CERT

INFOSEC WORLD
CONFERENCE & EXPO 2014

# Why Can’t Johnny Program Securely?

Session #A9
Wednesday, April 9
9:45 AM – 10:45 AM


Robert C. Seacord
Secure Coding Technical Manager

DM-0001394

# What Is Secure Software Development?

Not the same as developing security software, such as

- Firewalls, intrusion detection, encryption
- Protection of the environment within which the software operates

Secure software development is building defect-free software that can function robustly in its operational production environment and is resistant to attack.

# Application Security

# Most Vulnerabilities Are Caused by Programming Errors

64% of the vulnerabilities in the National Vulnerability Database in 2004 were due to programming errors

- 51% of those were due to classic errors like buffer overflows, cross-site scripting, injection flaws
- Heffley/Meunier (2004): Can Source Code Auditing Software Identify Common Vulnerabilities and Be Used to Evaluate Software Security?

Cross-site scripting, SQL injection at top of the statistics (CVE, Bugtraq) in 2006

“We wouldn’t need so much network security if we didn’t have such bad software security.”

—Bruce Schneier

# Agenda

Education and assessment of programmers in major software markets

Programming is hard

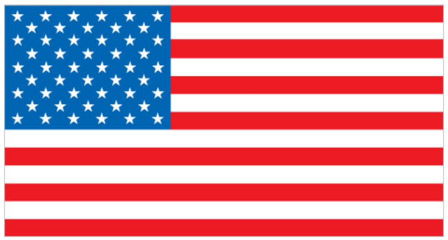
Limitations of analysis and testing

Use and application of secure coding standards

Conformance testing using SCALe (Source Code Analysis Laboratory)

# Software Developer Demand

There are about 18.2 million software developers worldwide; due to rise to 26.4 million by 2019, a 45% increase*



U.S. leads the world in software developers, with about 3.6 million.

The U.S. Bureau of Labor Statistics estimates that

- 76,000 software development jobs are added annually
- software developer employment will grow 22% from 2012 to 2022

The Indian IT industry employs nearly 2.75 million people and added 180,000 new positions in 2013.

By 2018, India will have 5.2 million developers, a nearly 90% increase.

* Evans Data Corp. in its latest Global Developer Population and Demographic Study

# Gap in Computer Science Workforce

The U.S. Bureau of Labor Statistics forecasts that during the period of 2008–2018

– close to 140,000 job openings in computing fields will be created

– only 50,000 students will receive degrees in computer science and related areas.

India's National Association of Software and Service Companies (NASSCOM) studies indicate that of roughly 400,000 university graduates earning technical degrees in 2006-2007 only 100,000 suitable candidates were found suitable by Indian Companies for training.

# Computer Science Education at CMU

The School of Computer Science at Carnegie Mellon University is undergoing major revisions to its introductory course sequence.

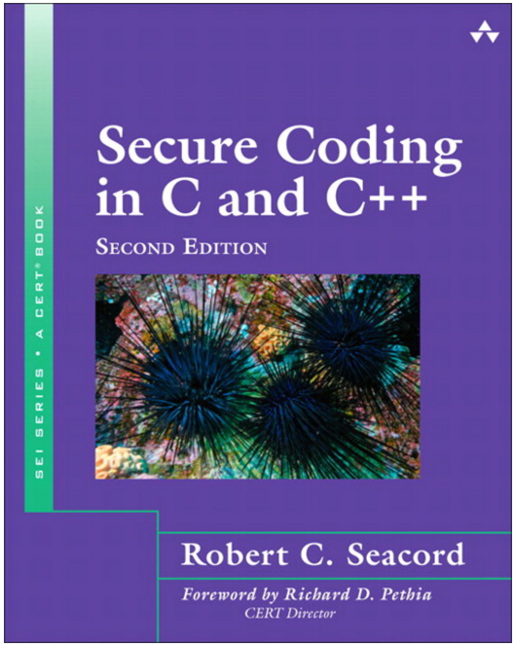
Major changes include:

- Promoting computational thinking
- Increasing software reliability
  - Safety critical systems
  - Security vulnerabilities
- Preparing for a world of parallel computation

# Secure Coding at CMU

The Computer Science Department at CMU has offered CS 15-392 “Secure Programming” as a computer science elective since 2007.

