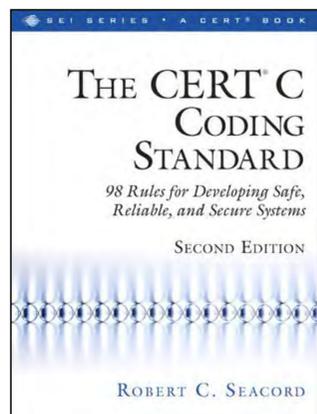


Increasing Adoption of Secure Coding

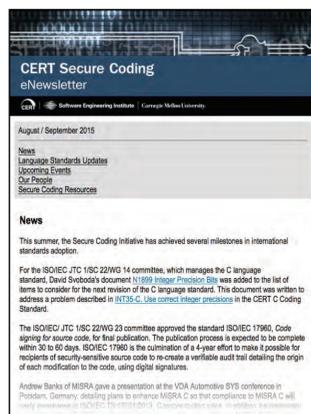
Coding standards are an integral part of the secure software development lifecycle and increasingly a requirement. Proper coding results in fewer weaknesses, fewer vulnerabilities, and reduced costs for cyber protection. This research provides the foundational prescriptive rules as well as practical support for putting the rules into practice.

The Definitive Reference for C Programs



The definitive reference for prescriptive guidance in writing C programs

CERT Bimonthly Secure Coding eNewsletter



Over 1,600 contributors provide input and updates to the coding rules wiki.

Providing the foundational rules for coding.

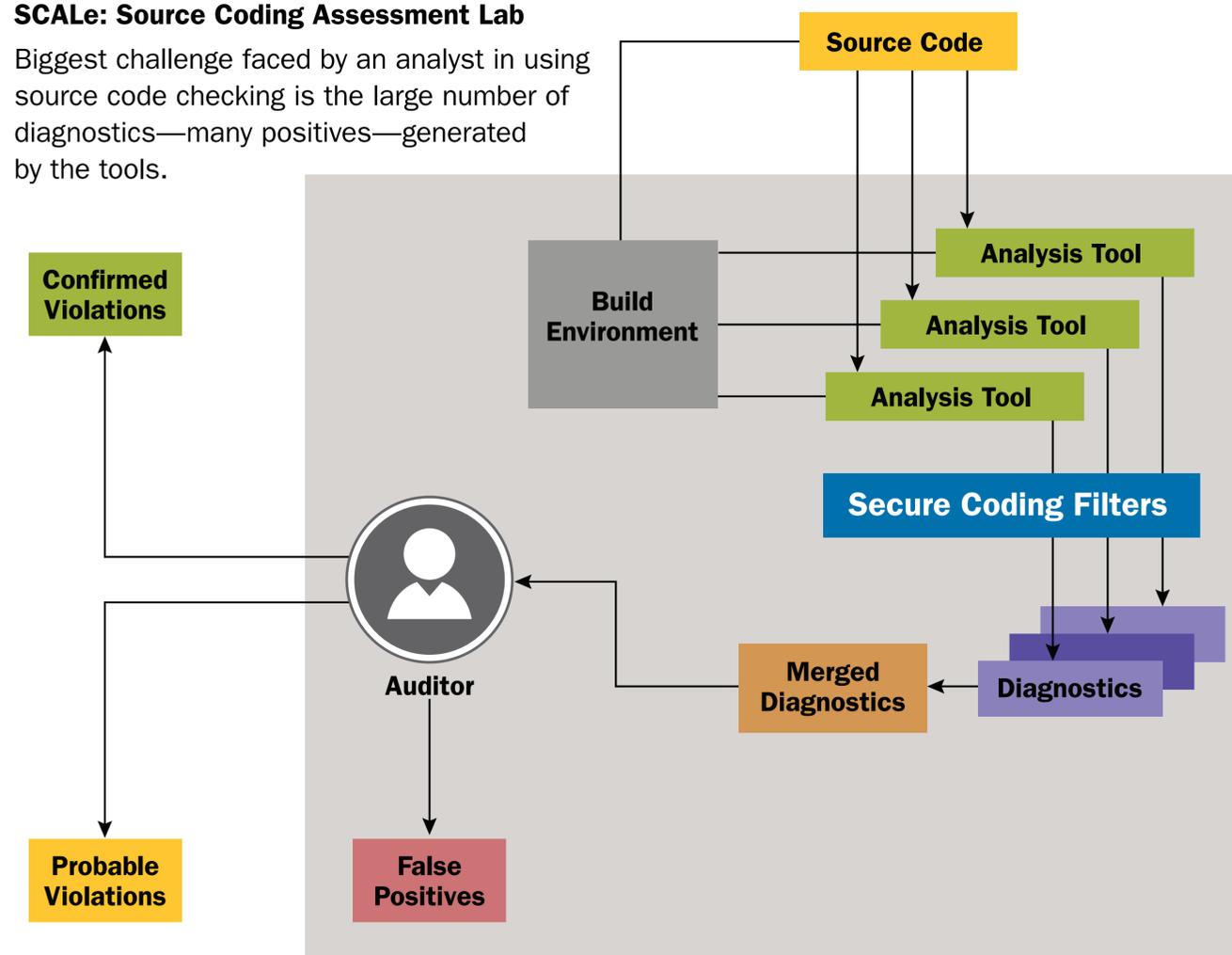
The project's research has provided foundational rules for programming in C and Java.

