

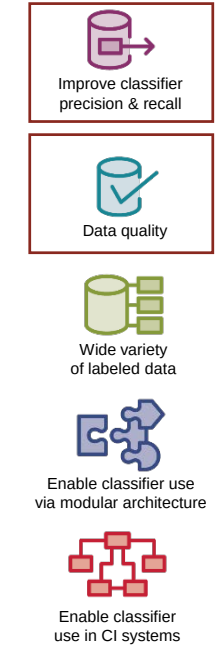




Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

Meta-Alert Classifiers and Data Quality

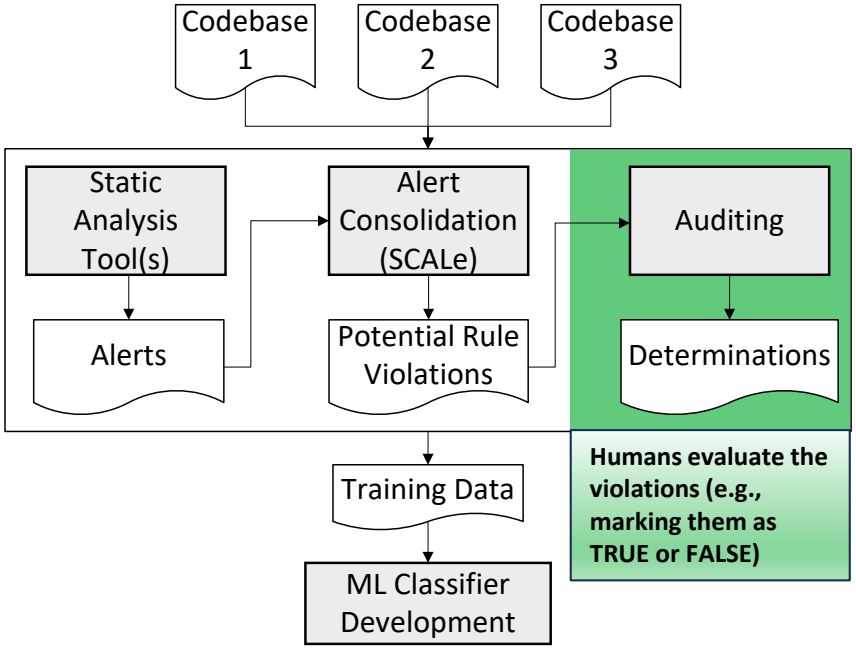


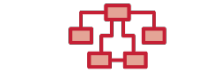


- MITRE Common Weakness Enumeration (CWE) https://cwe.mitre.org/index.html
- SEI Secure Coding Standards https://wiki.sei.cmu.edu/confluence/display/seccode

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

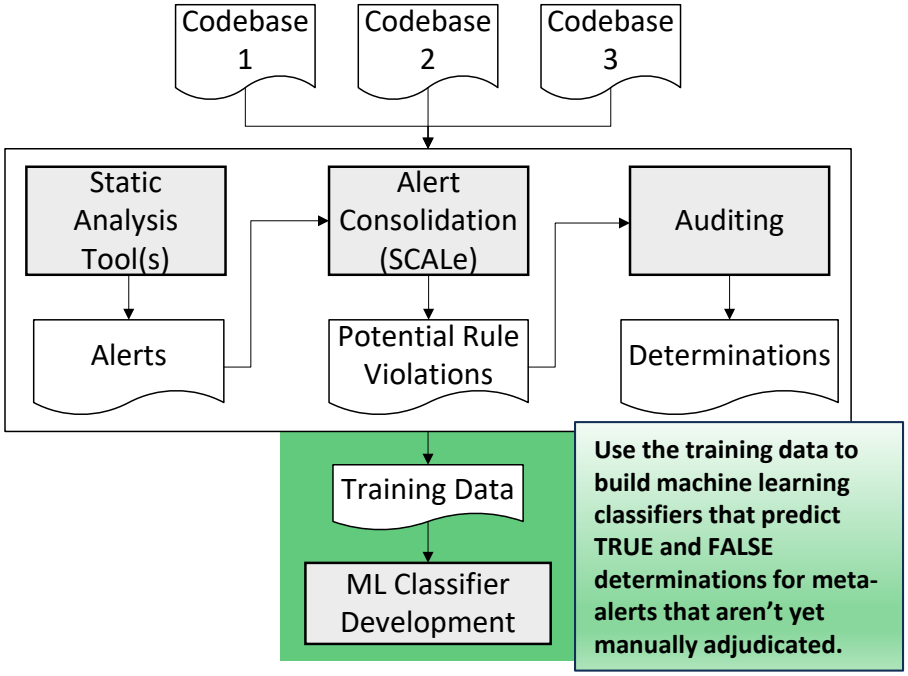
Meta-Alert Classifiers and Data Quality

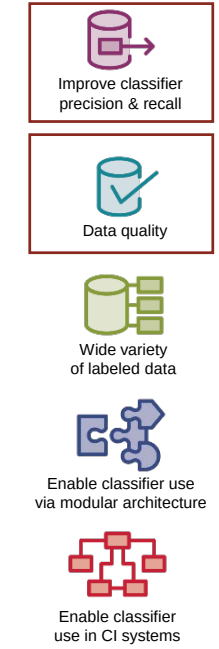




Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

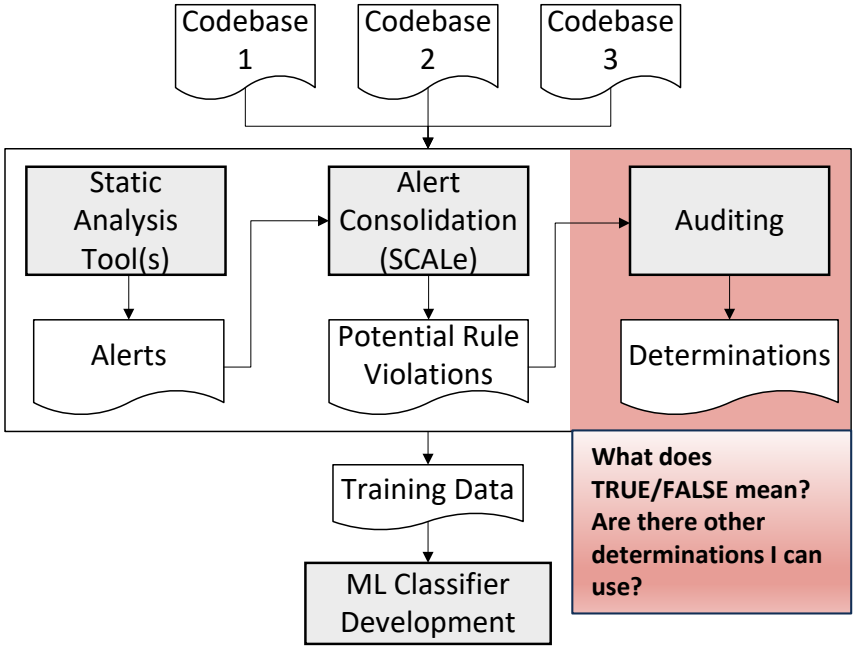
Meta-Alert Classifiers and Data Quality

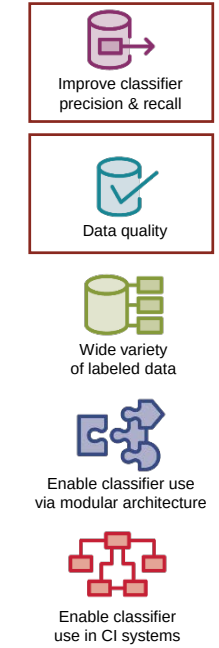




Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

Meta-Alert Classifiers and Data Quality



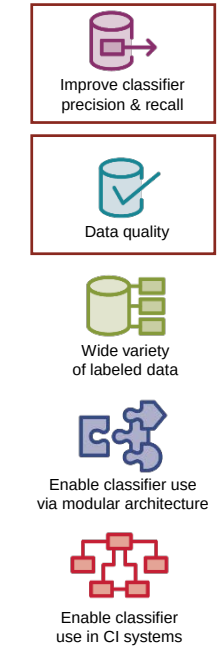


Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

Data Quality: What Is Truth?

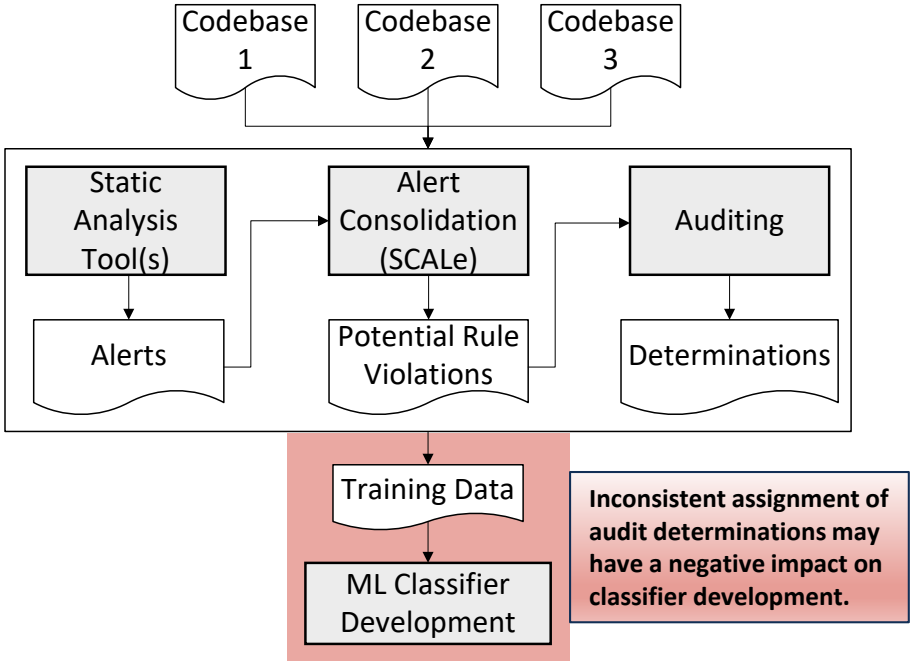
One collaborator reported using the determination true to indicate that the issue reported by the meta-alert was a real problem in the code.

