CERT

Secure Coding Initiative

Robert C. Seacord
NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE 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.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation 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 use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.
Presenter Bio

Robert Seacord began programming (professionally) for IBM in 1982 and has been programming in C since 1985. Robert leads the Secure Coding Initiative at the CERT, located at Carnegie Mellon’s Software Engineering Institute (SEI). He is author of The CERT C Secure Coding Standard (Addison-Wesley, 2009), Secure Coding in C and C++ (Addison-Wesley, 2005), Building Systems from Commercial Components (Addison-Wesley, 2002) and Modernizing Legacy Systems (Addison-Wesley, 2003).
Secure Coding Initiative

Initiative Goals
Work with software developers and software development organizations to eliminate vulnerabilities resulting from coding errors before they are deployed.

Overall Thrusts
Advance the state of the practice in secure coding
Identify common programming errors that lead to software vulnerabilities
Establish standard secure coding practices
Educate software developers

Current Capabilities
Secure coding standards
www.securecoding.cert.org
Source code analysis and conformance testing
Training courses
Involved in international standards development.

The CERT® C Secure Coding Standard
Secure Coding in C and C++
Robert C. Seacord
Secure Coding in the SDLC

- Requirements
- Architecture
- Design
- Implementation
- Testing
- Deployment
- Operation and Maintenance

SEI CERT®

Secure Coding

Improved Systems

Vulnerability Remediation

Repaired Systems
Increasing Vulnerabilities

Reacting to vulnerabilities in existing systems is not working
CERT Secure Coding Initiative

**Reduce** the number of vulnerabilities to a level where they can be handled by computer security incident response teams (CSIRTs)

**Decrease** remediation costs by eliminating vulnerabilities *before* software is deployed
Fun With Integers

```c
char x, y;
x = -128;
y = -x;

if (x == y) puts("1");
if ((x - y) == 0) puts("2");
if ((x + y) == 2 * x) puts("3");
if (((char)(-x) + x) != 0) puts("4");
if (x != -y) puts("5");
```

Lesson: Process is irrelevant without a strong fundamental knowledge of the language and environment.
Secure Coding Roadmap

University courses
- CMU
- Purdue
- University of Florida
- Santa Clara University
- St. John Fisher College

SEI Secure Coding Course

Licensed to:
- Computer Associates
- Siemens
- SANS

Influence International Standard Bodies

Tool Test Suite

Application Conformance Testing

Adoption by Analyzer Tools

Adoption by software developers
- Lockheed Martin Aeronautics
- General Atomics

Secure Design Patterns

Breadth of impact

2003
Time
2010
Products and Services

CERT Secure Coding Standards
CERT SCALe (Source Code Analysis Laboratory)
TSP Secure
Training courses
Research
CERT Secure Coding Standards

Establish coding guidelines for commonly used programming languages that can be used to improve the security of software systems under development. Based on documented standard language versions as defined by official or de facto standards organizations.

Secure coding standards are under development for:

- C programming language (ISO/IEC 9899:1999)
- C++ programming language (ISO/IEC 14882-2003)
- Java Platform Standard Edition 6
Secure Coding Web Site (Wiki)

www.securecoding.cert.org

Rules are solicited from the community

Published as candidate rules and recommendations on the CERT Wiki.

Threaded discussions used for public vetting

Candidate coding practices are moved into a secure coding standard when consensus is reached
Noncompliant Examples & Compliant Solutions

Noncompliant Code Example
In this noncompliant code example, the `char` pointer `p` is initialized to the address of a string literal. Attempting to modify the string literal results in undefined behavior.

```
char *p = "string literal"; p[0] = 'S';
```

Compliant Solution
As an array initializer, a string literal specifies the initial values of characters in an array as well as the size of the array. This code creates a copy of the string literal in the space allocated to the character array `a`. The string stored in `a` can be safely modified.

```
char a[] = "string literal"; a[0] = 'S';
```
CERT C Secure Coding Standard Rules (89)

- Preprocessor (PRE)
- Declarations and Initialization (DCL)
- Expressions (EXP)
- Integers (INT)
- Floating Point (FLP)
- Arrays (ARR)
- Characters and Strings (STR)
- Memory Management (MEM)
- Input Output (FIO)
- Environment (ENV)
- Signals (SIG)
- Error Handling (ERR)
- Miscellaneous (MSC)
- POSIX (POS)

CERT C Secure Coding Standard Recommendations (132)

- Preprocessor (PRE)
- Declarations and Initialization (DCL)
- Expressions (EXP)
- Integers (INT)
- Floating Point (FLP)
- Arrays (ARR)
- Characters and Strings (STR)
- Memory Management (MEM)
- Input Output (FIO)
- Environment (ENV)
- Signals (SIG)
- Error Handling (ERR)
- Miscellaneous (MSC)
- POSIX (POS)
CERT Mitigation Information

Vulnerability Note VU#649732
This vulnerability occurred as a result of failing to comply with rule FIO30-C of the CERT C Programming Language Secure Coding Standard.

Examples of vulnerabilities resulting from the violation of this recommendation can be found on the CERT website.