CMU’s Information Networking Institute has also offered 14-735 “Secure Software Engineering” in its Master of Science in Information Technology Information Security Track (MSIT-IS).



# Increasing Capacity

Increased capacity can be addressed, in part, by an increase in the productivity and efficiency of learners, that is, moving ever more learners ever more rapidly through course materials.

This need for throughput is matched by the need for quality.

Students must be able to apply what they have learned and be able to learn new things.

Effective secure coding requires a balance between

- high-level theory
- detailed programming-language expertise
- the ability to apply both in the context of developing secure software.

# Leveraged Expertise

Educating software developers properly requires great expertise.

While this expertise does exist, it tends to reside in individuals and organizations that are isolated from one another.

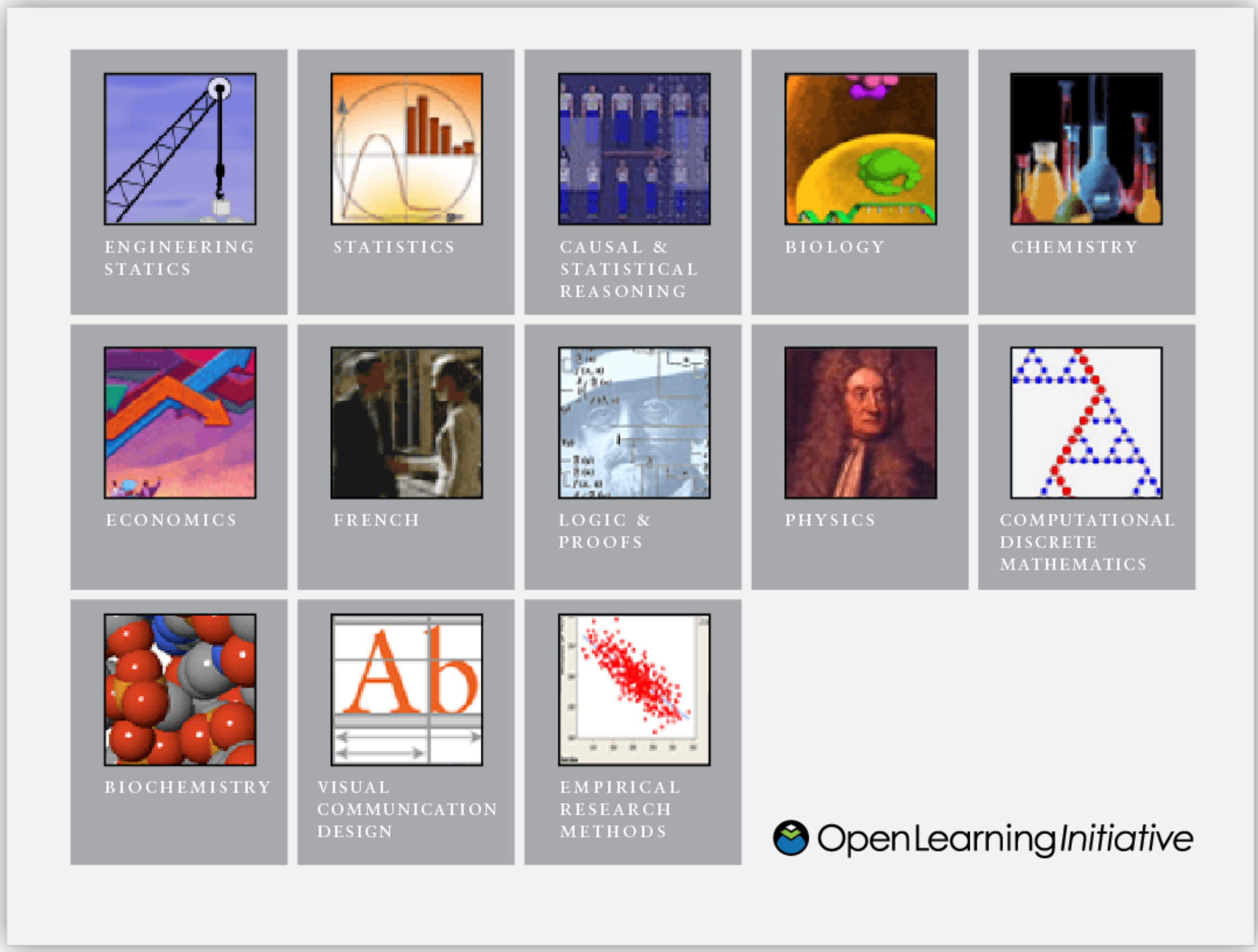
- These pockets of excellence, effective within their spheres, do not scale to meet the national demand.
- Even when practitioners do achieve significant improvement in the effectiveness of their instruction, this success is not shared or systematized.

