## Increase Adoption of Secure Coding Standards

Dan Plakosh

Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213



© 2015 Carnegie Mellon University

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

Software Engineering Institute Carnegie Mellon University



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.

This material was prepared for the exclusive use of SEI Research Review and may not be used for any other purpose without the written consent of permission@sei.cmu.edu.

Carnegie Mellon® and CERT® are registered marks of Carnegie Mellon University. DM-0002771



## Outline



### Fundamentals: Prescriptive Rules

- Maintain the C rules
- New computation models: threads
- Major language updates: C++
- Reducing friction of adoption
  - Improving analyst productivity
  - Immediate feedback and correction
  - Catching more violations
- Summary

## **Fundamentals: C Coding Rules**

#### 🚔 SEI SERIES • A CERT® BOOK

## The CERT<sup>®</sup> C Coding Standard

98 Rules for Developing Safe, Reliable, and Secure Systems

Second Edition

# Robert C. Seacord

- Specific, prescriptive advice for programmers, checkers and IDEs
- Collected wisdom of programmers and tools vendors
  - Fed by community wiki started in Spring 2006
  - 1,576 registered contributors
  - Basis for ISO Standard
- Continuously updated to reflect best practices and language evolution

## **Fundamentals: Expanding Coverage**





#### New computation model: C threads

- 9 unspecified behaviors representing programming weaknesses in two broad categories
  - Inter-thread communication
  - Thread-specific storage

Example: the tss\_create function which creates thread-specific storage and assigns a destructor but does not specify when the destructor is called.

#### Major language updates for

C++

- 24 new rules in FY15 specifying C++ weaknesses
- 60 existing C++ rules updated in FY15

New and updated rules published on http://www.cert.org/securecoding/publications/secure-codingenewsletter.cfm

> SEI Research Review 2015 October 7–8, 2015

## **Adoption: Improving Analyst Productivity**



Improve expert review productivity by focusing on high priority violations

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

Maintain record of decisions

SEI Research Review 2015 October 7–8, 2015 © 2015 Carnegie Mellon University 6 Distribution Statement A: Approved for Public Release: Distribution is Unlimited

## **Adoption: Immediate Feedback**

🕨 🔹 🕨 🔠 🔳 Ittel) 🛲 Ny	Inc. Sort Ready   Today It P ( 8 PM	A 1	= +
5 0 <b>0</b> 0 0 0 0	🔢 🔍 🗦 📘 teo) 🛅 teo) 💭 main ago) his Devenion	< <u>A</u> :	D ()
Normal Section 4.1	<pre>// Provide State St</pre>		Hanning and the second

ile Edit Source Refactor Nevigate	Search Broject Run Win	idove Hdp		
4 • 12 • 12 10 (2) 17 (2) 10 • 13 • 13	• 0 • % • @ 0 # •	12 × 2 × 9 a • 4 ×	Quick Access 🛛 🕫 🕸 Java 🔶 Deb	ug 📩 Git 🏴 Findbugs
9 (a) folder 1 (a) (b) (c) (c) (c) (c) (c) (c) (c) (c) (c) (c	D Herealywa         D Hereal/Wei           40         public star           5         new Fil           6         )           7*         public star           8*         public star           9         byte di           11         ///.           12         )           13         )           14         15*           15         public voii           16         int ini           17         byte di	<pre>kelousedajava ? Dim tic void main(String) le("foo").mkoir(); d nccei(fileInputStri tic; (Coita = (byte) in.r d csi(FileInputStream buff; ata;</pre>	writhdynoidejone (] angs) { eam in) throws IDException { main) throws IDException {	
	sug Receves II	v o	sag into a     longine 10     longine 10     longine 10     longine 10     longino     saccolfittingtSteam nervoe reas         metalogino         metalogino	A B B B B B B B B B B B B B B B B B
·			Rank: Scary (7), confidence: No	rmal -

# Moving rules into IDE improves application of secure coding

- · Early feedback corrects errors on introduction
- · Exceptions are understood in context
- Exceptions can be marked as resolved to eliminate redundant consideration

#### Clang static analyzer (C based languages)

- Widely used open source front end for popular compilers
   and IDEs
- Checkers available now in "Top-of-Tree" by early adopters
- Expect to be generally available in Clang's yearly release

#### FindBugs (Java)

· Integrated into Eclipse and Jdeveloper

SEI Research Review 2015 October 7-8, 2015 © 2015 Carnegie Mellon University Distribution Statement A: Approved for Public Release Distribution is Unlimited

## **Adoption: Catching More Violations Thru Checkers**



Increase adoption through automated checkers of rule violations

- Checking C/C++ rule violations
  - Exception
     Constructor
  - Function return
     Assertion
  - Evaluation ordering / side effects
    Example:
     int a = 14;
     int b = sizeof(a++);
     std::cout << a << ", " << b << std::endl;
     a is still 14 after b has been initialized</pre>
- Checking Java rule violations
  - Override
    - I/O
      Example:
      byte data;
      while ((data = (byte) in.read()) != -1) {
       // ...
      }

SEI Research Review 2015 October 7–8, 2015 © 2015 Carnegie Mellon University

Distribution is Unlimited

## Summary

- Maintained C Coding rules updated to reflect best practices and language evolution
- Developed 25 new rules for C++ and updated
   60 existing C++ rules
- Developed a web application to improve analyst productivity
- Introducing checking earlier in the development process
  - ✓ developed checkers for clang and FindBugs