The current project extends the rules to encompass new programming models—threads—and the updated C/C++ language:

- 25 new rules in FY15 specifying C and C++ weaknesses
- 99 rules dedicated to C-specific weaknesses
- 74 rules dedicated to C++-specific weaknesses (148 total rules when including overlapping C rules)
- 9 new unspecified behaviors in C threads

SCALE: Source Coding Assessment Lab

Biggest challenge faced by an analyst in using source code checking is the large number of diagnostics—many positives—generated by the tools.



SCALE makes expert review more productive by focusing on high priority violations out of the volume of diagnostics provided by tools.

- Filter select secure coding rule violations
- Eliminate irrelevant diagnostics
- Convert to common CERT Secure Coding rule labeling
- Provide single view into code and all diagnostics

SCALE maintains a record of decisions so that results about a review are maintained as code is changed and not revisited.

Integrating checkers into Integrated Development Environments (IDE)

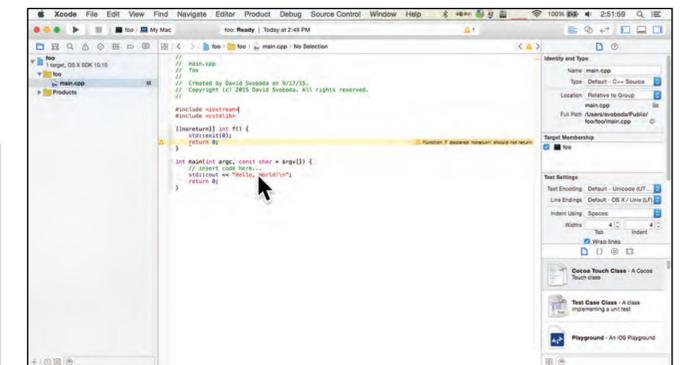
Providing immediate feedback to developers about weaknesses in programs as code is being developed encourages secure coding without generating vast collections of diagnostics later in the SDLC.

Checking C/C++ rule violations

- Exception
- Evaluation ordering
- Function return
- Constructor
- Assertion

Checking Java rule violations

- Override
- I/O



New rules for threaded programs and C++ form a more secure underpinning for object-oriented software deployed on multithreaded architectures. New checkers can locate potential vulnerabilities, and, when integrated into IDEs, provide immediate and effective feedback to developers.

Contact: Daniel Plakosh dplakosh@cert.org

Copyright 2015 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

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.

This material has been approved for public release and unlimited distribution except as restricted below.

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

DM-0002947