Just as contemporary models for software development have rejected the isolated “hero programmer” in favor of a team- and process-driven engineering approach, current best practices in educational technology and research in learning science point away from the solo educator.

In the words of Herbert Simon, “Improvement in post-secondary education will require converting teaching from a ‘solo sport’ to a community based research activity.”

# What is CMU's Open Learning Initiative?

Scientifically-based online learning environments designed to improve both quality and productivity in higher education



# Secure Coding Course: Objectives 1

## Strings

- Recognize the different string types in C and C++ language programs.
- Select the appropriate byte character types for a given purpose.
- Identify common string manipulation errors.
- Explain how vulnerabilities from common string manipulation errors can be exploited.
- Identify applicable mitigation strategies, evaluate candidate mitigation strategies, and select the most appropriate mitigation strategy (or strategies) for a given context.
- Apply mitigation strategies to reduce the introduction of errors into new code or repair security flaws in existing code.

## Integer Security

- Explain and predict how integer values are represented for a given implementation.
- Predict how and when conversions are performed and describe their pitfalls.
- Select appropriate type for a given situation.
- Programmatically detect erroneous conditions for assignment, addition, subtraction, multiplication, division, and left and right shift.
- Recognize when implicit conversions and truncation occur as a result of assignment.
- Apply mitigation strategies to reduce introduction of errors into new code or repair security flaws in existing code.

