Rapid Construction of Accurate Automatic Alert Handling System

Problem

Static analysis alerts for security-related code flaws require too much manual effort to triage efficiently. **Organizations are reluctant to fully adopt automated alert classifier technology because of barriers, including high cost, lack of expertise, and shortage of labeled data.**

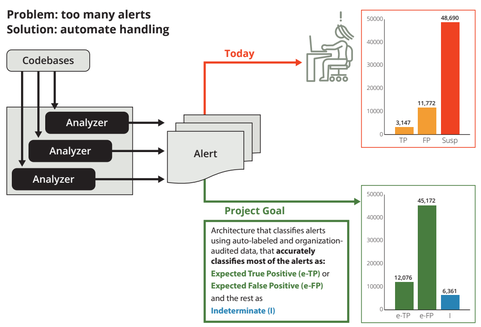
Solution

Develop an extensible architecture that supports classification and advanced prioritization, and builds on a novel test-suite-data method we developed.

- We developed a model and code intended to enable organizations to quickly start using classifiers and advanced prioritization by making API calls from their alert auditing tools.
- We implemented a prototype of the model.
- We developed adaptive heuristics for classifiers to adapt as they learn from test suite and natural program data.

Approach

- Design the architecture.
- Develop an API definition.
- Implement a prototype system.
- Develop adaptive heuristics and test them with datasets that combine test-suite and real-world (e.g., DoD) data.
- Test the architecture and prototype with collaborators.



To **overcome cost** and **data barriers**, we prototyped a **modular architecture that enables** the rapid adoption of **automated classifiers for static analysis alerts.**

SCAIFE Architecture

Source Code Analysis Integrated Framework Environment (SCAIFE)

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

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

User Interface

Registration Module
- Stores local projects
- Displays project and alert data

API Calls

Registration Module
- Generates registration tokens
- Provides authentication

Prioritization Module
- Stores and evaluates prioritization formulas

API Calls

API Calls

DataHub Module
- Stores tool and alert information
- Stores test suite metadata and alert determinations
- Generates speculative mappings

API Calls

Statistics Module
- Creates, runs, and stores classifiers
- Stores adaptive heuristic algorithms
- Stores automated hyperparameter optimization algorithms

FY19 Artifacts

Code and Test Results

- Beta SCAIFE prototype VM (v1, v2) released to collaborators (August & September 2019)
- API definitions (0.0.2-0.0.5) YAML publication (GitHub + SCAIFE VM)
- SCALe v3 and v4: tool released with new features for collaborators to generate data
- SCALe DevOps improvements for research transitionability
- SCALe v.r.4.*: released to collaborators with features for SCAIFE integration (August & September 2019)
- Code developed for prototype
- Adaptive heuristics

Publications

- SCAIFE API Definition and Prototype
  - Manual: How to Review & Test the Beta SCAIFE (v1, v2) VM (August & September 2019)
  - SEI blog post: An Application Programming Interface for Classifying and Prioritizing Static Analysis Alerts (July 2019)
  - SEI whitepaper: SCAIFE API Definition Beta Version 0.0.2 for Developers (June 2019)
  - SEI technical report: Integration of Automated Static Analysis Alert Classification and Prioritization with Auditing Tools (May 2019)
  - SEI blog post: SCALe v3: Automated Classification and Advanced Prioritization of Static Analysis Alerts (December 2018)
  - SwACon paper: Introduction to Source Code Analysis Laboratory (SCALe) (November 2018)
  - SEI webinar: How can I use new features in the CERT SCALe tool to improve how my team audits static analysis alerts? (November 2018)
- Classifier Development Research
  - Presentation: Automating Static Analysis Alert Handling with Machine Learning: 2016-2018 (October 2018)
  - Four in-progress papers addressing precise mapping, architecture for rapid alert classification, test suites for classifier training data, and API development

Project members developed (1) an architecture, (2) an API definition, and (3) a prototype system for static analysis alert classification and advanced alert prioritization.

Carnegie Mellon University
Software Engineering Institute

Dr. Lori Flynn | lflynn@sei.cmu.edu
Ebonie McNeil, Derek Leung, Matt Sisk, David Svoboda

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.

DM19-1027