Secure Coding Initiative

Robert C. Seacord

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

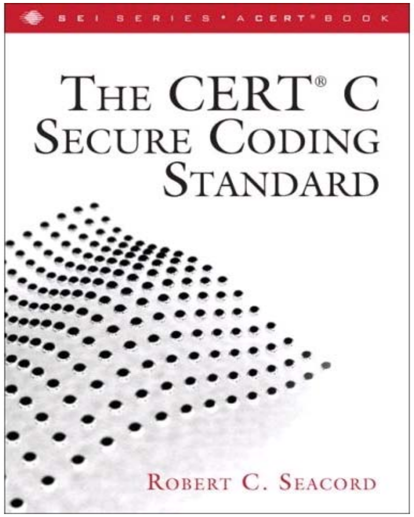
Secure coding standards

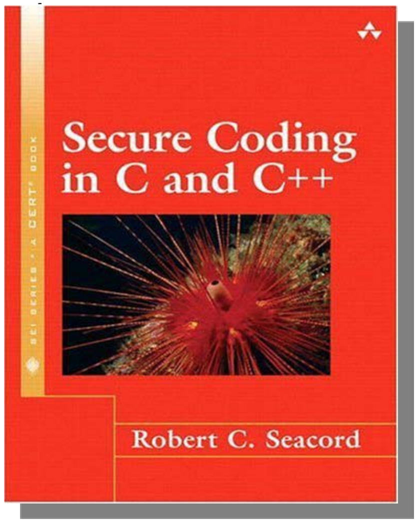
www.securecoding.cert.org

Source code analysis and conformance testing

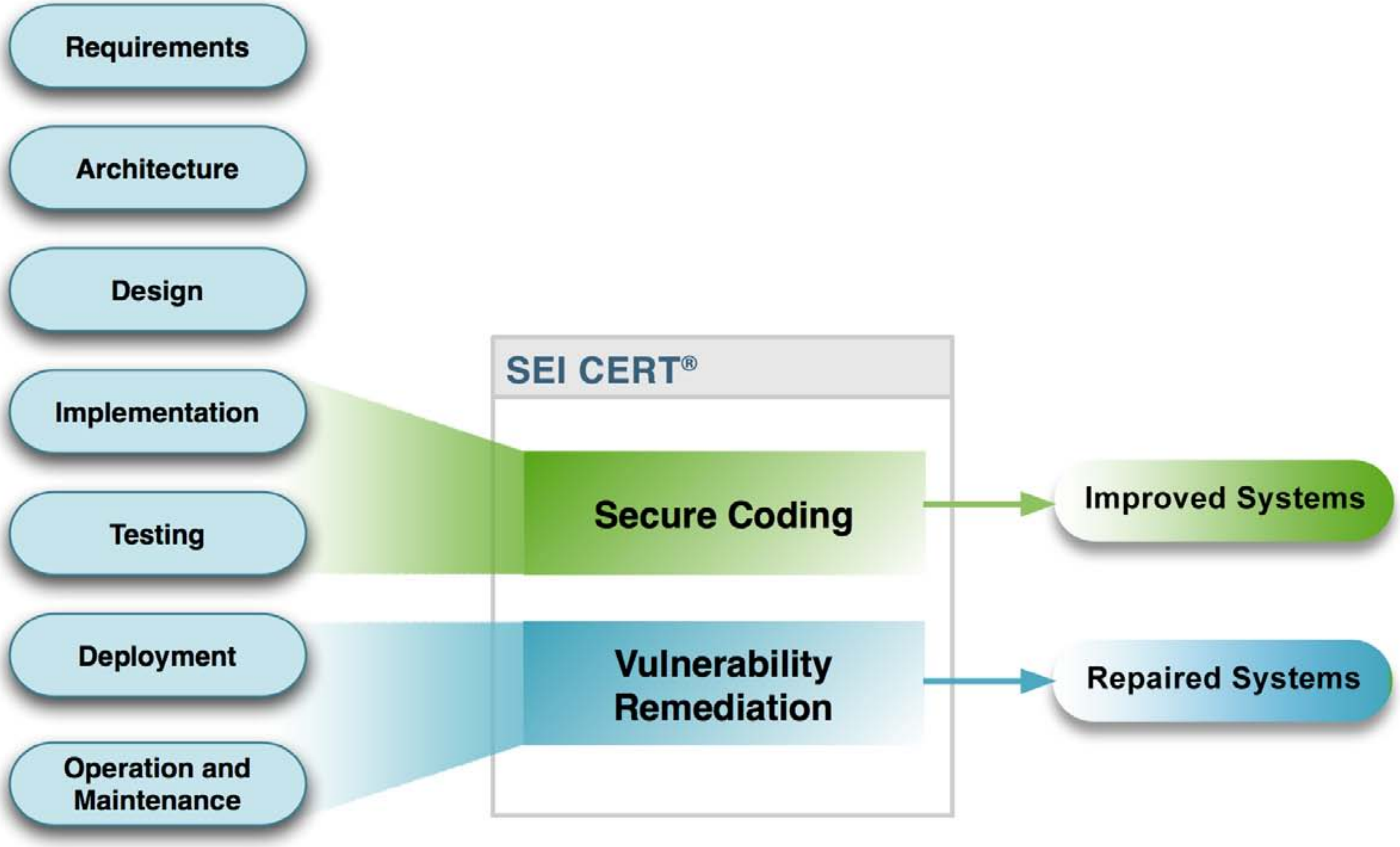
Training courses

Involved in international standards development.



