

Automating Alert Handling Reduces Manual Effort

Static analysis tools search code for flaws without executing it – providing alerts about flaws that cyber intruders might exploit as vulnerabilities.

Today, those alerts require costly human effort to determine if they are true—about two minutes each—and to repair the code.

[show on screen or delete]

The number of alerts per lines of code varies according to the

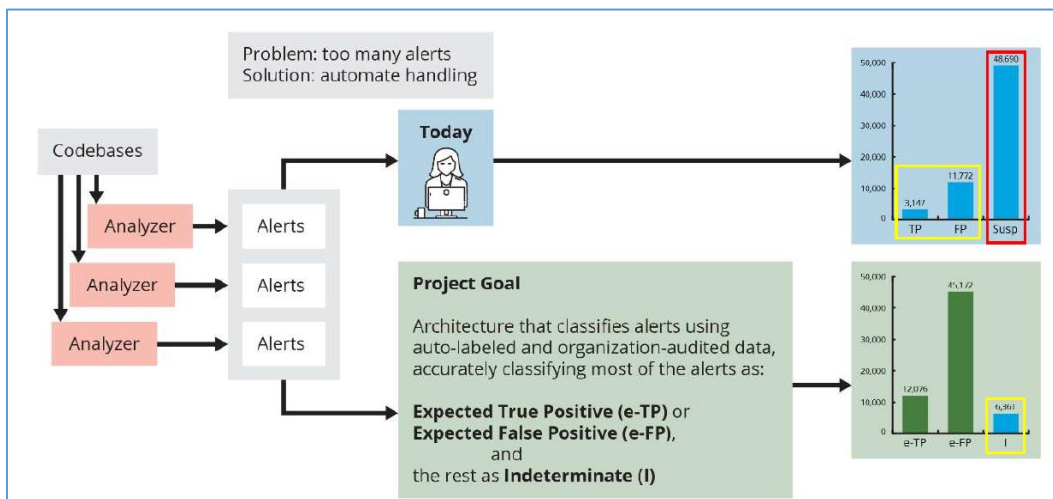
- code language,
- expertise of the coders, and
- quality of the static analysis tool

As a result, organizations often severely limit the types of alerts they **manually** examine to the types of code flaws they most worry about.

That approach results in a tradeoff where true flaws **never get fixed**.

To make alert handling more efficient, the SEI developed and tested novel software that enables the rapid deployment of a method to classify alerts—automatically and accurately.

[possible screen view under voice-over]



Those classified as *expected true* go directly to code repair; others recognized as *expected false* are ignored; and the remainder identified as *indeterminate* are prioritized for manual adjudication.

The goal: focus a development team's manual effort only on the flaws that are most likely to yield vulnerabilities.

We aim to significantly reduce the effort needed to inspect static analysis results and prioritize confirmed defects for repair.

We are implementing our solution in a new version of the SEI's SCALe – the Source Code Analysis Lab – application.

[B-roll: identify SCALe as Source Code Analysis Lab and show views from the SCALe interface that track with (1) finding flaws, (2) fusing flaws using the taxonomy, and (3) classifying alerts as true, false, or prioritized indeterminate]