Another collaborator used the determination true to indicate that something was wrong with the diagnosed code, even if the specific issue reported by the meta-alert was a false positive.



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

Meta-Alert Classifiers and Data Quality



Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

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

Data Quality: Lexicon and Rules

- We developed a **lexicon** and auditing **rule set** for our collaborators.
- It includes a standard set of well-defined **determinations** for static analysis meta-alerts.
- It also includes a set of **auditing rules** to help auditors make consistent decisions in commonly encountered situations and corner cases.

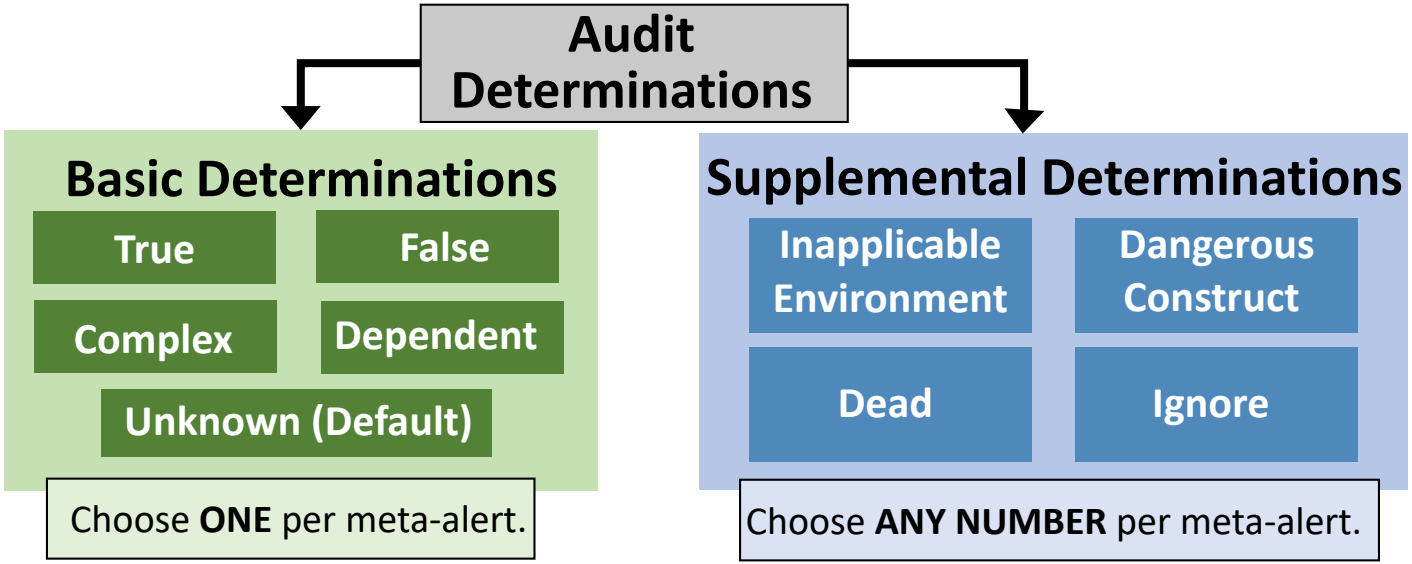
Different auditors should make the **same determination** for a given meta-alert.

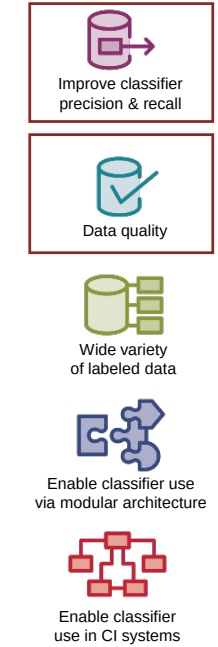
Improve the **quality and consistency** of audit data for the purpose of building **machine learning classifiers**.

Help organizations make **better-informed** decisions about **bug fixes**, **development**, and **future audits**.

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

Lexicon: Audit Determinations





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

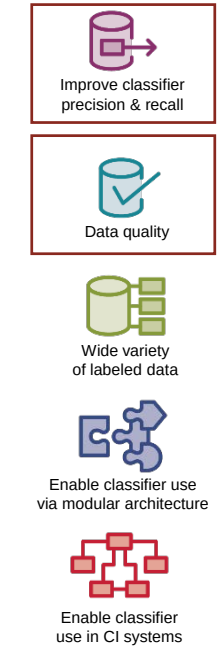
Data Quality: Audit Rules

Goals

- Clarify **ambiguous or complex** auditing scenarios.
- Establish **assumptions** auditors can make.
- Overall, help make audit determinations **more consistent**.

We developed **12 rules**:

- We drew on our experiences auditing code bases in the SEI CERT Division.
- We trained three groups of engineers on the rules, and we incorporated their feedback.



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

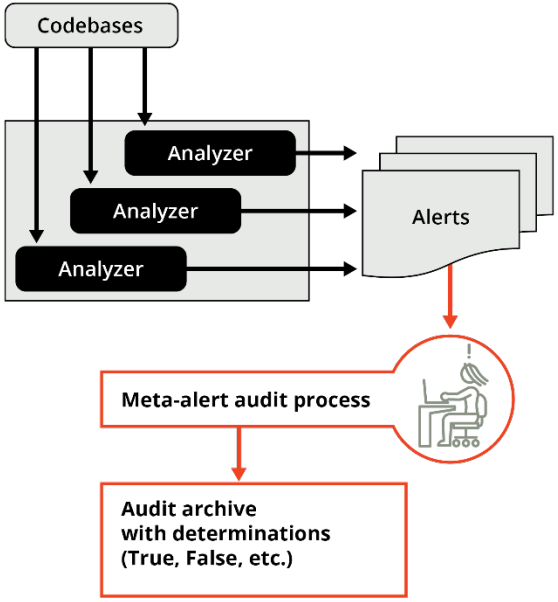
RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

Classifier Development, Data, & Enabling Architectures

Research Tooling Too

SEI SCALe Framework: Background



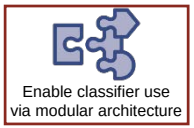
Static Analysis Meta-Alert Auditing Framework

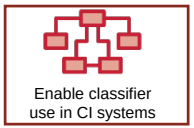
Developed by the SEI for ~10 years.

- GUI front end to examine meta-alerts and associated code
- Meta-alert adjudications (true, false) stored in database

Use for Research Projects

- We enhanced the framework with features for research.
- Collaborators use it on their codebases.
- Researchers analyze audit data.





After running SA tools, meta-alert adjudication can happen at any point in the software development lifecycle.

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

Archive Sanitizer for Collaborator Data Sharing

We added a data sanitizer to SCALe that has the following functions:

- Anonymizes sensitive fields
- Has an SHA-256 hash with salt
- Enables analysis of features correlated with meta-alert confidence

The audit archive for the project is in a database:

- Database fields may contain sensitive information.
- The sanitizing script anonymizes or discards fields:
  - Diagnostic message
  - Path, including directories and filename
  - Function name
  - Class name
  - Namespace/package
  - Project filename

Practical use of classification

Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

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

Data Used for Classifiers

Data used to create and validate classifiers:

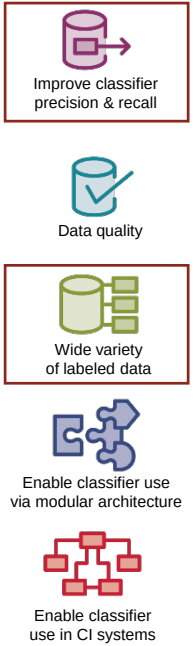
- CERT-audited meta-alerts:
  - ~7,500 audited meta-alerts
- Three collaborators that audit their own codebases with our auditing research prototype tool called “enhanced SCALe”

We pooled data (CERT and collaborators) and segmented it:

- Segment 1 (70% of data): train model
- Segment 2 (30% of data): testing

We added classifier variations on a dataset:

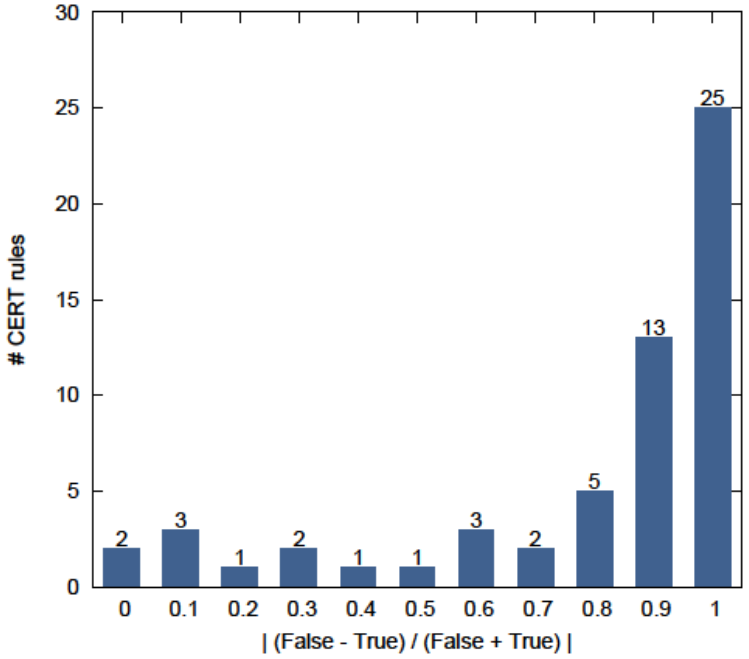
- Per-rule
- Per-language
- With/without tools
- Others



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

CERT-Audited Archives Characterization

- We had labeled data for 158 of 382 CERT rules.*
- There were 58 CERT coding rules with 20 or more audited (labeled) meta-alerts.
- The other 324 CERT rules have little or no labeled data.
- For 25 rules, all (or close) were determined one way (true or false).
- 2,487 were true, and 4,980 were false



Improve classifier precision & recall

Data quality

Wide variety of labeled data

Enable classifier use via modular architecture

Enable classifier use in CI systems

* SEI CERT Secure Coding Standards: https://wiki.sei.cmu.edu/confluence/display/seccode

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

Analysis of Juliet Test Suite: Initial 2018 Results

Automated Adjudication	Labeled Meta-Alert (counts a fused alertCondition once)
TRUE	**13,330**
FALSE	**24,523**

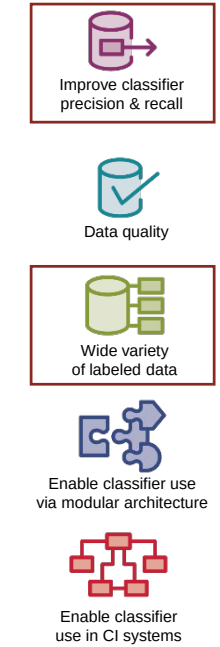
Lots of new data for creating classifiers

(37,853 labeled meta-alerts)

Big savings: A manual audit of *any* 37,853 meta-alerts from 'natural' programs estimated at 1,230 hours minimum (117 seconds per meta-alert audit*).