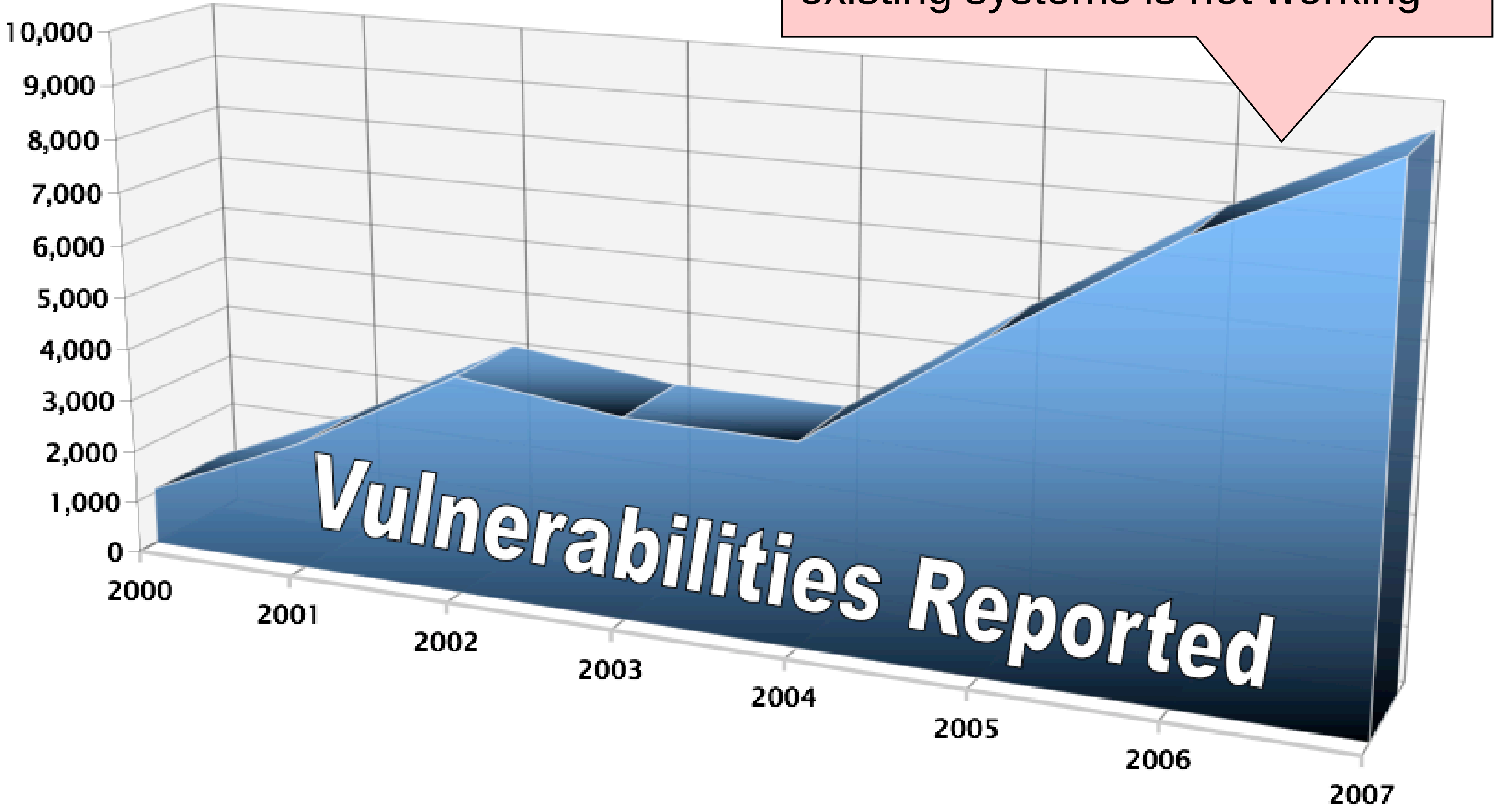


Secure Coding in the SDLC



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

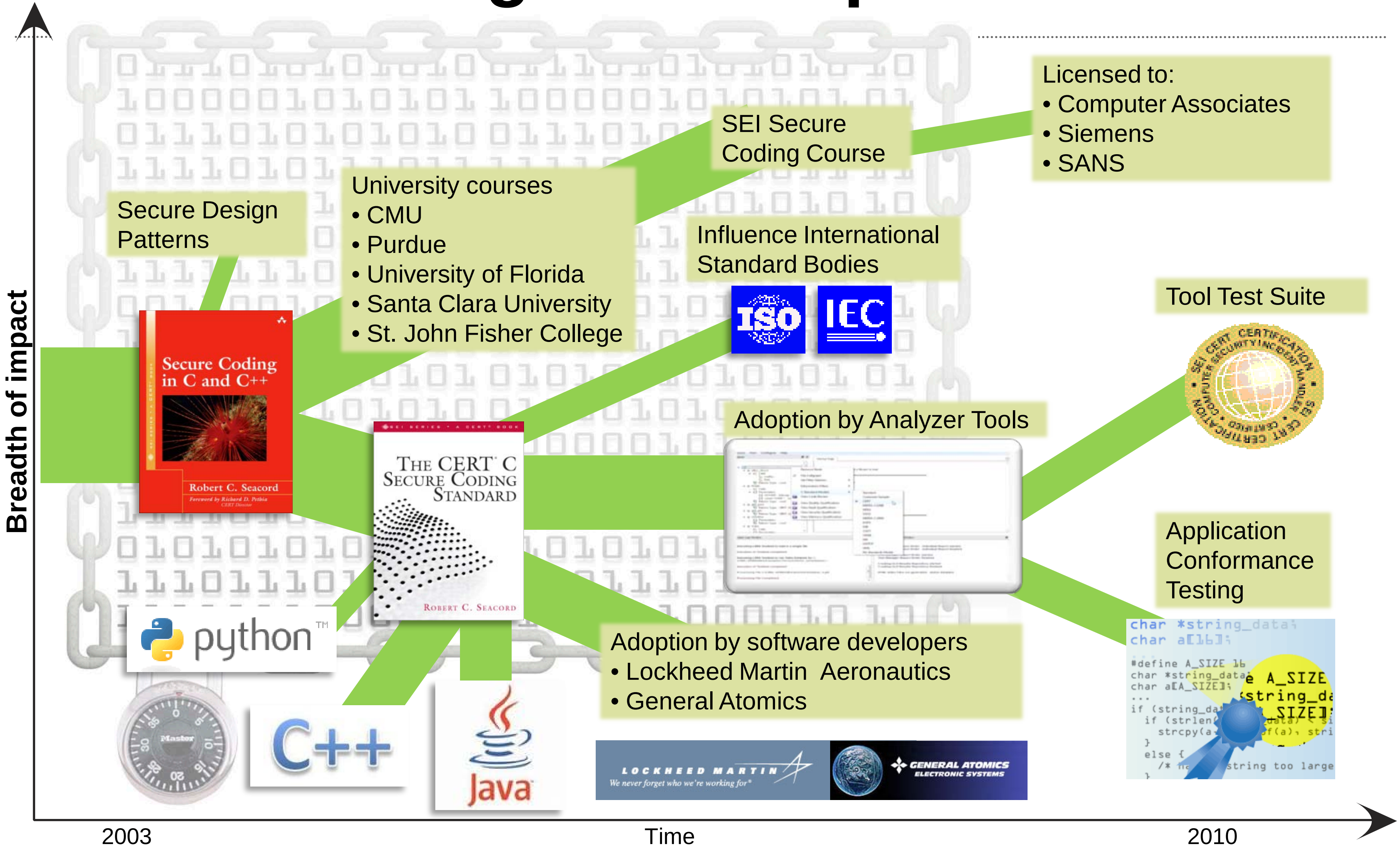
Fun With Integers

```
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



Secure Design Patterns

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

Adoption by Analyzer Tools

Application Conformance Testing

Adoption by software developers
- Lockheed Martin Aeronautics
- General Atomics

Breadth of impact
Time
2003
2010
Secure Coding in C and C++
Robert C. Seacord
The CERT C Secure Coding Standard
Robert C. Seacord
ISO
IEC
python
C++
Java
LOCKHEED MARTIN
We never forget who we're working for
GENERAL ATOMICS ELECTRONIC SYSTEMS
SEI CERT CERTIFICATION

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

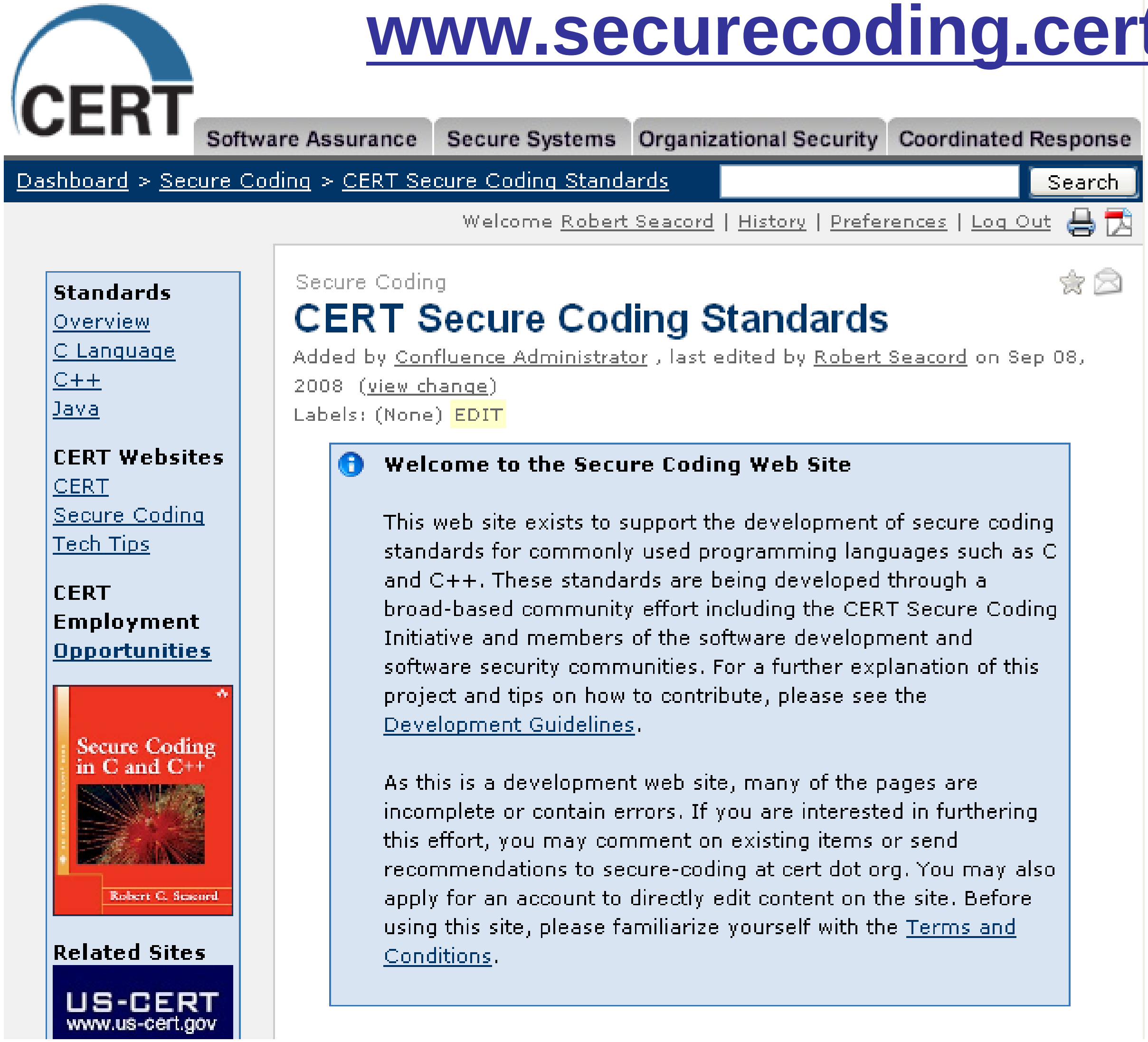
Based on documented standard language versions as defined by official or de facto standards organizations

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