# Secure Coding Course: Objectives 2

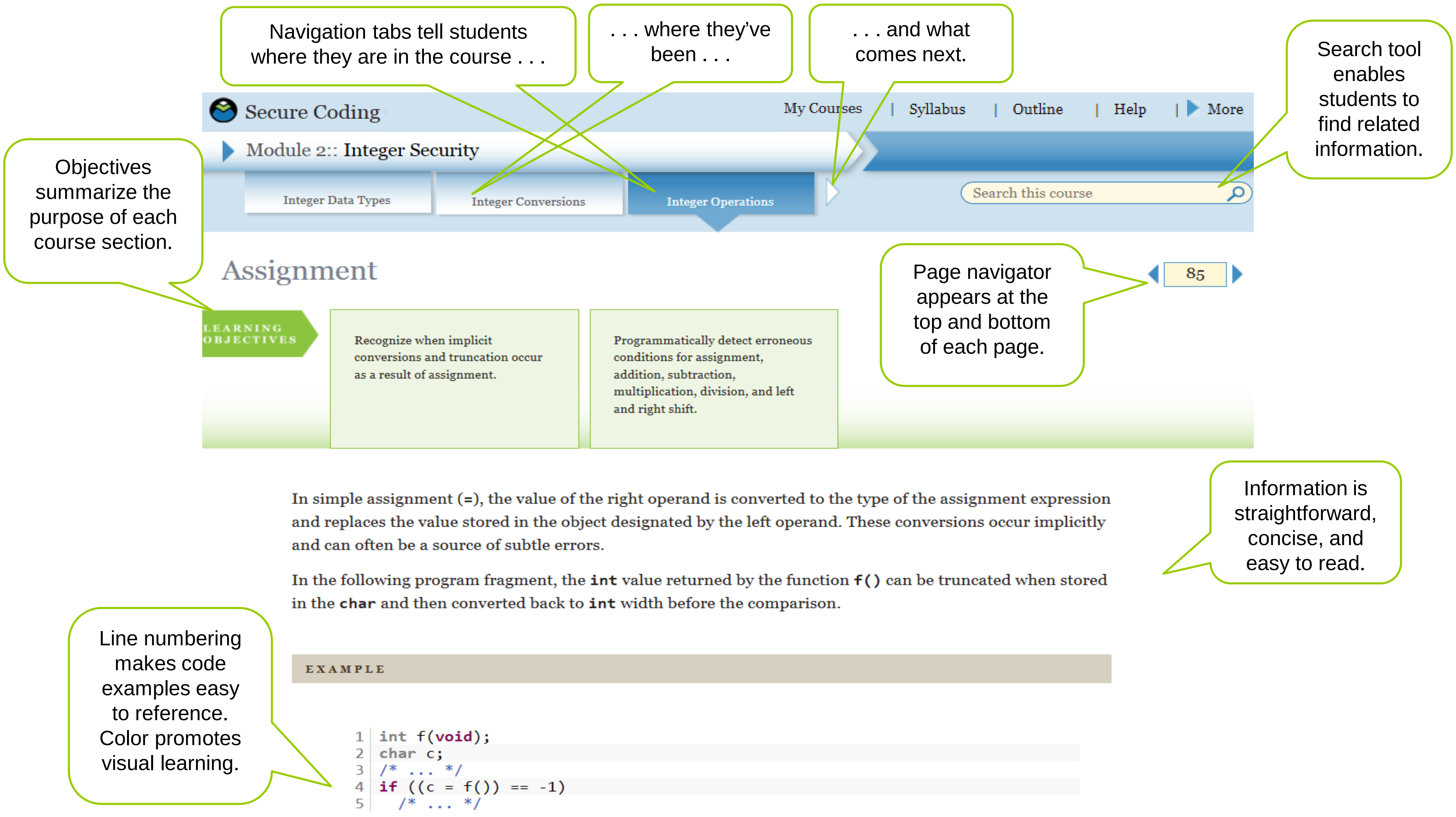
## Dynamic Memory

- Use standard C memory management functions securely.
- Align memory suitably.
- Explain how vulnerabilities from common dynamic memory management errors can be exploited.
- Identify common dynamic memory management errors.
- Perform C++ memory management securely.
- Identify common C++ programming errors when performing dynamic memory allocation and deallocation.
- Identify common dynamic memory management errors.