Show on screen in conjunction with the B-roll

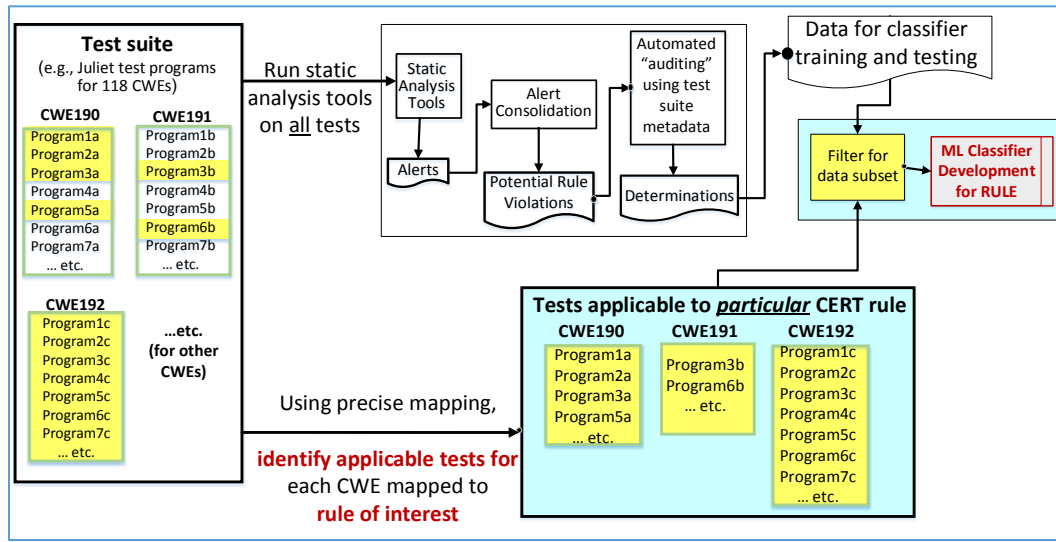
- *SCALe uses source code and output from static analysis tools that were run on the code as input.*
- *It provides the analyst with a browser-based interface to the alerts and their associated code.*
- *And it enables analysts to prioritize alerts with relevant information about*
 - *potential vulnerabilities and*
 - *how to fix the code based on the CERT Secure Coding Standards and common weakness enumerations or CWEs*

We started by generating more alerts **for more types of code flaws**, by using multiple static analysis tools on the same set of programs.

Why increase the number of alerts to adjudicate? Because we can mine the enlarged set of results for features to train our alert classifier – to make it more accurate.

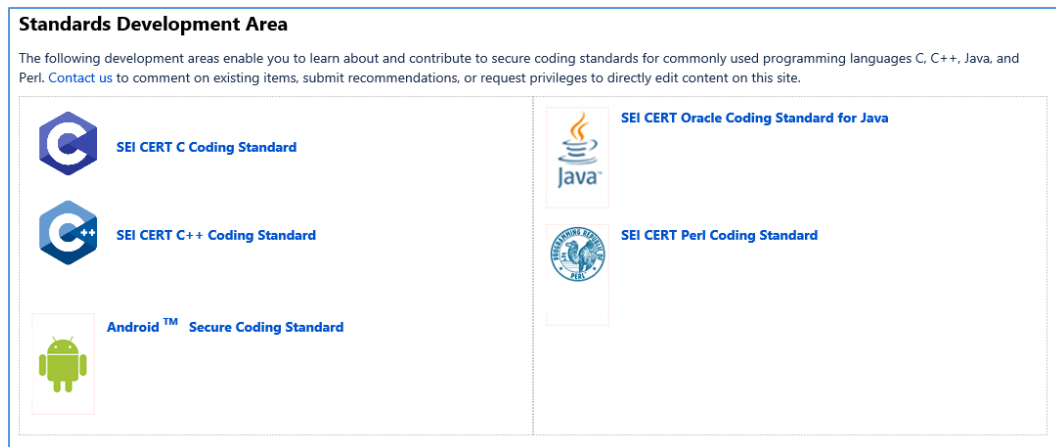
We developed our training data using the Juliet test suite, a collection of over 81,000 synthetic C/C++ and Java programs with known flaws. We used the pre-audited true and false results on those programs from eight static analysis tools.

[possible screen view under voice-over]



Then, we automatically fused alerts from the different tools for the same code flow, creating classifiers for both CWEs and for CERT secure coding rules.

[possible screen view under voice-over]



Our fusion script also counts alerts per file and per function. From this fusion step, we identify *features* for the classifiers, such as

- significant lines of code,
- cyclomatic complexity metrics,
- coupling metrics, and

Automating Alert Handling

- language

Next, we classified alerts using regression methods and machine learning algorithms.