Software Assurance | Secure Systems | Organizational Security | Coordinated Response

Dashboard > Secure Coding > CERT Secure Coding Standards

Welcome Robert Seacord | History | Preferences | Log Out

Standards
Overview
C Language
C++
Java

CERT Websites
CERT
Secure Coding
Tech Tips

CERT Employment Opportunities

Secure Coding in C and C++
Robert C. Seacord

Related Sites

US-CERT
www.us-cert.gov

Secure Coding

CERT Secure Coding Standards

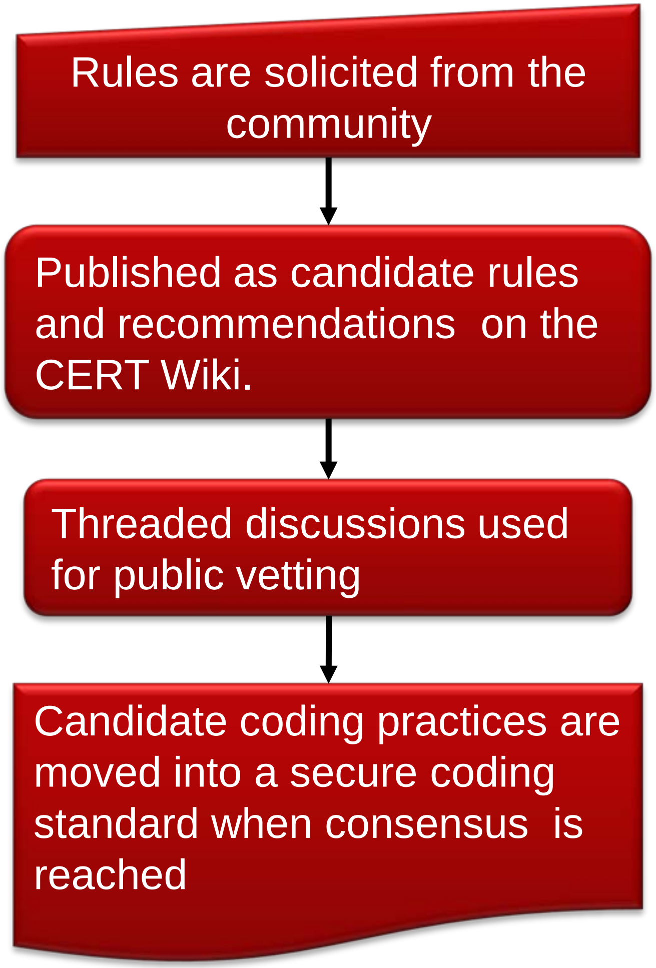
Added by Confluence Administrator , last edited by Robert Seacord on Sep 08, 2008 (view change)

Labels: (None) EDIT

Welcome to the Secure Coding Web Site

This web site exists to support the development of secure coding standards for commonly used programming languages such as C and C++. These standards are being developed through a broad-based community effort including the CERT Secure Coding Initiative and members of the software development and software security communities. For a further explanation of this project and tips on how to contribute, please see the Development Guidelines.