- It's unlikely that these meta-alerts would cover many conditions/flaws covered by the Juliet test suite.
- We needed true and false labels for classifiers.
- **Realistically**, a hugely larger manual auditing time is required to develop equivalent data; way more than 1,230 hours would be required.

These are initial metrics; we will collect more data as we use more tools and test suites.



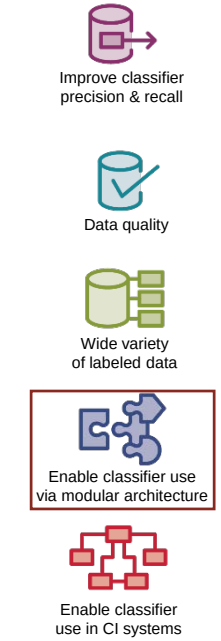
*N. Ayewah and W. Pugh. "The Google FindBugs Fixit", *International Symposium on Software Testing and Analysis,* ACM, 2010.

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

SCAIFE Definitions

SCAIFE is **a modular architecture that enables static analysis meta-alert classification** plus advanced prioritization.

- The **SCAIFE API** defines interfaces between the modular parts.
- **SCAIFE systems** are software systems that instantiate the API.
- Our SCAIFE system releases include a SCALe module plus much more.



SCAIFE = Source Code Analysis Integrated Framework Environment

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

SCAIFE Architecture Approach

To efficiently develop a robust API that enables widespread classifier use, we need a system architecture that does the following:

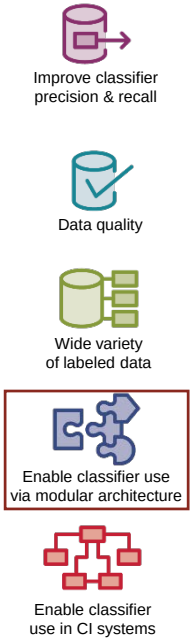
- Integrates with existing static analysis tools and aggregators (including SCALe)
- Supports classification and adaptive heuristic functionality
- Demonstrates fast response times for average and worst-case scenarios
- Provides extensibility for future research in static analysis, classification, architecture, and SecDevOps

Swagger/OpenAPI Open Source Development Toolset

- Quickly develops APIs following the OpenAPI standard
- Auto-generates code for servers and clients in many languages
- Tests server and client controllers with Swagger UI
- Is widely used (10,000 downloads/day)

- Big O analysis was useful.
- Design decisions required balancing goals and analyzing tradeoffs.

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



SCAIFE Architecture

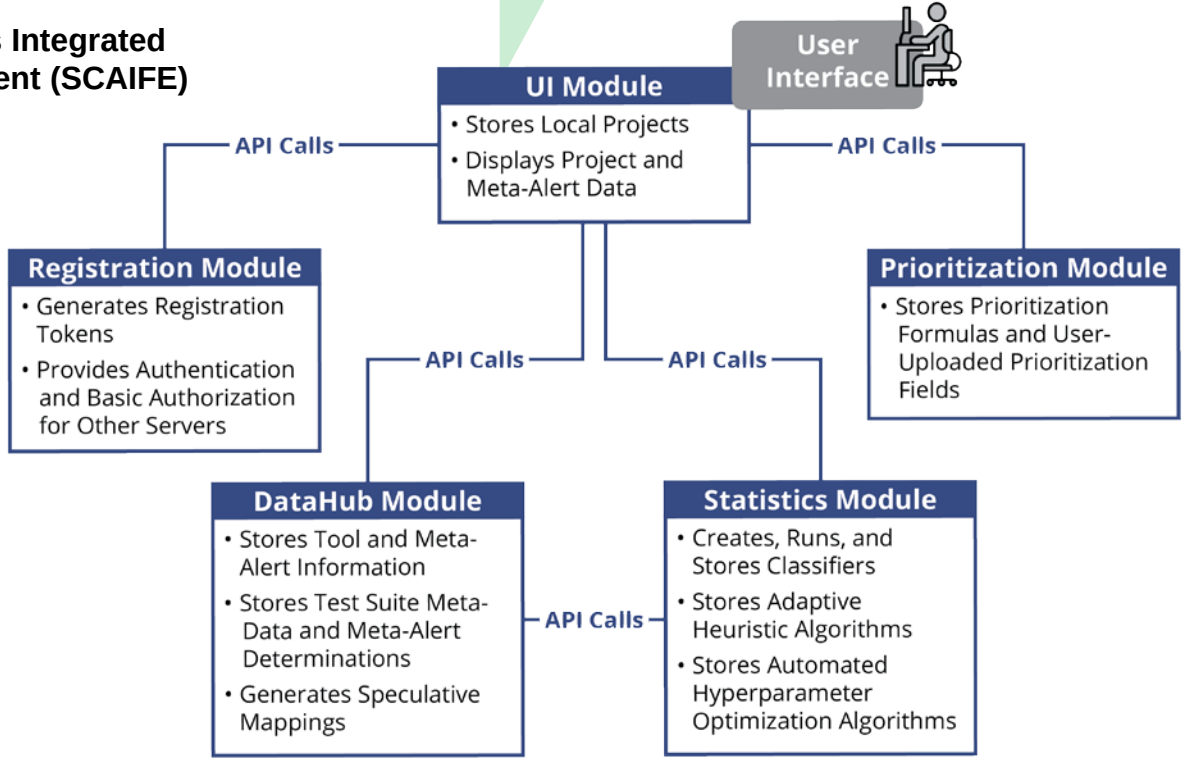
Any static analysis tool can instantiate APIs to become a UI Module. For example

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

Source Code Analysis Integrated Framework Environment (SCAIFE)

SCAIFE is a modular architecture that **enables users to efficiently start to use classifiers with a wide variety of systems and tools:**

- The formal SCAIFE API definition enables automated code generation to quickly instantiate API calls and generate server stubs in many code languages. This reduces the effort required to integrate existing systems and tools.
- The UI Module instantiation of SCALe is publicly available (GitHub scaife-scale branch).
- Collaborators can get a full SCAIFE instantiation and use it as-is or substitute any module(s) and use the others.

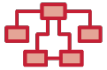



Improve classifier precision & recall


Data quality


Wide variety of labeled data

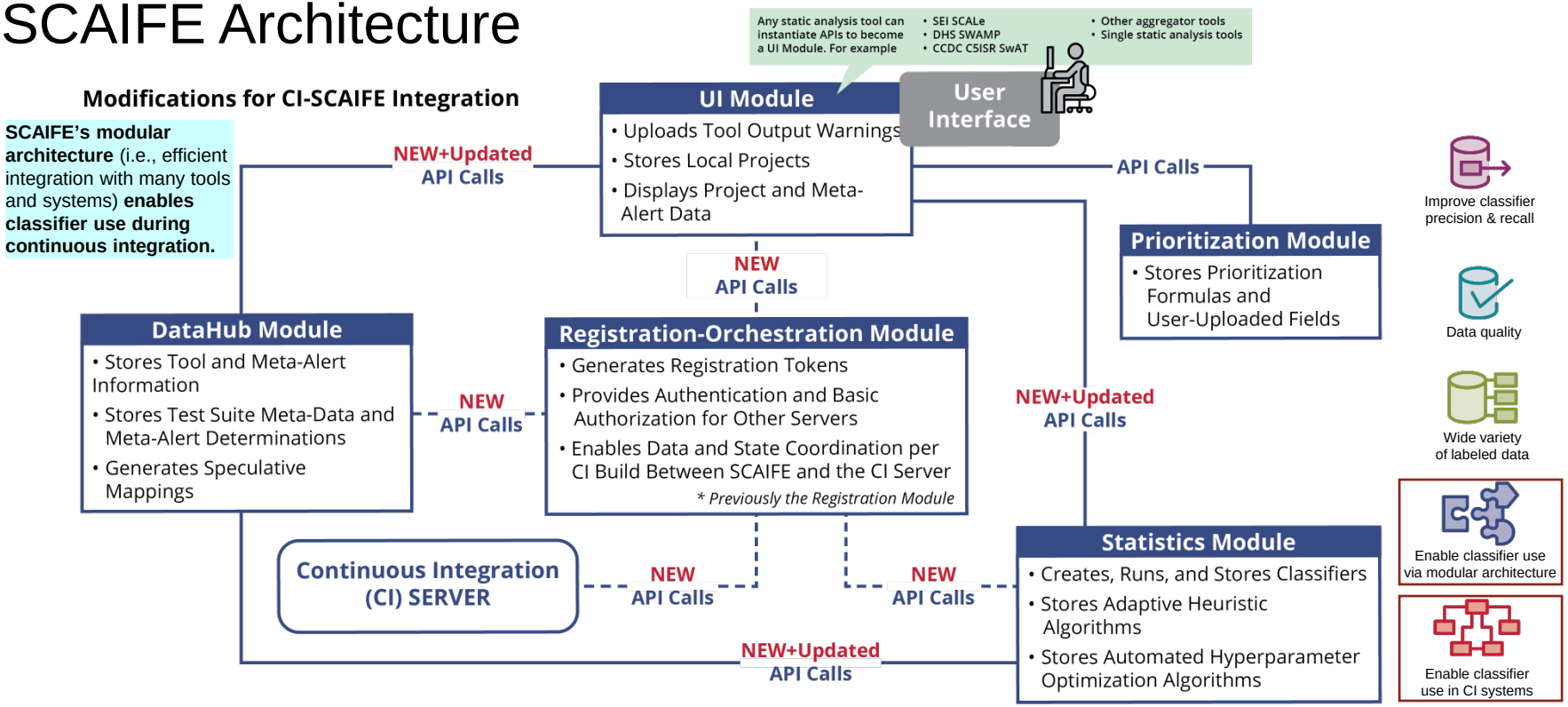

Enable classifier use via modular architecture


Enable classifier use in CI systems

L. Flynn, E. McNeil, and J. Yankel. "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code." SEI Technical Manual. July 2020.