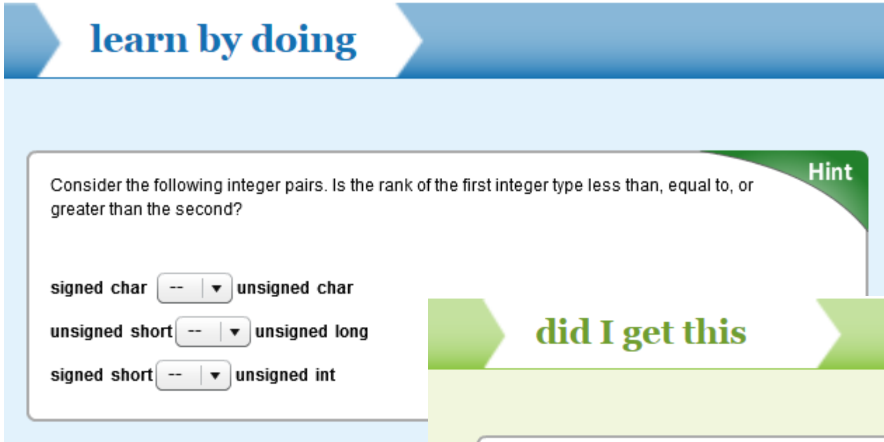
## Concurrency

- Define concurrency and it’s relationship with multithreading and parallelism.
- Calculate the potential performance benefits of parallelism in specific instances.
- Identify common errors in concurrency implementations.
- Identify common errors and attack vectors C++ concurrency programming.
- Apply common approaches for mitigating risks in C++ concurrency programming.
- Describe common vulnerabilities that occur from the incorrect use of concurrency.