As this is a development web site, many of the pages are incomplete or contain errors. If you are interested in furthering this effort, you may comment on existing items or send recommendations to secure-coding at cert dot org. You may also apply for an account to directly edit content on the site. Before using this site, please familiarize yourself with the Terms and Conditions.



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.

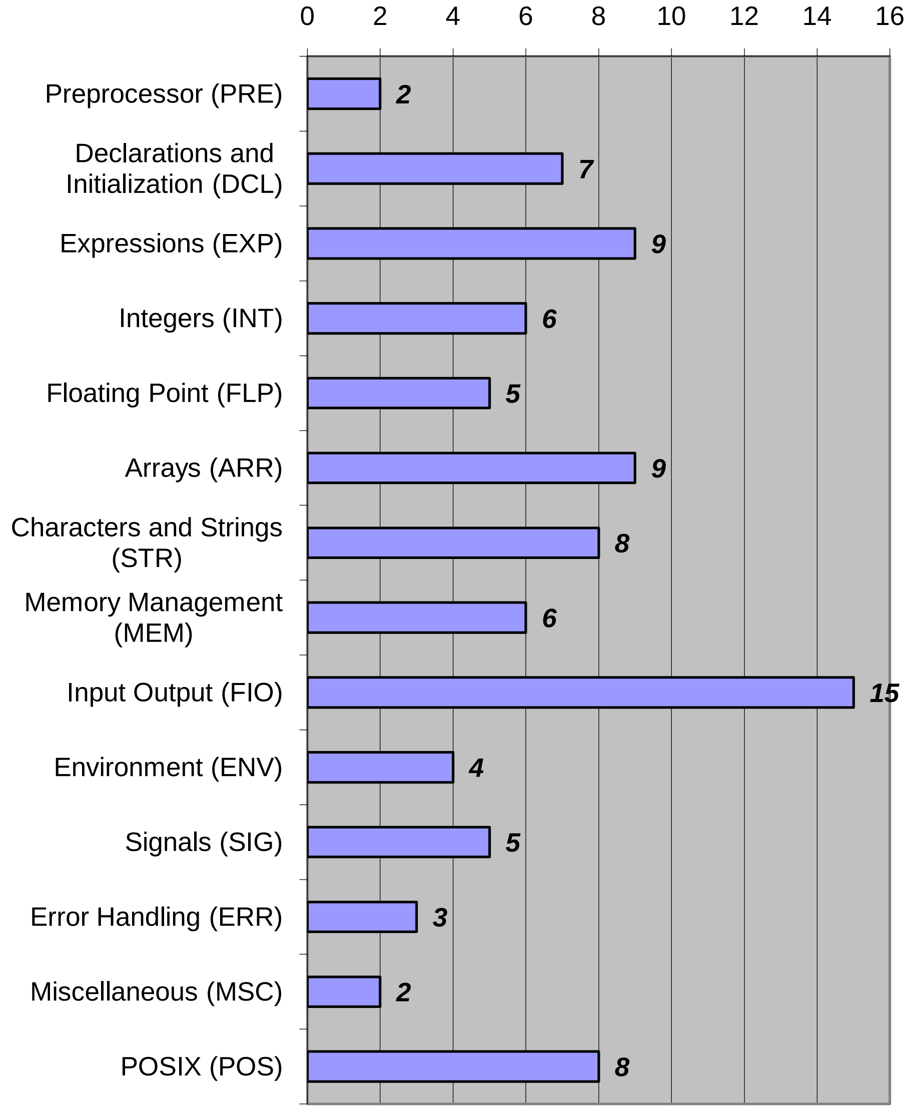
```
char *p = "string literal"; p[0] = 'S';
```

Compliant Solution

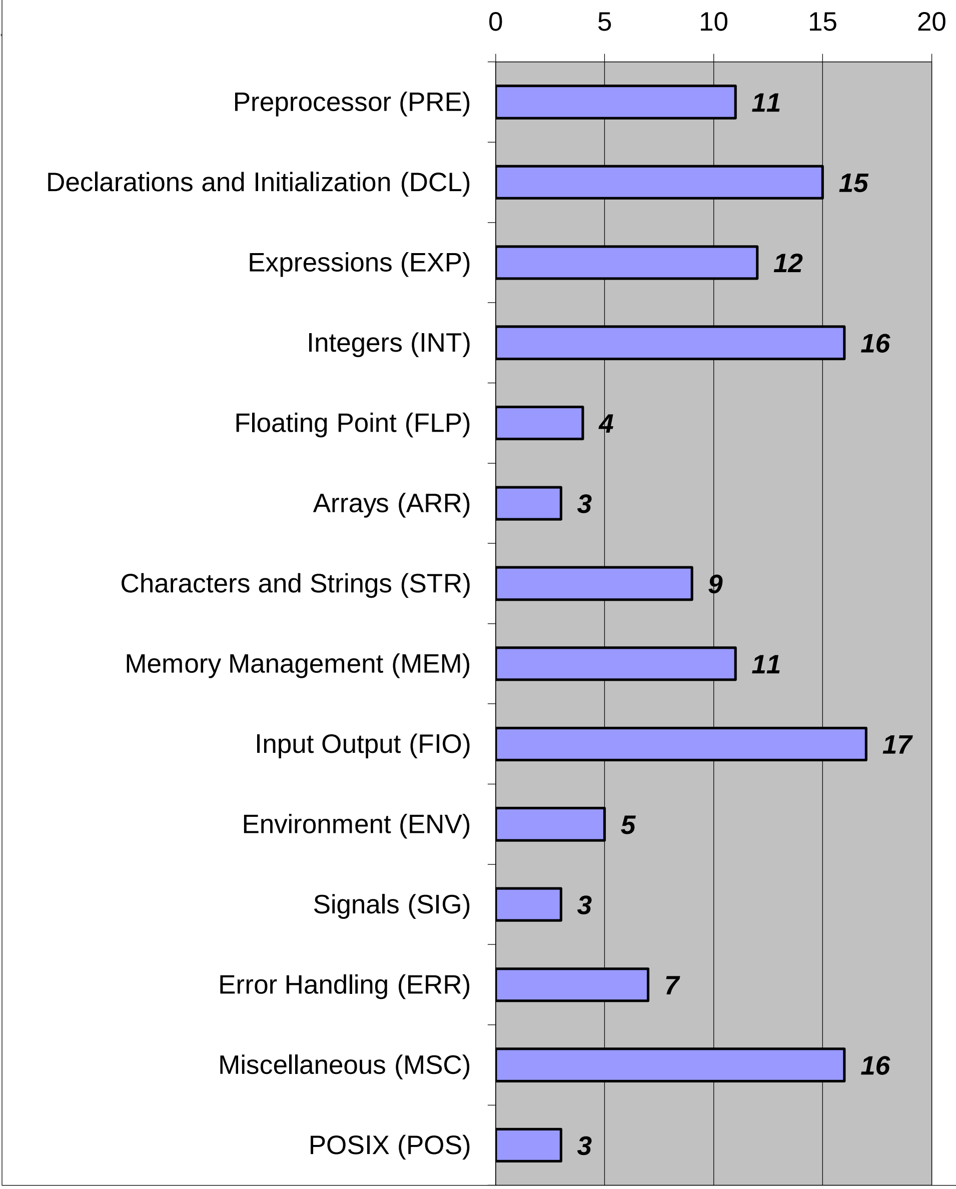
As an array initializer, a string literal specifies the initial values of characters in an array as well as the size of the array. This code creates a copy of the string literal in the space allocated to the character array **a**. The string stored in **a** can be safely modified.

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

CERT C Secure Coding Standard Rules (89)