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

SCAIFE Architecture



L. Flynn, E. McNeil, and J. Yankel. "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code." SEI Technical Manual. July 2020.

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

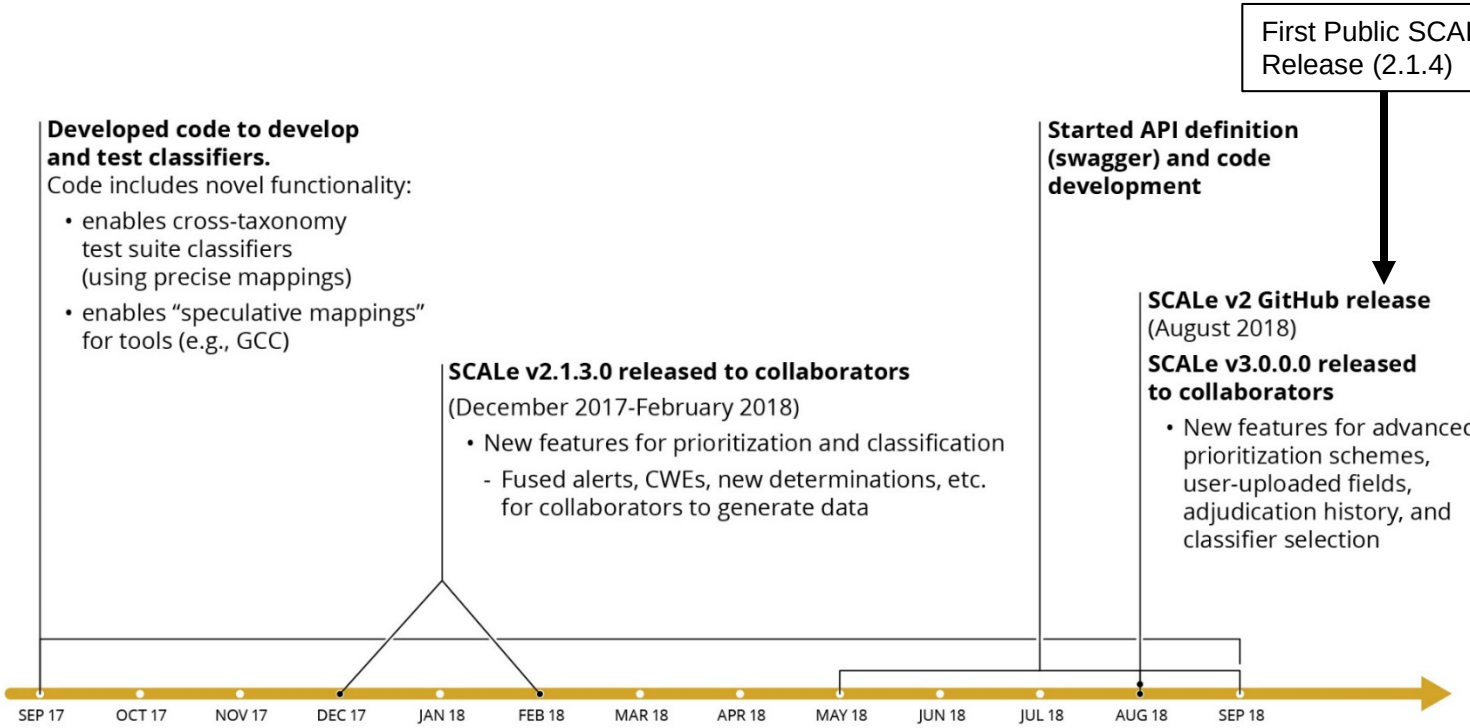
RESEARCH REVIEW 2020

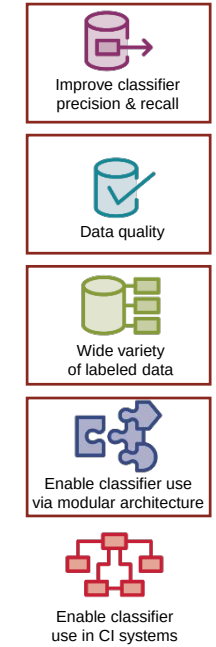
Static Analysis Classification:
Line-Funded Research FY16-20

FY18-19 Artifacts

FY18 Software Artifacts

- More recent versions available
- Notable: Multiple releases for collaborator feedback throughout



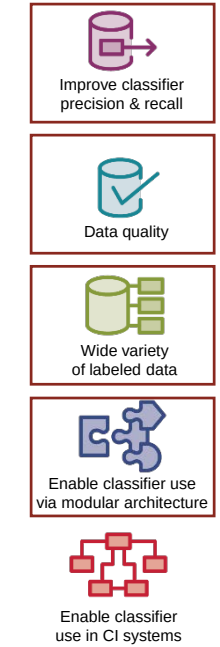


Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

FY18: Non-Code Publications

These resources remain useful

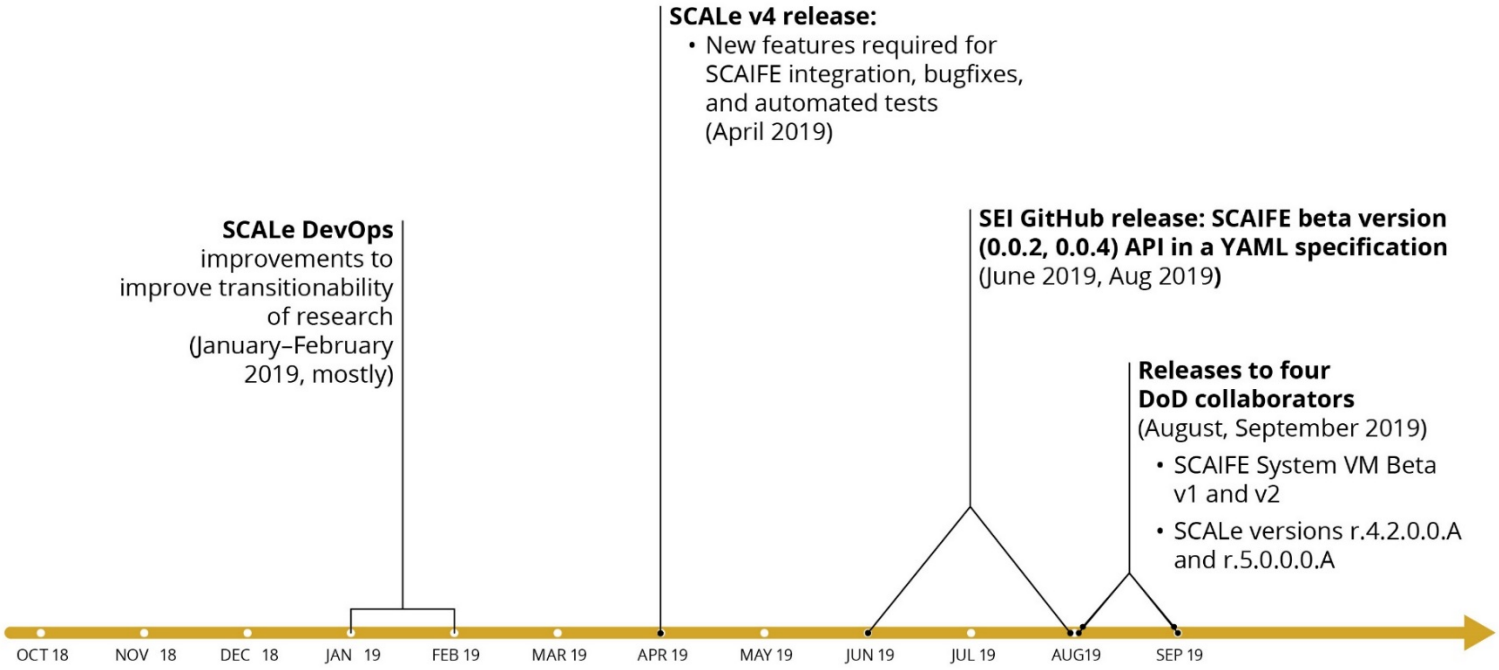
Publication Goal	Publications and Papers
Help developers and analysts provide feedback on our API, and use new SCALe features.	• SEI special report: *Integration of Automated Static Analysis Alert Classification and Prioritization with Auditing Tools* (August 2018) • SEI blog post: *SCALe: A Tool for Managing Output from Static Code Analyzers* (September 2018)
Explain classifier development research methods and results.	• Paper: *Prioritizing Alerts from Multiple Static Analysis Tools, Using Classification Models*, SQUADE (ICSE workshop) • SEI blog post: *Test Suites as a Source of Training Data for Static Analysis Alert Classifiers* (April 2018) • SEI podcast (video): *Static Analysis Alert Classification with Test Suites* (September 2018)
Enable developers and analysts to better understand tool coverage for code flaws using our inter-taxonomy precise mapping method.	• CERT manifest for Juliet (created to test CWEs) for testing CERT rule coverage with tens of thousands of tests (previously under 100) • Per-rule precise CWE mapping in two new CERT C Standard sections [1] [2]

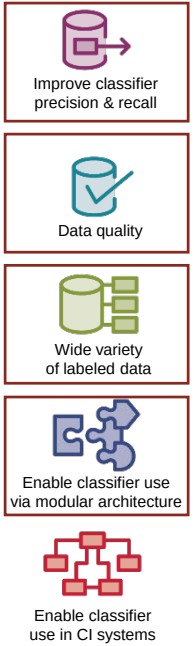


Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

FY19 Releases: Software and YAML API Definitions

- More recent versions available
- Notable: Multiple releases for collaborator feedback throughout





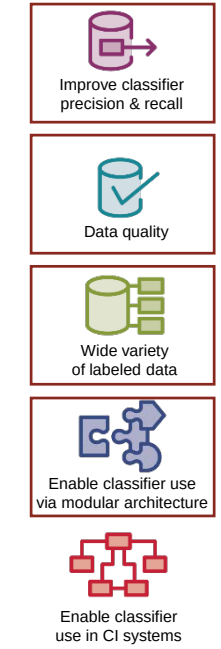
Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