US CERT Technical Alerts
Secure Coding Standard Applications

Establish secure coding practices within an organization

- may be extended with organization-specific rules
- cannot replace or remove existing rules

Train software professionals

Certify programmers in secure coding

Establish requirements for software analysis tools

Certify software systems
Industry Adoption

Software developers that require code to conform to The CERT C Secure Coding Standard:

Software tools that (partially) enforce The CERT C Secure Coding Standard:
Industry Adoption

LDRA ships new TBsecure™ complete with CERT C Secure Coding programming checker

Screenshot from the LDRA tool suite shows the selection of the CERT C secure coding standard from the C standards models
Products and Services

CERT Secure Coding Standards
CERT SCALe (Source Code Analysis Laboratory)
TSP Secure
Training courses
Research
Enforcing Coding Standards

Increasingly, application source code reviews are dictated.

The Payment Card Industry (PCI) Data Security Standard requires that companies with stored credit card or other consumer financial data

- install application firewalls around all Internet-facing applications or
- have all the applications' code reviewed for security flaws.

This requirement could be met by a manual review of application source code or the proper use of automated application source code analyzer tools.
CERT SCALe (Source Code Analysis Lab)

Satisfy demand for source code assessments for both government and industry organizations.

Assess source code against one or more secure coding standards.

Provided a detailed report of findings.

Assist customers in developing conforming systems.
The use of secure coding standards defines a proscriptive set of rules and recommendations to which the source code can be evaluated for compliance.

- **INT30-C.** Provably nonconforming
- **INT31-C.** Documented deviation
- **INT32-C.** Conforming
- **INT33-C.** Provably Conforming
Products and Services

CERT Secure Coding Standards
CERT SCALe (Source Code Analysis Laboratory)
TSP Secure
Training courses
Research
Secure TSP

221 Guidelines

Security Manager

Source Code

static analysis tools, unit tests, and fuzz testing

Deploy

The CERT® C Secure Coding Standard

Robert C. Seacord
Products and Services

CERT Secure Coding Standards
CERT SCALe (Source Code Analysis Laboratory)
TSP Secure
Training Courses
Research
Secure Coding in C/C++ Course

Four day course provides practical guidance on secure programming

- provides a detailed explanation of common programming errors
- describes how errors can lead to vulnerable code
- evaluates available mitigation strategies
- [http://www.sei.cmu.edu/products/courses/p63.html](http://www.sei.cmu.edu/products/courses/p63.html)

Useful to anyone involved in developing secure C and C++ programs regardless of the application

Direct offerings in Pittsburgh, Arlington, and other cities

Partnered with industry

- Licensed to Computer Associates to train 9000+ internal software developers
- Licensed to SANS to provide public training
CMU CS 15-392 Secure Programming

Offered as an undergraduate elective in the School of Computer Science in S07, S08 and S09

• More of a vocational course than an “enduring knowledge” course.
• Students are interested in taking a class that goes beyond “policy”

Secure Software Engineering graduate course offered at INI in F08, F09

Working with NSF to sponsor a workshop in Mauritius to help universities throughout the world teach secure coding
Products and Services

CERT Secure Coding Standards
CERT SCALe (Source Code Analysis Laboratory)
TSP Secure
Training Courses
Research
As-if Infinitely Ranged (AIR) Integers

AIR integers is a model for automating the elimination of integer overflow and truncation in C and C++ code.

- integer operations either succeed or trap
- uses the runtime-constraint handling mechanisms defined by ISO/IEC TR 24731-1
- generates constraint violations for
  - signed overflow for addition, subtraction, multiplication, negation, and left shifts
  - unsigned wrapping for addition, subtraction, and multiplication
  - truncation resulting from coercion (not included in benchmarks)

SPECINT2006 macro-benchmarks

<table>
<thead>
<tr>
<th>Optimization Level</th>
<th>Control Ratio</th>
<th>Analyzable Ratio</th>
<th>% Slowdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>-00</td>
<td>4.92</td>
<td>4.60</td>
<td>6.96</td>
</tr>
<tr>
<td>-01</td>
<td>7.21</td>
<td>6.77</td>
<td>6.50</td>
</tr>
<tr>
<td>-02</td>
<td>7.38</td>
<td>6.99</td>
<td>5.58</td>
</tr>
</tbody>
</table>
CERT C and C++

Develop a holistic solution to the problem that includes

- An analyzability annex for the C1X standard
- As-if infinitely ranged (“AIR”) integers
- Safe Secure C/C++ methods (SSCC)
- C and C++ Secure Coding Guidelines

This solution eliminates the vulnerabilities:

- Writing outside the bounds of an object (e.g., buffer overflow)
- Reading outside the bounds of an object
- Arbitrary reads/writes (e.g., wild-pointer stores)
- Integer overflow and truncation

Prototype using Compass/ROSE and GCC
Prototype Design

Compiler

Source file

ROSE

diagnostics

Run-time pointer-checking library

Pre-linker

Linker

Safe/Secure Executable

Advice file

Internal representation (IR)

Modified Compiler Backend

Compiler Frontend
For More Information

Visit CERT® web sites:
http://www.cert.org/secure-coding/
https://www.securecoding.cert.org/

Contact Presenter
Robert C. Seacord
rcs@cert.org
(412) 268-7608

Contact CERT:
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh PA 15213-3890
USA