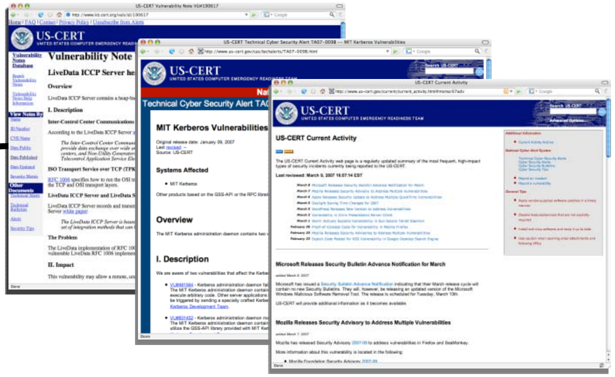


CERT C Secure Coding Standard Recommendations (132)



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.



US-CERT Vulnerability Note VU#190617

Vulnerability Note

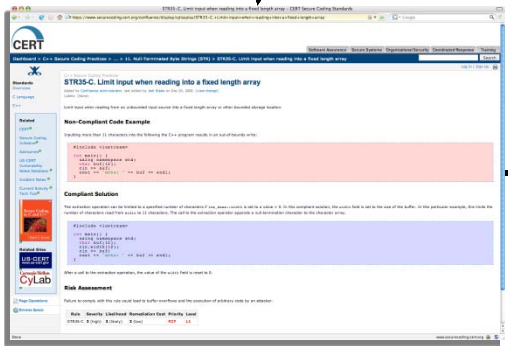
LiveData ICCP Server he

Technical Cyber Security Alert TA07-009B -- MIT Kerberos Vulnerabilities

MIT Kerberos Vulnerabilities

US-CERT Current Activity

US CERT Technical Alerts



STR35-C. Limit input when reading into a fixed length array - CERT Secure Coding Standards

Dashboard > C++ Secure Coding Practices > ... > 11. Null-Terminated Byte Strings (STR) > STR35-C. Limit input when reading into a fixed length array

STR35-C. Limit input when reading into a fixed length array

Limit input when reading from an unbounded input source into a fixed length array or other bounded storage location.

Non-Compliant Code Example

Inputting more than 11 characters into the following the C++ program results in an out-of-bounds write:

```
#include <iostream>

int main() {
  using namespace std;
  char buf[12];
  cin >> buf;
  cout << "echo: " << buf << endl;
}
```

Compliant Solution

```
#include <iostream>

int main() {
  using namespace std;
  char buf[12];
  cin.width(12);
  cin >> buf;
  cout << "echo: " << buf << endl;
}
```

After a call to the extraction operation, the value of the width field is reset to 0.

Risk Assessment

Failure to comply with this rule could lead to buffer overflows and the execution of arbitrary code by an attacker.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
STR35-C	3 (high)	3 (likely)	3 (low)	P27	L1

CERT Secure Coding Standard

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

Secure Coding Standard Applications

Establish secure coding practices within an organization

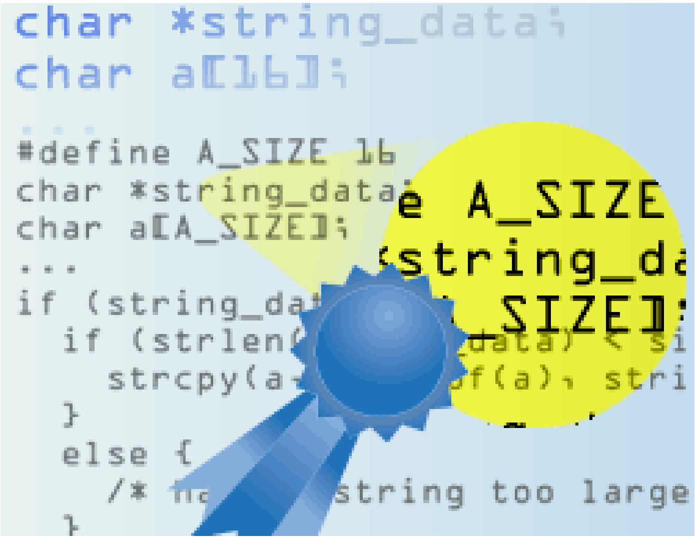
- may be extended with organization-specific rules
- cannot replace or remove existing rules

Train software professionals

Certify programmers in secure coding

Establish requirements for software analysis tools

Certify software systems



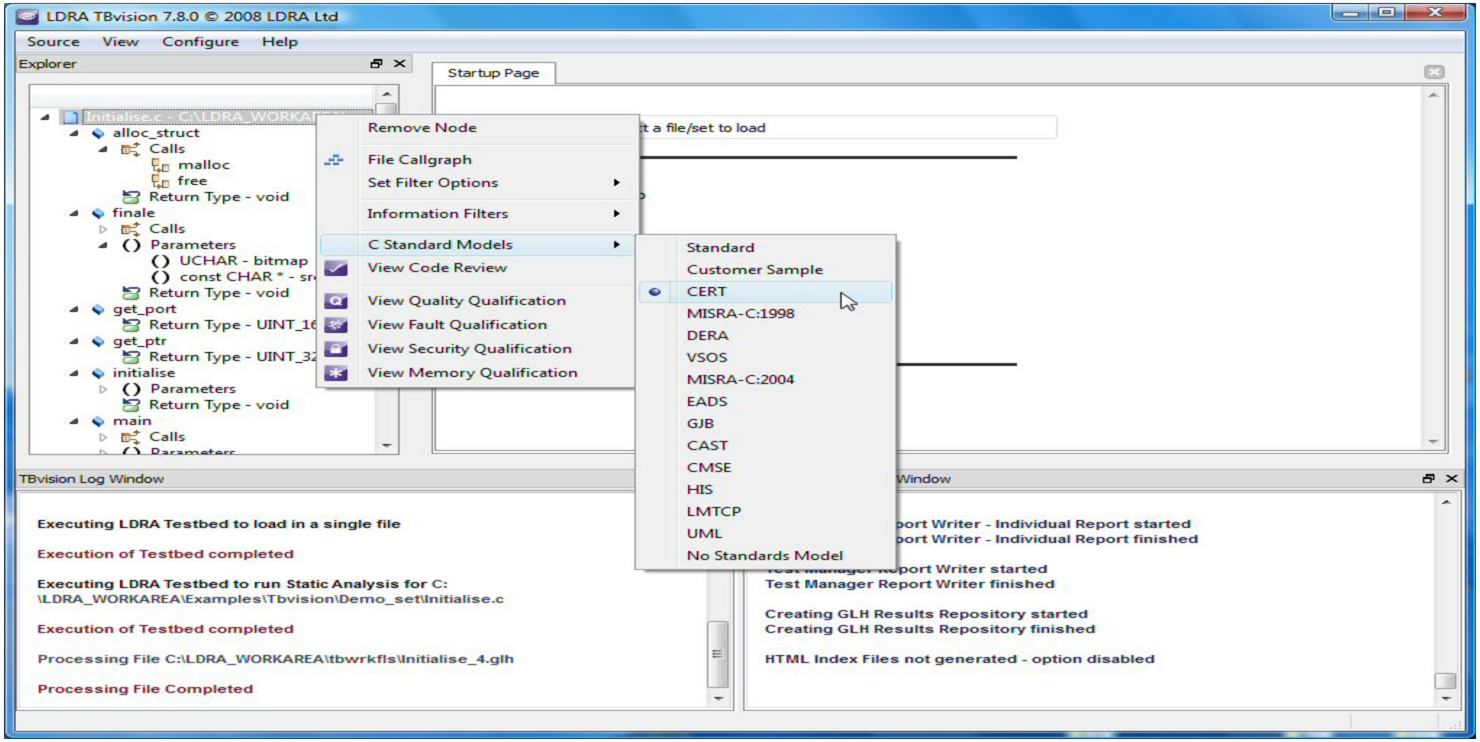
Industry Adoption

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



```
Executing LDRA Testbed to load in a single file

Execution of Testbed completed

Executing LDRA Testbed to run Static Analysis for C:
\LDRA_WORKAREA\Examples\Tbvision\Demo_set\Initialise.c

Execution of Testbed completed

Processing File C:\LDRA_WORKAREA\tbwrkfls\Initialise_4.glh

Processing File Completed
```

```
port Writer - Individual Report started
port Writer - Individual Report finished