FY19: Select Non-Code Publications –1

These resources remain useful

Publications to Explain Research and Development Methods and Results

- SEI blog post: *An Application Programming Interface for Classifying and Prioritizing Static Analysis Alerts* by Lori Flynn and Ebonie McNeil (July 2019)
- SEI whitepaper: *SCAIFE API Definition Beta Version 0.0.2 for Developers* by Lori Flynn and Ebonie McNeil (June 2019)
- SEI technical report: *Integration of Automated Static Analysis Alert Classification and Prioritization with Auditing Tools: Special Focus on SCALe* by Lori Flynn, Ebonie McNeil, David Svoboda, Derek Leung, Zach Kurtz, and Jiyeon Lee (May 2019)
- SEI blog post: *SCALe v3: Automated Classification and Advanced Prioritization of Static Analysis Alerts* by Lori Flynn and Ebonie McNeil (December 2018)
- Presentation: *Automating Static Analysis Alert Handling with Machine Learning: 2016-2018* (one-hour presentation at Raytheon's CyberSecurity Technical Interchange Meeting) by Lori Flynn (October 2018)



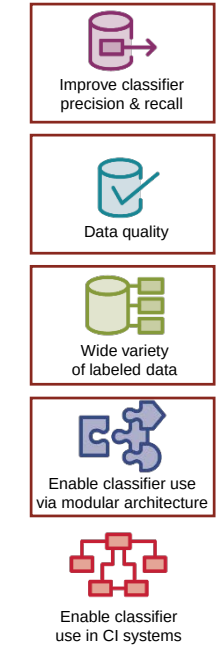
Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

FY19: Select Non-Code Publications –2

These resources remain useful

Publications to Demonstrate New Features of SCALe and SCAIFE

- Manual: *How to Review & Test the Beta SCAIFE VM* by L. Flynn and E. McNeil (v1 August 2019, v2 September 2019)
- SEI Cyber Minute by Ebonie McNeil (August 2019)
- SEI webinar: *How can I use new features in CERT's SCALe tool to improve how my team audits static analysis alerts?* (video and slides) by Lori Flynn (November 2018)
- Presentation: *Introduction to Source Code Analysis Laboratory (SCALe)* (one-hour presentation, including demo at Software Assurance Conference [SwACon]) by Lori Flynn (November 2018)



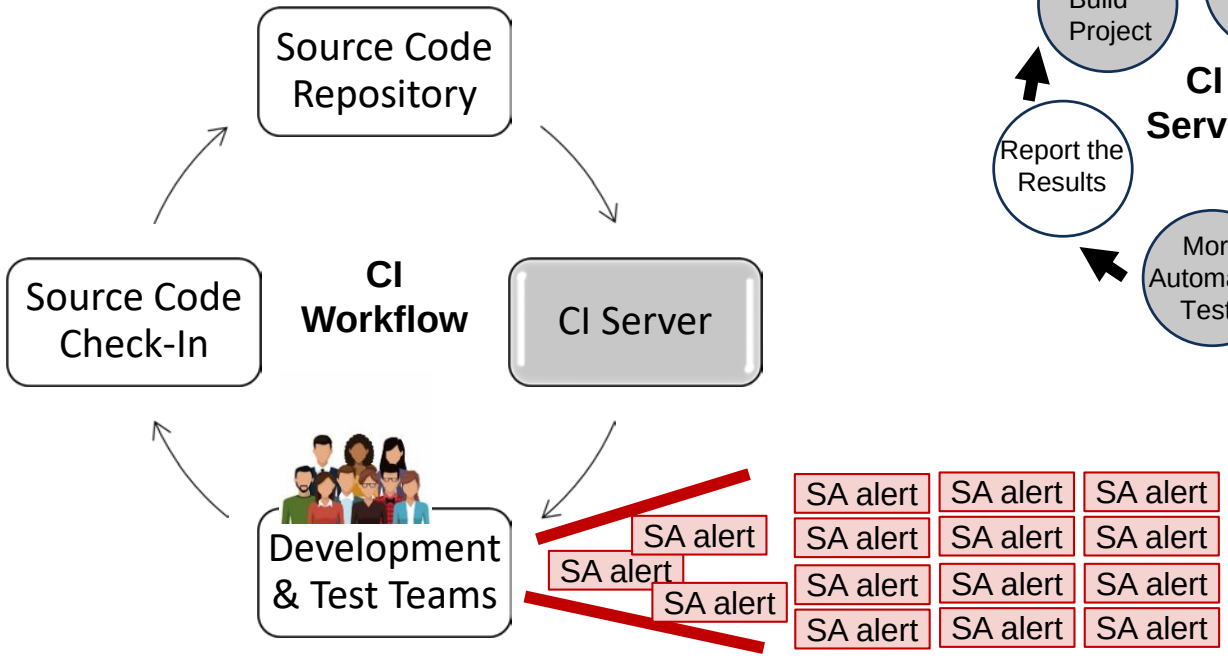
Goal: Enable **practical** automated classification so all meta-alerts can be addressed.

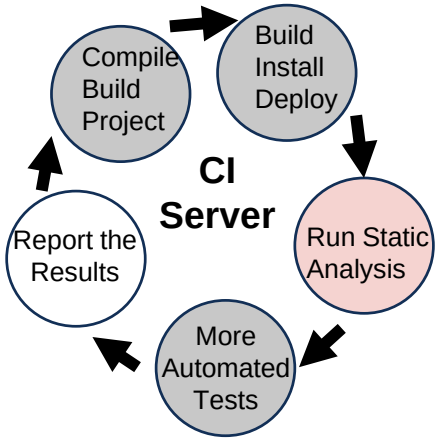
RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

FY20 Research Topic Detail and Artifacts

Rapid Adjudication of Static Analysis Meta-Alerts During CI






Improve classifier precision & recall


Data quality

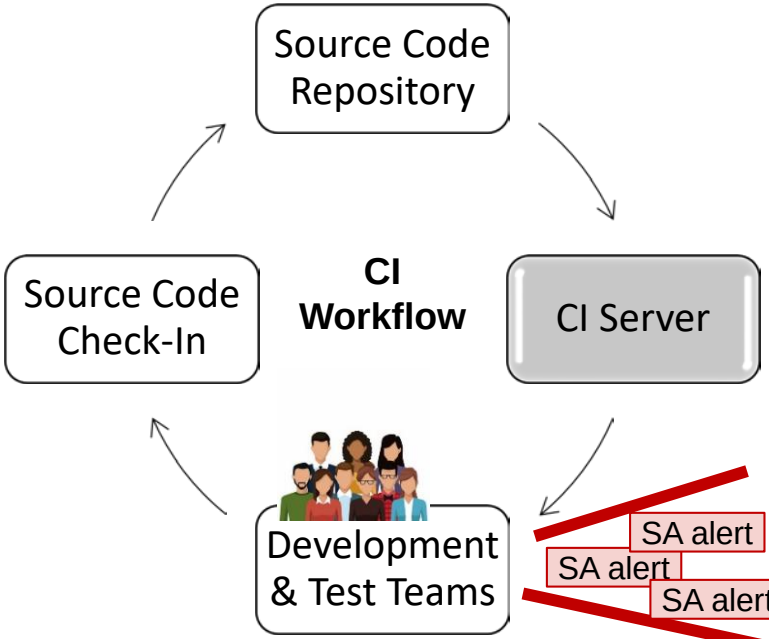

Wide variety of labeled data

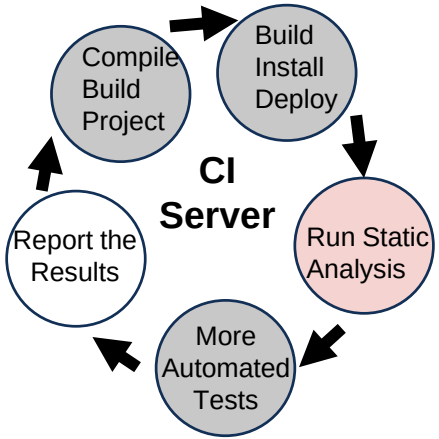

Enable classifier use via modular architecture

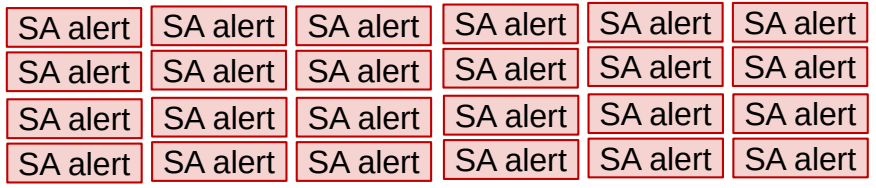

Enable classifier use in CI systems

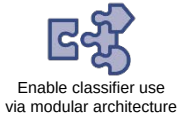
Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

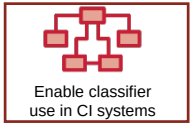
Rapid Adjudication of Static Analysis Meta-Alerts During CI