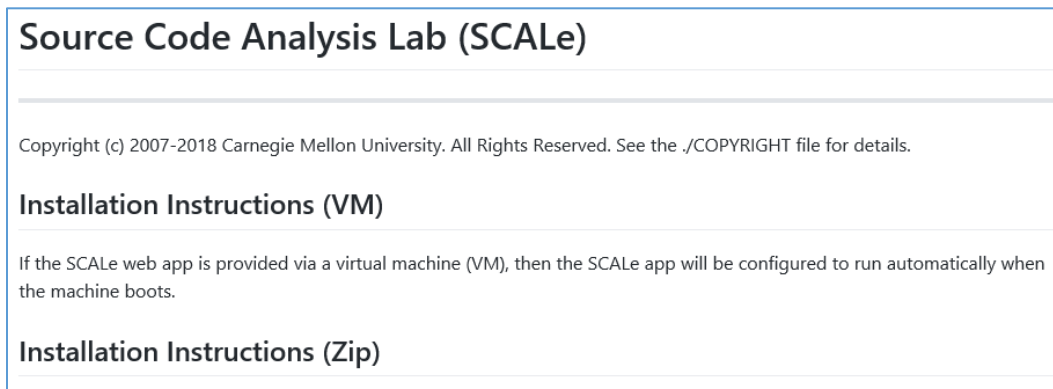
[show the classification techniques on screen]

- *Lasso logistic regression (LR),*
- *classification and regression trees (CART),*
- *random forest (RF), and*
- *eXtreme Gradient Boosting (XGBoost)*

Finally, we validated our classifiers using DoD-collaborator audits of their own codebases. We compared the predictions our classifiers made for the alerts to the adjudications made by the collaborators' auditors.

SCALE available for download from GitHub.

[possible screen view under voice-over]



Source Code Analysis Lab (SCALE)

Copyright (c) 2007-2018 Carnegie Mellon University. All Rights Reserved. See the ./COPYRIGHT file for details.

Installation Instructions (VM)

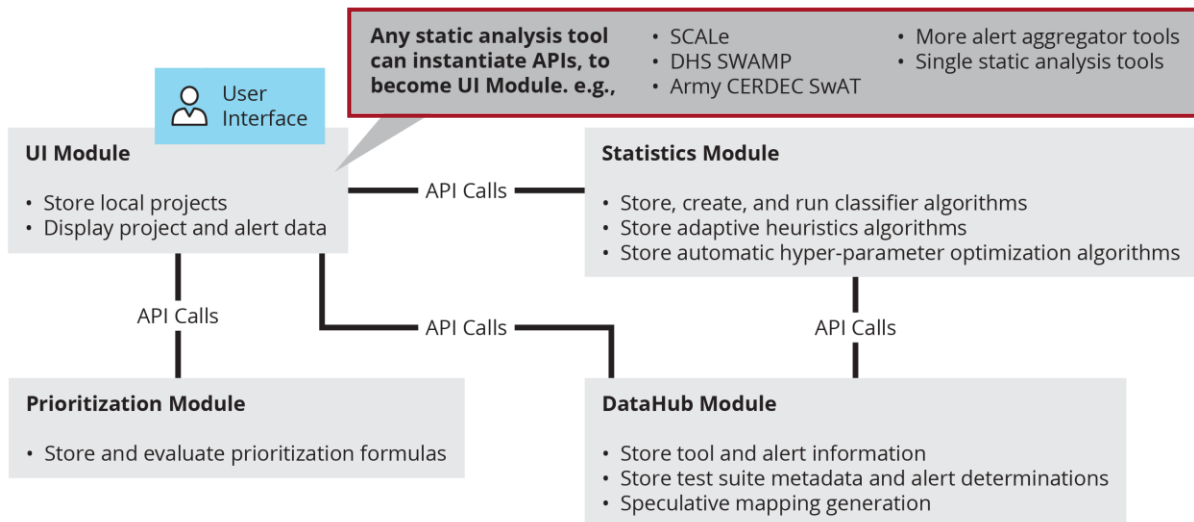
If the SCALE web app is provided via a virtual machine (VM), then the SCALE app will be configured to run automatically when the machine boots.

Installation Instructions (Zip)

Our research to rapidly deploy automated alert classifiers continues, including.

- algorithms to pre-seed classifier development with test suites
- an adaptive heuristic for precise alert classifiers
- automatic hyper-parameter optimization, and
- APIs for wide variety of applicable tools

[possible screen view under voice-over]



Our reference architecture and research focus areas

For more information, visit the below URL.

[on screen – url below points to information on SEI website but is unwieldly for use on screen; as an alternate, we could point to <https://github.com/cmu-sei/SCALe>, though pointing to our website is preferable]

https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=6453

[Word count and length estimate 415/130 = 3.2 min.]

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

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.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* 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 external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

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

DM19-0793