Report Writer started
Test Manager Report Writer finished

Creating GLH Results Repository started
Creating GLH Results Repository finished

HTML Index Files not generated - option disabled
```

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.

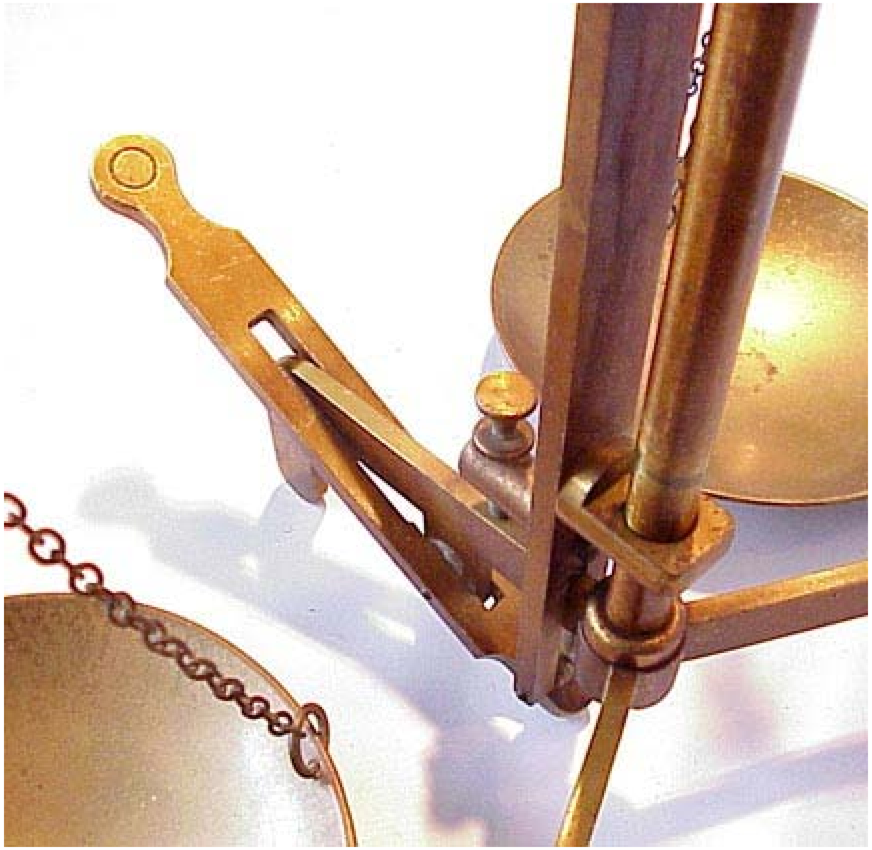
CERT SCALe (Source Code Analysis Lab)

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

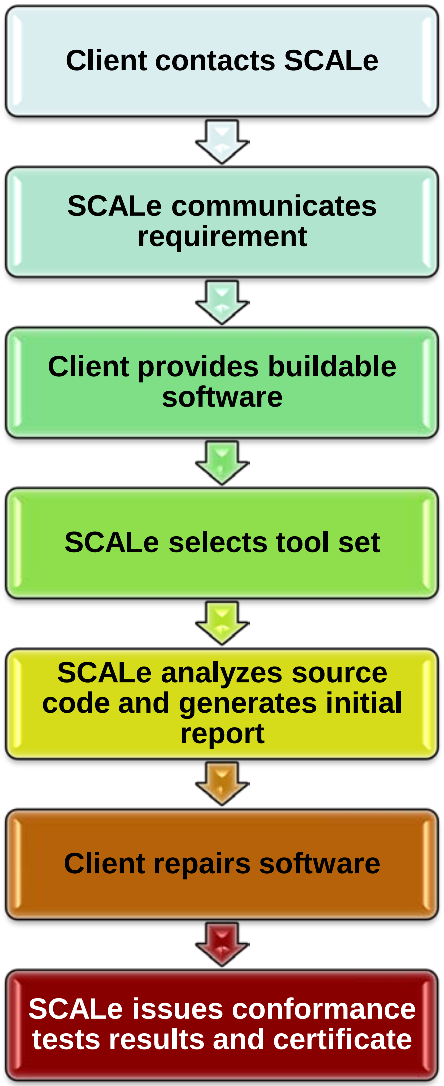
Assess source code against one or more secure coding standards.

Provided a detailed report of findings.

Assist customers in developing conforming systems.



Conformance Testing


Client contacts SCALe
SCALe communicates requirement
Client provides buildable software
SCALe selects tool set
SCALe analyzes source code and generates initial report
Client repairs software
SCALe issues conformance tests results and certificate

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