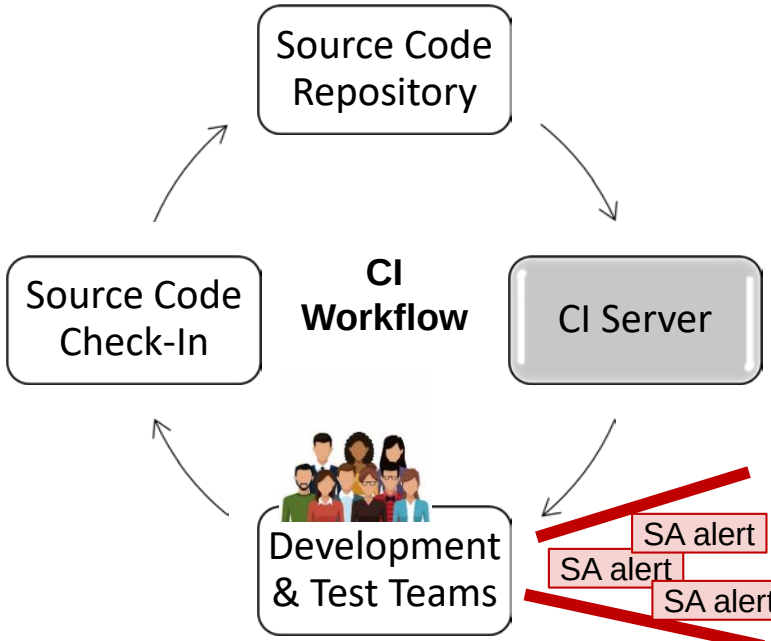


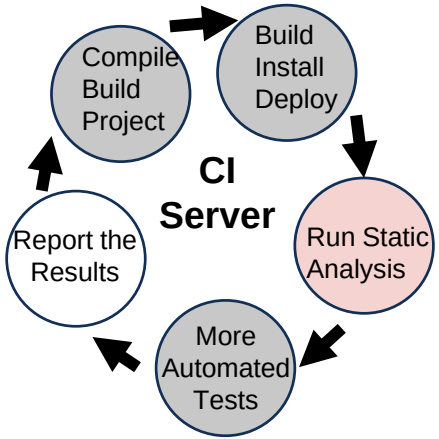




Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

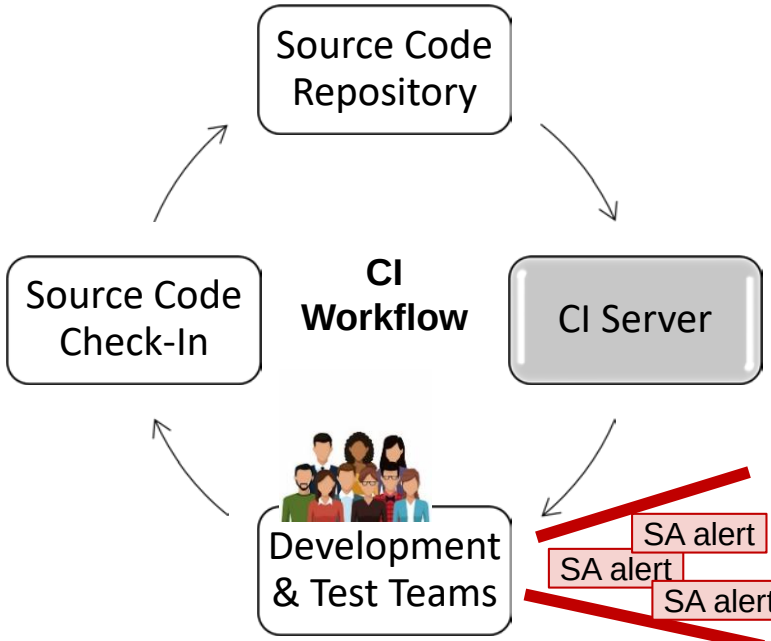
Rapid Adjudication of Static Analysis Meta-Alerts During CI

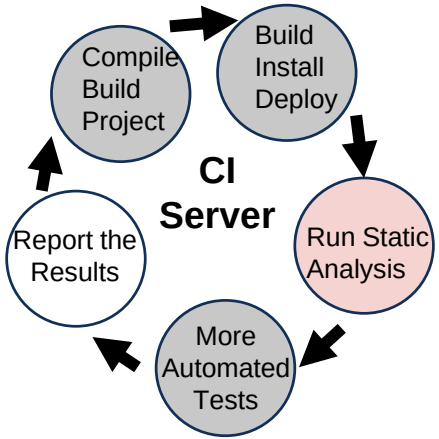




Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

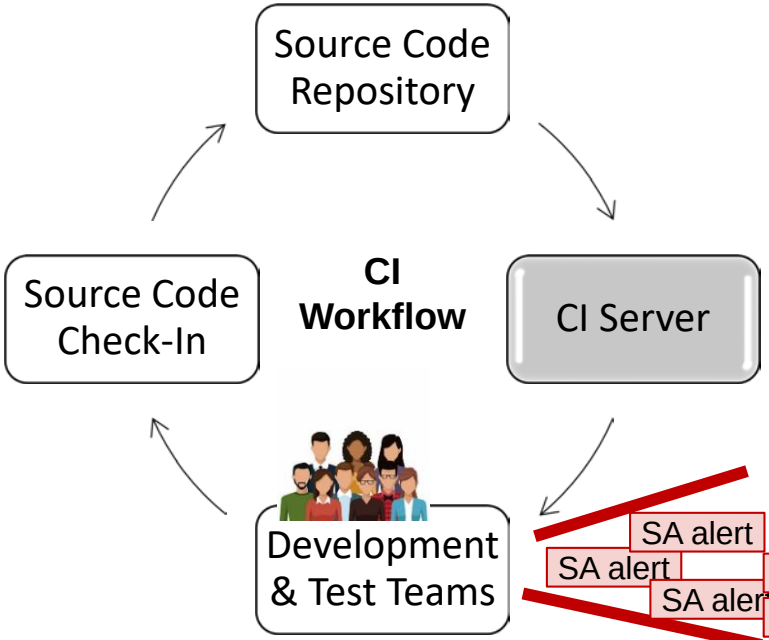
Rapid Adjudication of Static Analysis Meta-Alerts During CI

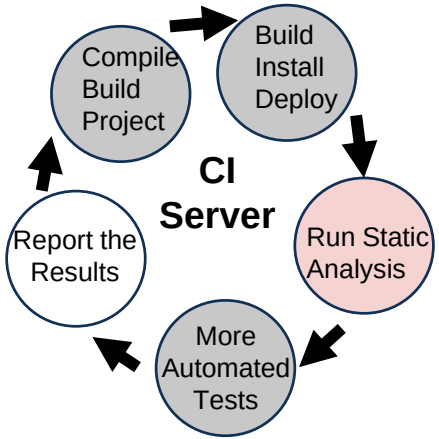


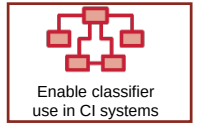


Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

Rapid Adjudication of Static Analysis Meta-Alerts During CI







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

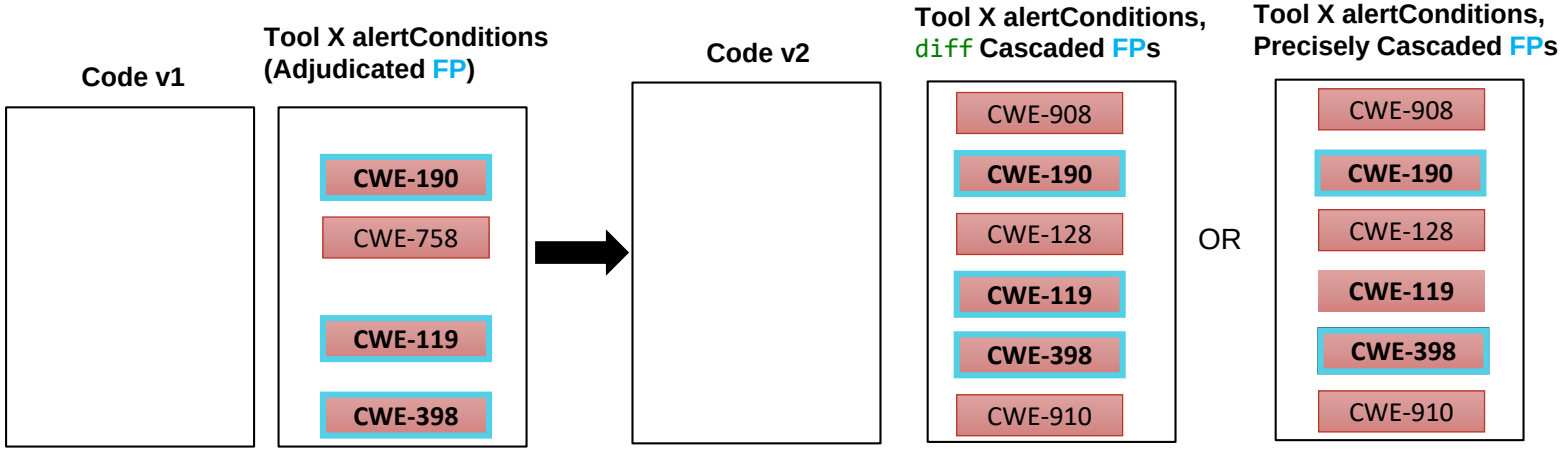
Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

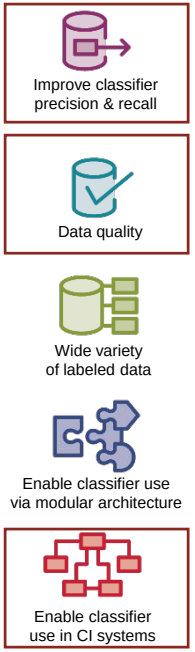
Precisely Cascading Adjudications from Previous Code Versions

1. Develop a new static analysis method that matches flaws in different versions of Java or C++ code.
2. Test programs for correctness. Test on open source codebases, then compare to `diff`, cascading to measure improvement.