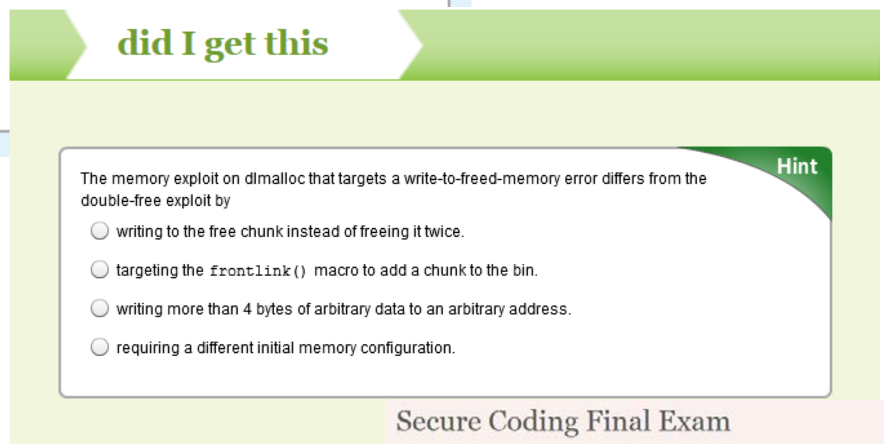
# Secure Coding Course Interface

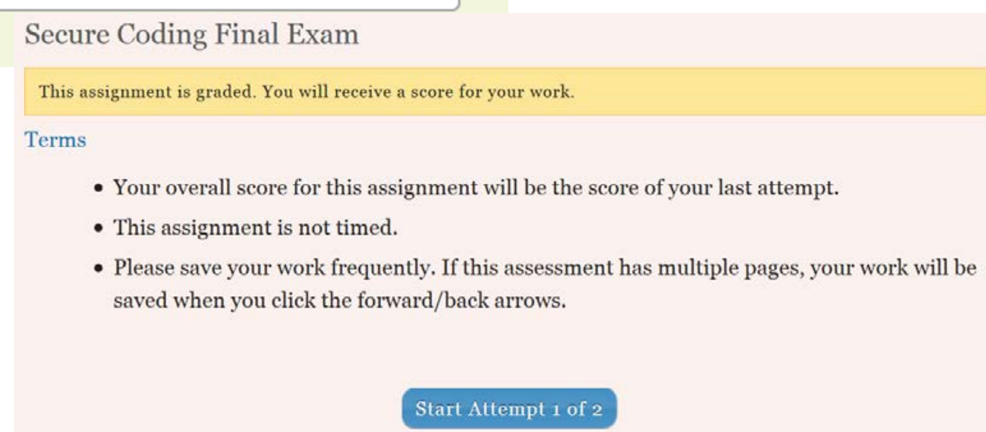


# Secure Coding Online Assessments



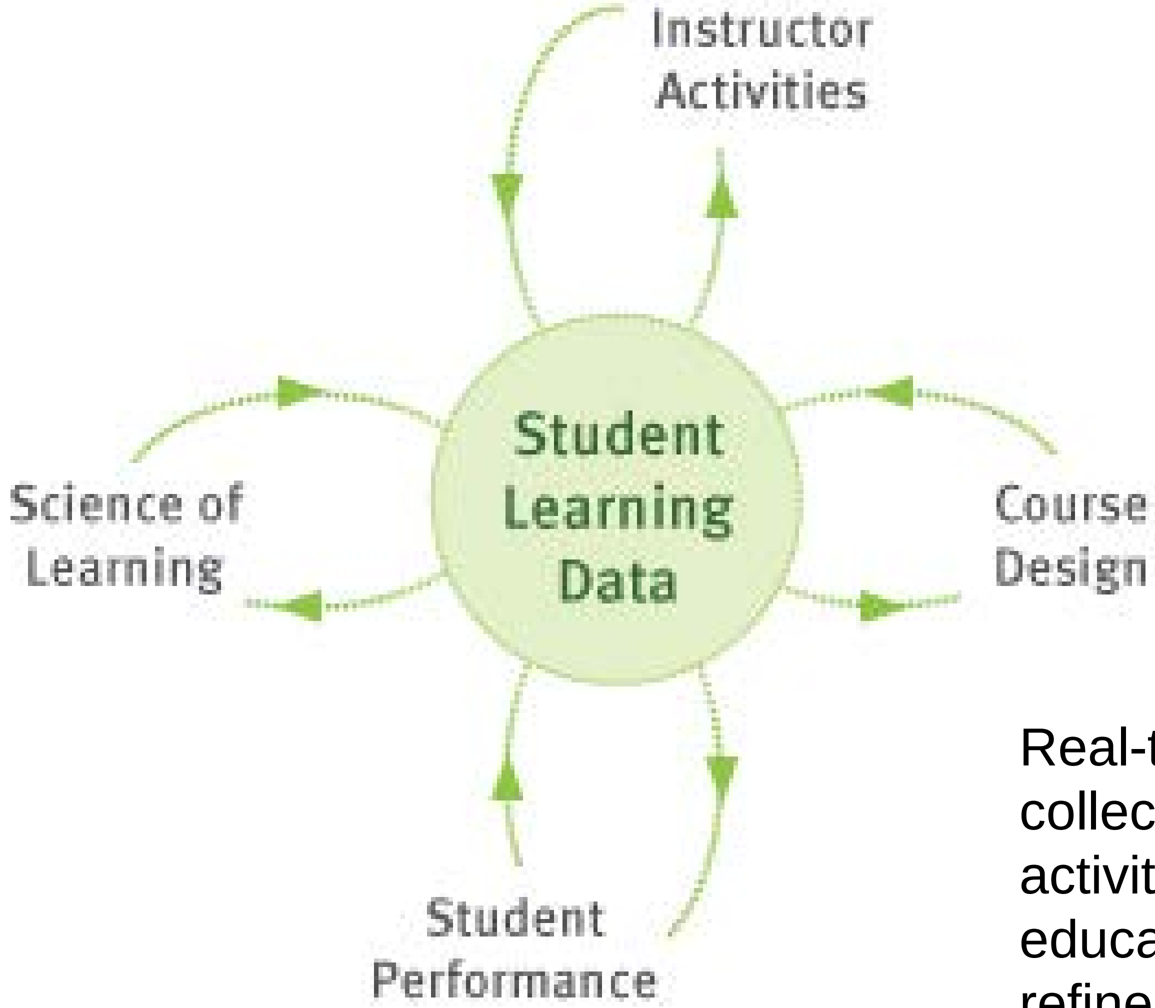

Learn by Doing and Did I Get This? activities reinforce information and help students check their progress.




Each module ends with a graded final exam.

# Feedback Loops




Real-time data collection of student activity enables educators to iteratively refine their courses

# Assessment

Objective assessment, such as multiple-choice questions

- provide a cost-effective means for determining examinee knowledge about areas such as language syntax
- much less successfully assess the ability of an examinee to create or modify working computer programs.