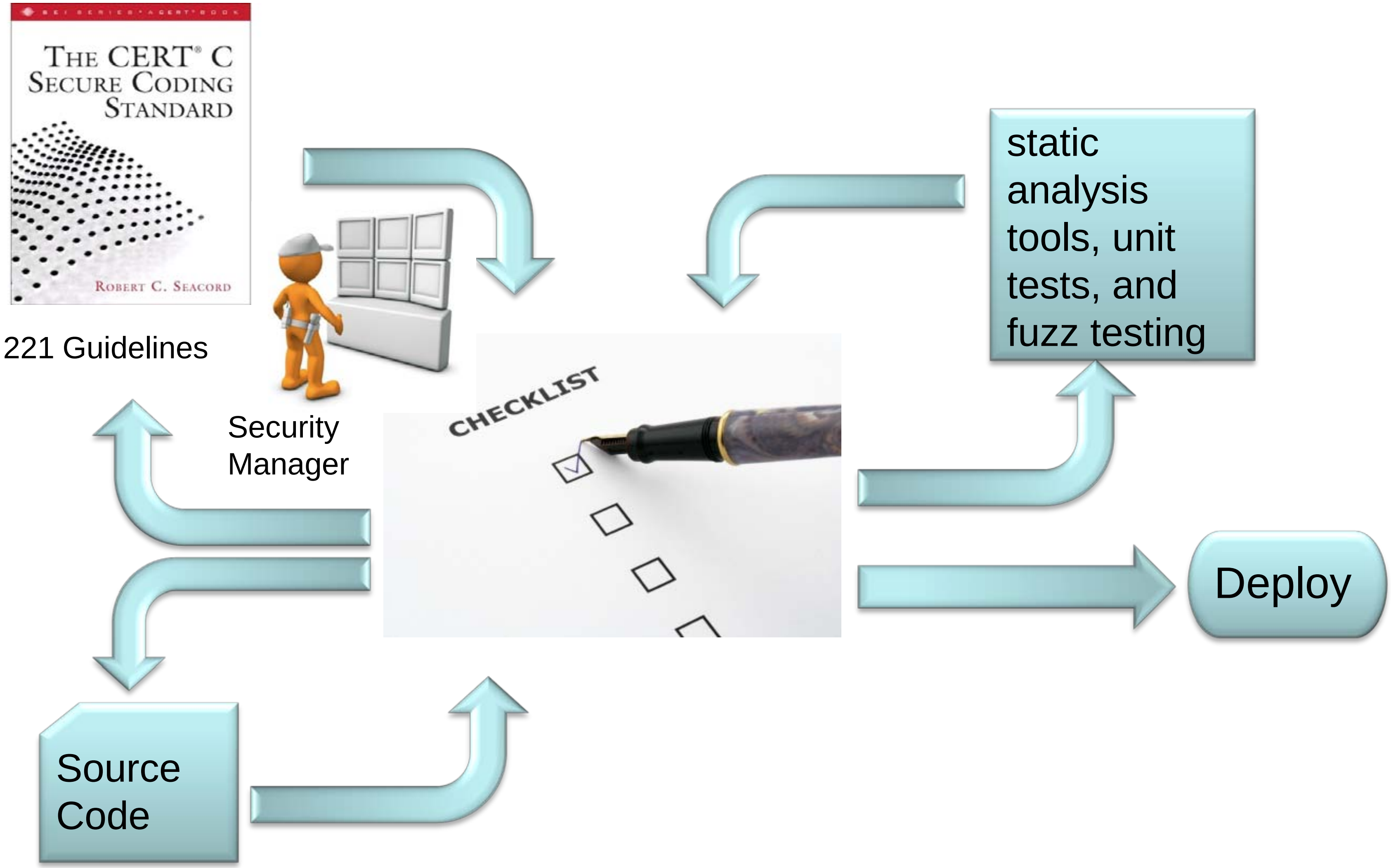
CERT SCALe (Source Code Analysis Laboratory)

TSP Secure

Training courses

Research

Secure TSP



221 Guidelines

Security Manager

static analysis tools, unit tests, and fuzz testing

Deploy

Source Code

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

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

Optimization Level	Control Ratio	Analyzable Ratio	% Slowdown
-O0	4.92	4.60	6.96
-O1	7.21	6.77	6.50
-O2	7.38	6.99	5.58

CERT C and C++

Develop a holistic solution to the problem that includes

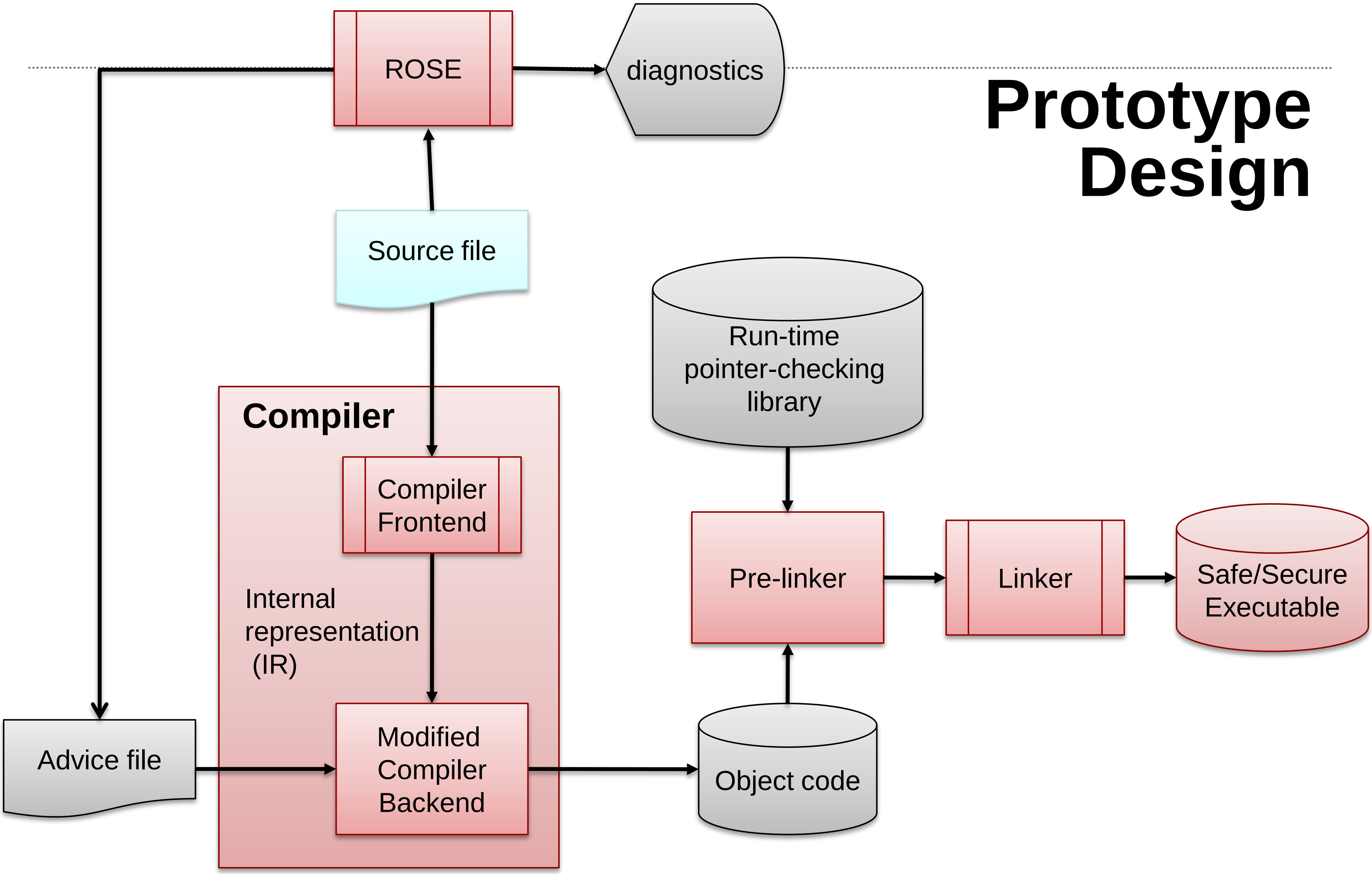
- An analyzability annex for the C1X standard
- As-if infinitely ranged (“AIR”) integers
- Safe Secure C/C++ methods (SSCC)
- C and C++ Secure Coding Guidelines

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



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