Challenges

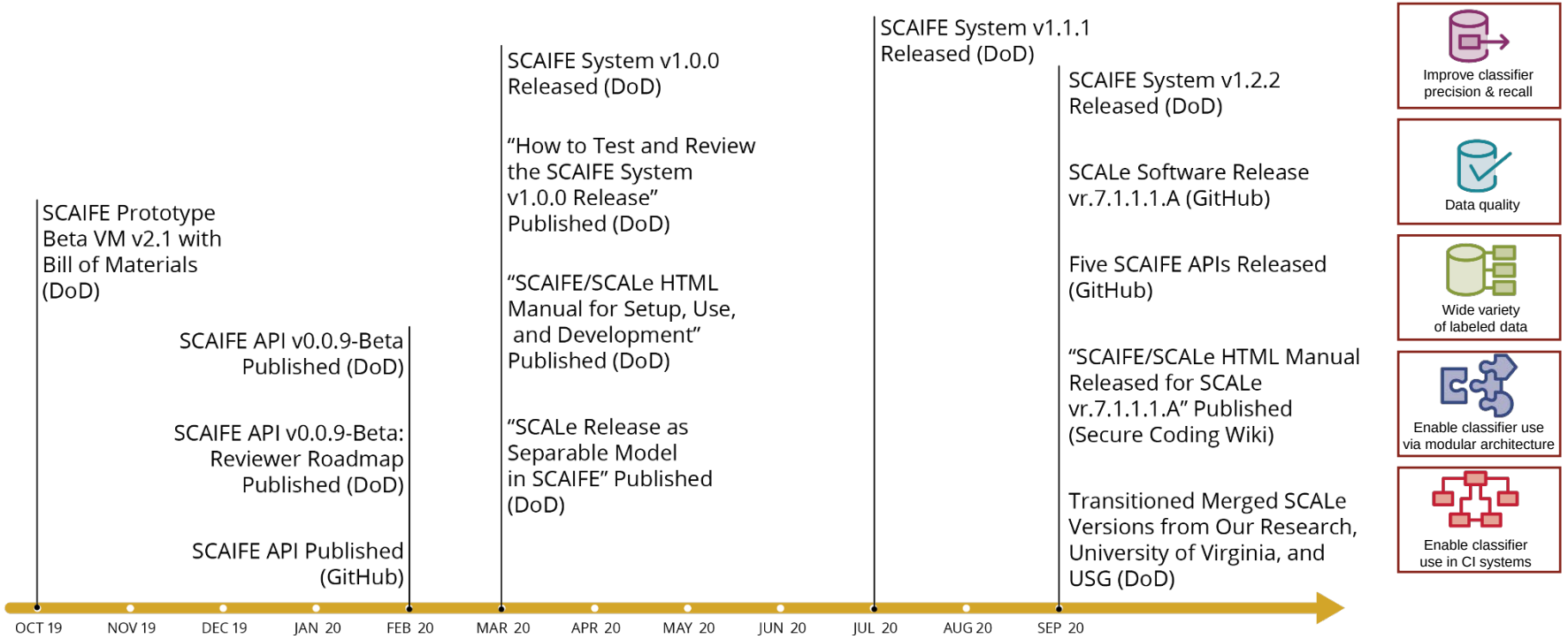
- Templates in C++
- Polymorphism and exception handling must be handled for C++ and Java
- Algorithm must be fast to work in CI system





FY20: Select Code/API Artifacts

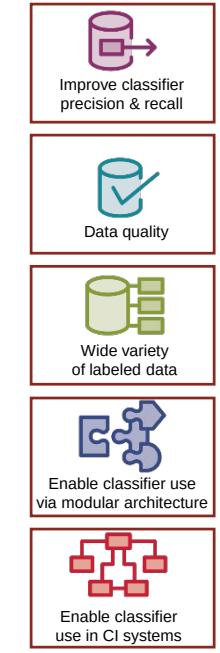
- DoD can get full implementation
- SCALe + SCAIFE API publicly-published (Sept 2020 versions)
- Significant CI integration; to be completed in FY21



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

FY20 Select Artifacts (New Detail or Item) –1

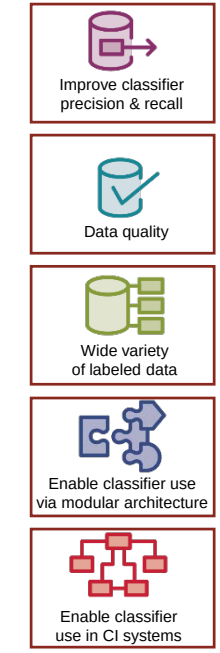
- (Oct 2019 and Feb, Apr, and Sep 2020) GitHub publication of SCAIFE API versions https://github.com/cmu-sei/SCAIFE-API
- (Apr 2020) Published the open dataset "RC_Data" for classifier research to the SEI CERT Secure Coding webpage "Open Dataset RC_Data for Classifier Research"; database with static analysis alerts from open-source tools, adjudications, code metrics, and more for two codebases
- (Jun 2020) Presentation "Automated Classifiers to Adjudicate Static Analysis Alerts: Challenges, Progress, and Next Steps" (Lori Flynn, Stephen Adams, and Tim Sherburne) to DoD's DEVCOM Cyber Community of Interest
- (Jun 2020) Presentation "Automated Classifiers to Adjudicate Static Analysis Alerts: Challenges, Progress, and Potential Collaborations with NASA IV&V" (L. Flynn) to leaders of the NASA IV&V Static Code Analysis Working Group (SCAWG)



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

FY20 Select Artifacts (New Detail or Item) –2

- (Jul 2020) Technical manual "How to Instantiate SCAIFE API Calls: Using SEI SCAIFE Code, the SCAIFE API, Swagger-Editor, and Developing Your Tool with Auto-Generated Code" (L. Flynn, E. McNeil, and J. Yankel) Instructions for three types of SCAIFE System code access: (1) none, (2) access to SCALe code, or (3) full access
- (Jul 2020) Auto-generated Java client code for the five SCAIFE API modules for a DoD collaborator, to help them quickly start to instantiate SCAIFE API calls from their tool
- (Sep 2020) Blog post "Managing Static Analysis Alerts with Efficient Instantiation of the SCAIFE API into Code and an Automatically Classifying System" by Lori Flynn
- (Sep 2020) Presentation "Using AI to Find Security Defects in Code / Build More Secure Software" at the Defense Science & Technology Agency (DSTA) Workshop
- (Sep 2020) Presentation "Rapid Adjudication of Static Analysis Meta-Alerts During Continuous Integration," Software Assurance Community of Practice (SwA CoP)
- (Sep 2020) SCALe code at https://github.com/cmu-sei/SCALe/tree/scaife-scale
- (Sep 2020) Test data generated for 'diff' cascading, for precise cascading comparison



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

RESEARCH REVIEW 2020

Static Analysis Classification Research FY16-20

Invitation to Collaborate

DoD Organizations That Do CI Development: Invitation to Test

I need DoD collaborators that do CI development to test our tooling:

- The current collaborators test but are not doing CI.
- The full system implementation release is currently limited to the DoD.
- CI testing does *not* have to include data sharing. (See the next slide.)
- If interested, please contact me at lflynn@cert.org.

Deployment and Testing Supported by Project

- release system containerized and with configuration files (ports, URLs, names) to ease integration in wide variety of systems
- comes with extensive documentation (We expanded the documentation significantly in the last year in response to collaborator feedback.)
- Part of the FY21 project is designed specifically to help collaborators use the system.

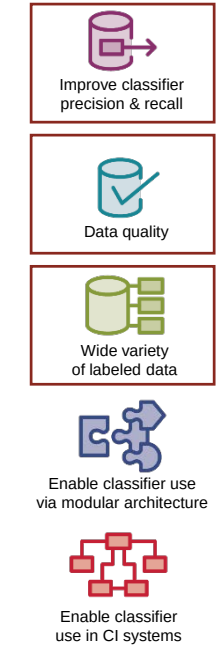
Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

Can You Help Us Get Labeled Data?

- We ask you to label data on particular open-source codebases.
- SCALe (scaife-scale branch) on GitHub can be used to do the adjudication and store the results.
- Even better, the SEI can provide the full SCAIFE system (including SCALe + classification, etc.) to DoD organizations.
- We provide auditing self-training support via published materials (next slide).
- You can use your own stored archives, providing that they are sanitized before you share them.

High-quality manually labeled data will help us improve our DoD sponsored classification research.

If our research succeeds, the improved classification techniques and data will help your organization (1) secure its code and (2) save money.



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

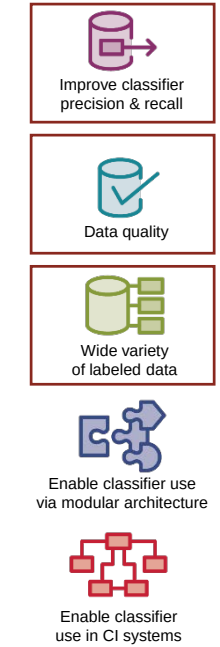
Self-Training Resources for Auditing Meta-Alerts

- Paper "Static Analysis Alert Audits: Lexicon & Rules" (D. Svoboda, L. Flynn, W. Snavely) IEEE SecDev
- Presentation "Hands-On Tutorial: Auditing Static Analysis Alerts Using a Lexicon and Rules" (L. Flynn, D. Svoboda, W. Snavely) https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=505451
- Webcast (1 hour video, hands-on SCALe use): "Improve Your Static Analysis Audits Using CERT SCALe's New Features" by L. Flynn. (The SCAIFE System includes the SCALe tool, as a separable part of SCAIFE.) https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=538843 (video) and https://resources.sei.cmu.edu/asset_files/Presentation/2018_017_101_532198.pdf (slides)
- Video "Rapid Construction of Accurate Automatic Alert Handling System" Nov. 2019 https://youtu.be/dwYbhgko3to
- Slides "Rapid Construction of Accurate Automatic Alert Handling System" Nov. 2019 https://resources.sei.cmu.edu/asset_files/Presentation/2019_017_001_635435.pdf

Increase the quality of your data by studying the definitions of the code flaw types ("conditions") that you will inspect static analysis meta-alerts for (as defined in a formal code flaw taxonomy).

Currently, for this classification research, the following taxonomies are of the most interest:
- MITRE CWE https://cwe.mitre.org/data/index.html
- CERT coding rules for C: https://wiki.sei.cmu.edu/confluence/display/c/SEI+CERT+C+Coding+Standard
- CERT coding rules for Java: https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java
- CERT coding rules for C++: https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

The SCALe (scaife-scale branch) GitHub release includes a SCAIFE/SCALe HTML manual with extensive information about how to use the SCAIFE and SCALe systems to adjudicate static analysis meta-alerts.



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

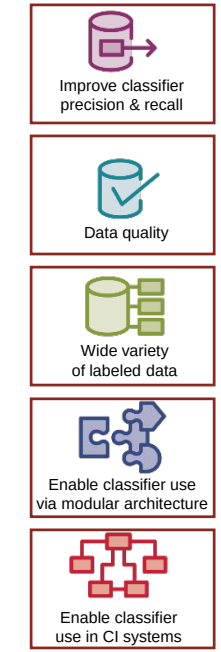
RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

Impacts

Project Impacts Time Frame

NEAR	MID	FAR
The public can use/review the SCAIFE API and SCALe* module. DoD collaborators will further test SCAIFE to • provide data and feedback • integrate their tools using the API The FY20-21 research project incorporates continuous integration (CI) into architecture design.	More collaborators (DoD and non-DoD) will test SCAIFE with CI. Design improvements for transition include • classification precision • latencies • bandwidth/disk/memory use • business continuity • scalability	A wide variety of systems will do automated meta-alert classification, using • SCAIFE System • SCAIFE API Goal: Provide better software security or require less time and cost for the same security (DoD and non-DoD).



** Version of SCALe used in SCAIFE System implementation*

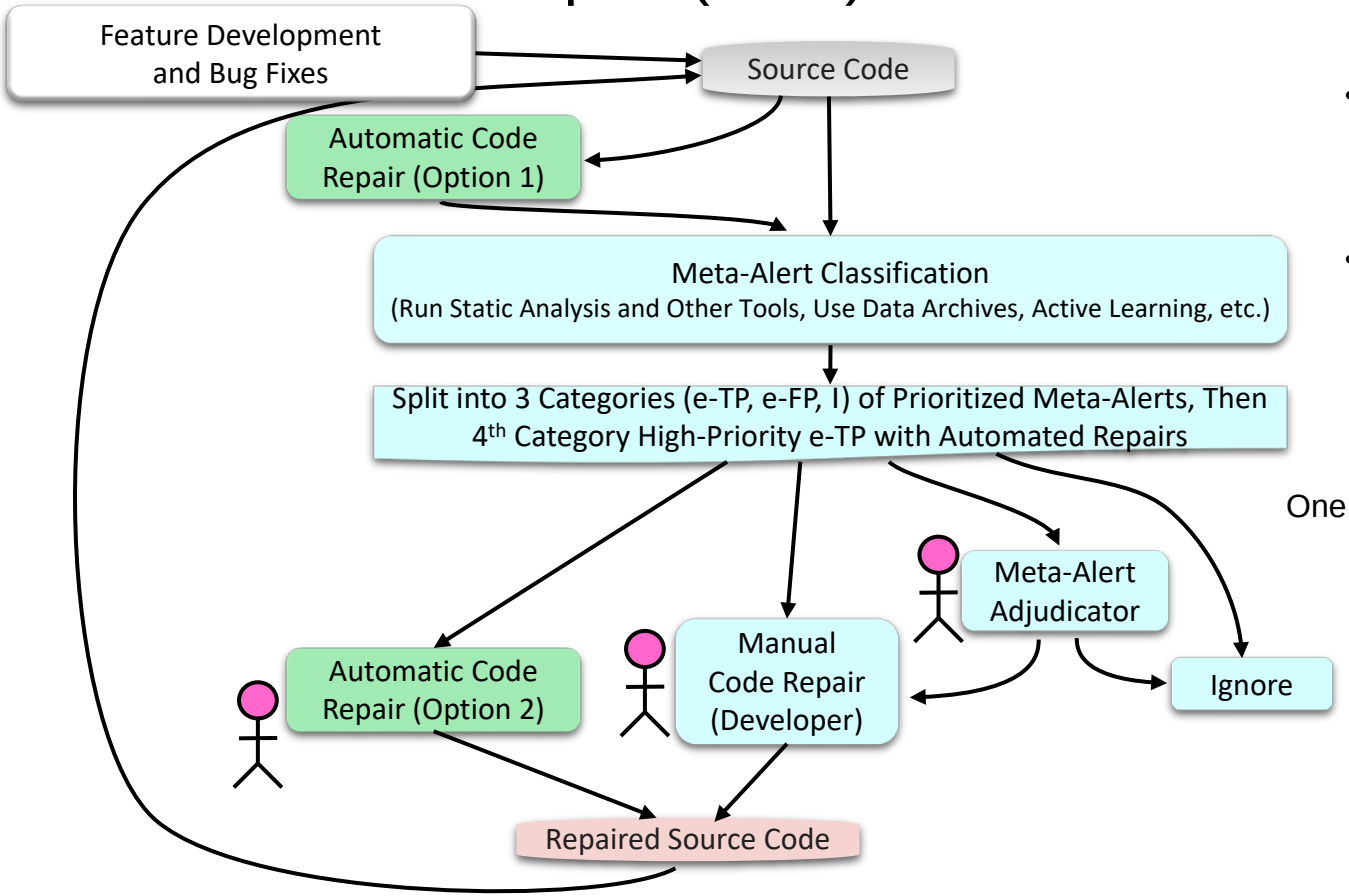
Goal: Enable **practical** automated classification for more secure software and lower cost/effort.

RESEARCH REVIEW 2020

Static Analysis Classification:
Line-Funded Research FY16-20

Combining SA Classification and Automated Code Repair

Automated Code Repair (ACR) & Meta-Alert Classification

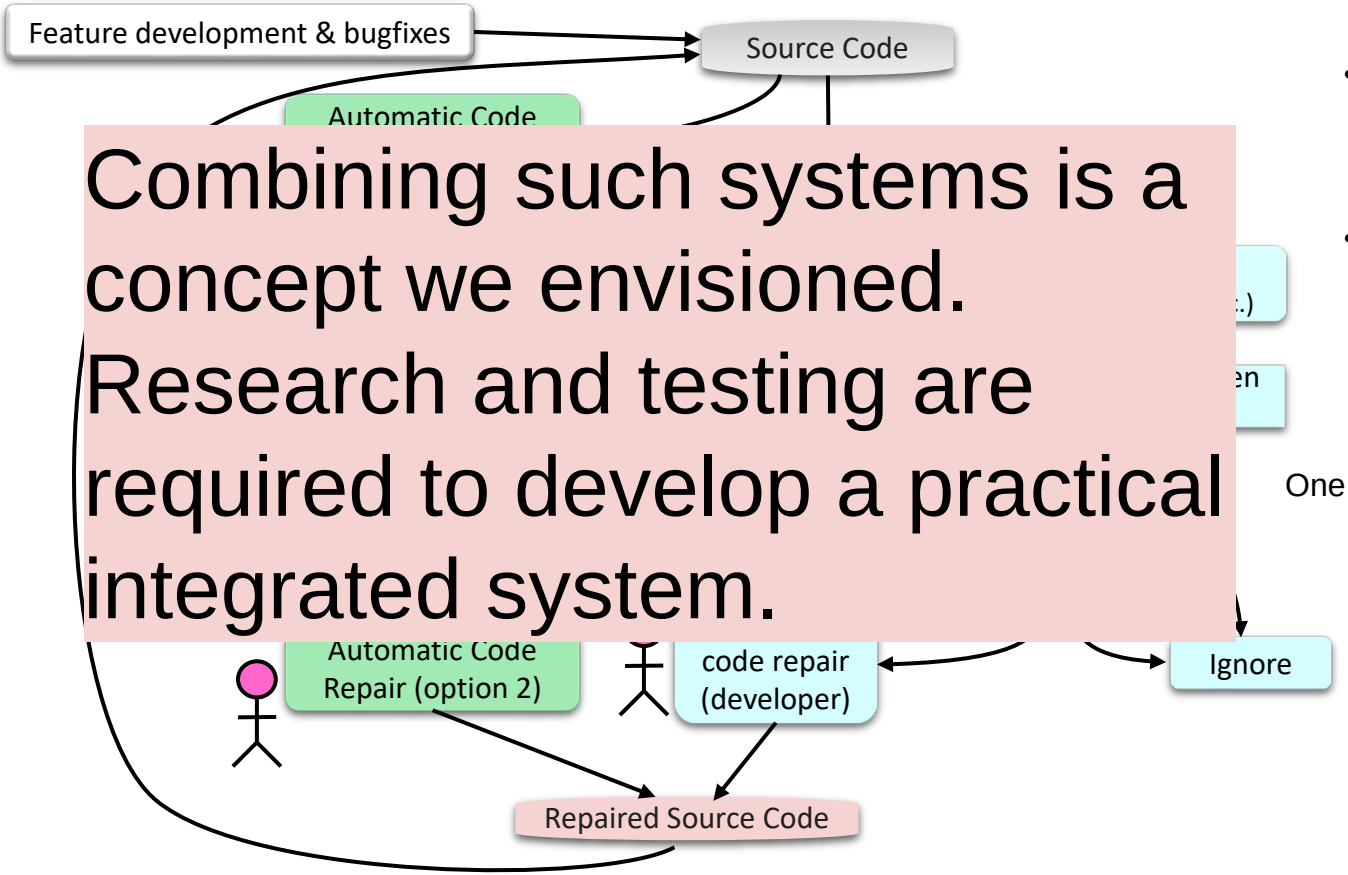


- ACR option 1: Make all possible automatic repairs (worse runtime overhead, better safety).
- ACR option 2: Repair only higher priority meta-alerts (less runtime overhead, but might leave unfixed vuls).

One vertical flow could happen:

- during one continuous integration build cycle (cycle until the build passes, then repeat each build)
- during a code security analysis+fix

Automated Code Repair (ACR), Semi-ACR, and Classification



Combining such systems is a concept we envisioned. Research and testing are required to develop a practical integrated system.

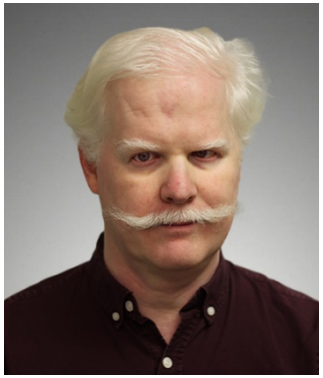
- ACR option 1: Make all possible automatic repairs (worse runtime overhead, better safety).
- ACR option 2: Repair only higher priority meta-alerts (less runtime overhead, but might leave unfixed vuls).

One vertical flow could happen:

- during one continuous integration build cycle (cycle until the build passes, then repeat each build)
- during a code security analysis+fix

FY20 Project Team





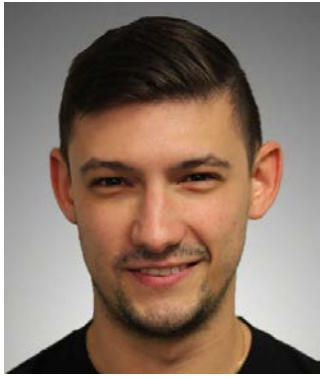

Dr. Lori Flynn

Ebonie McNeil

David Svoboda

Matt Sisk






Hasan Yasar

Joseph Yankel

Shane Ficorilli

David Shepard

For More Information

Contact Us

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

Phone: 412/268.5800 | 888.201.4479

Web: www.sei.cmu.edu

Email: info@sei.cmu.edu