Performance-based assessment, examinees are examined for their ability to write software.

- assessments generally take the form of short answer examinations typically asking examinees to generate code fragments.

# Short Answer Examinations

Provide some degree of performance-based assessment, but have several shortcomings.

- Involve minimal tasks, such as creating a few lines of code or identifying some performance parameter.
- Cannot evaluate the ability to comprehend and build upon even a small class library.
- Typically performed without access to any programming tools, the examinees have no way to test or even compile their solutions.
- Must be graded manually, limiting the ability to offer the exam at a reasonable price and at a global scale.

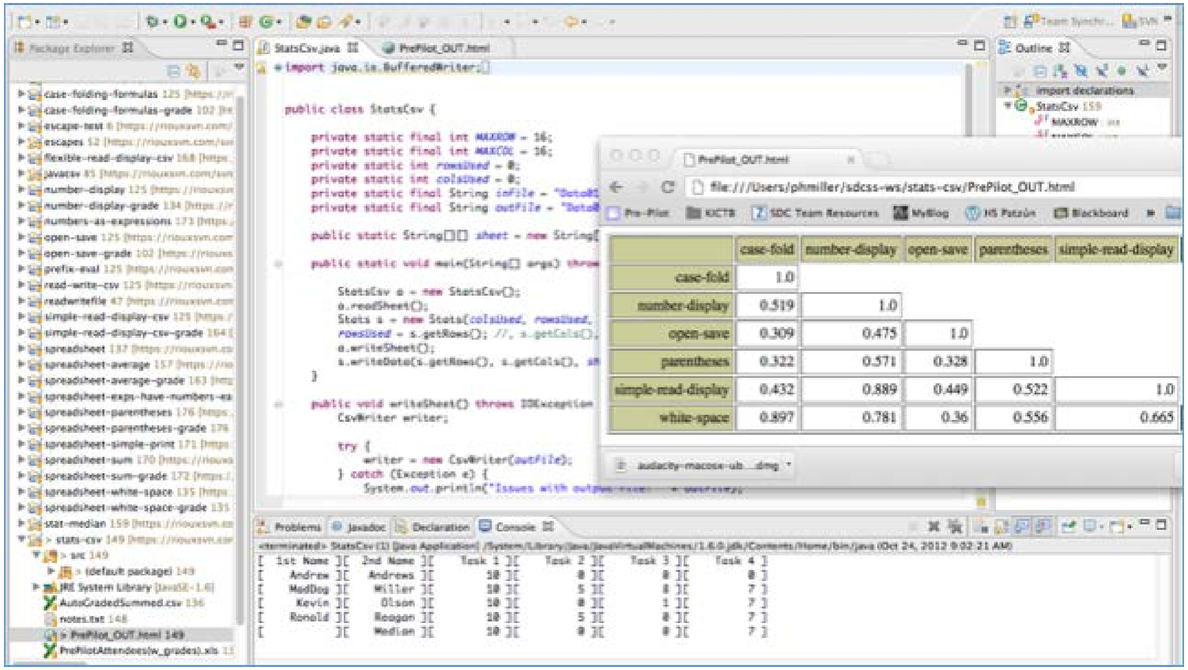
# Authentic Assessment

Create a testing environment that closely matches the working environment of software professionals and asking them to perform tasks typical of those performed software developers in similar roles.

The Software Developer Examination developed at CMU examines programmers by asking them to perform programming tasks using a normal development environment in a proctored setting and scoring their coding solutions.

# Authentic Assessment

Authentic assessment measures the test-takers' ability to program realistic problems in a professional programming environment.




The examinee is put in the role of a professional software developer and has an opportunity to demonstrate skills by building solutions to tasks defined in the context of real software projects.

# Agenda

Education and assessment of programmers in major software markets

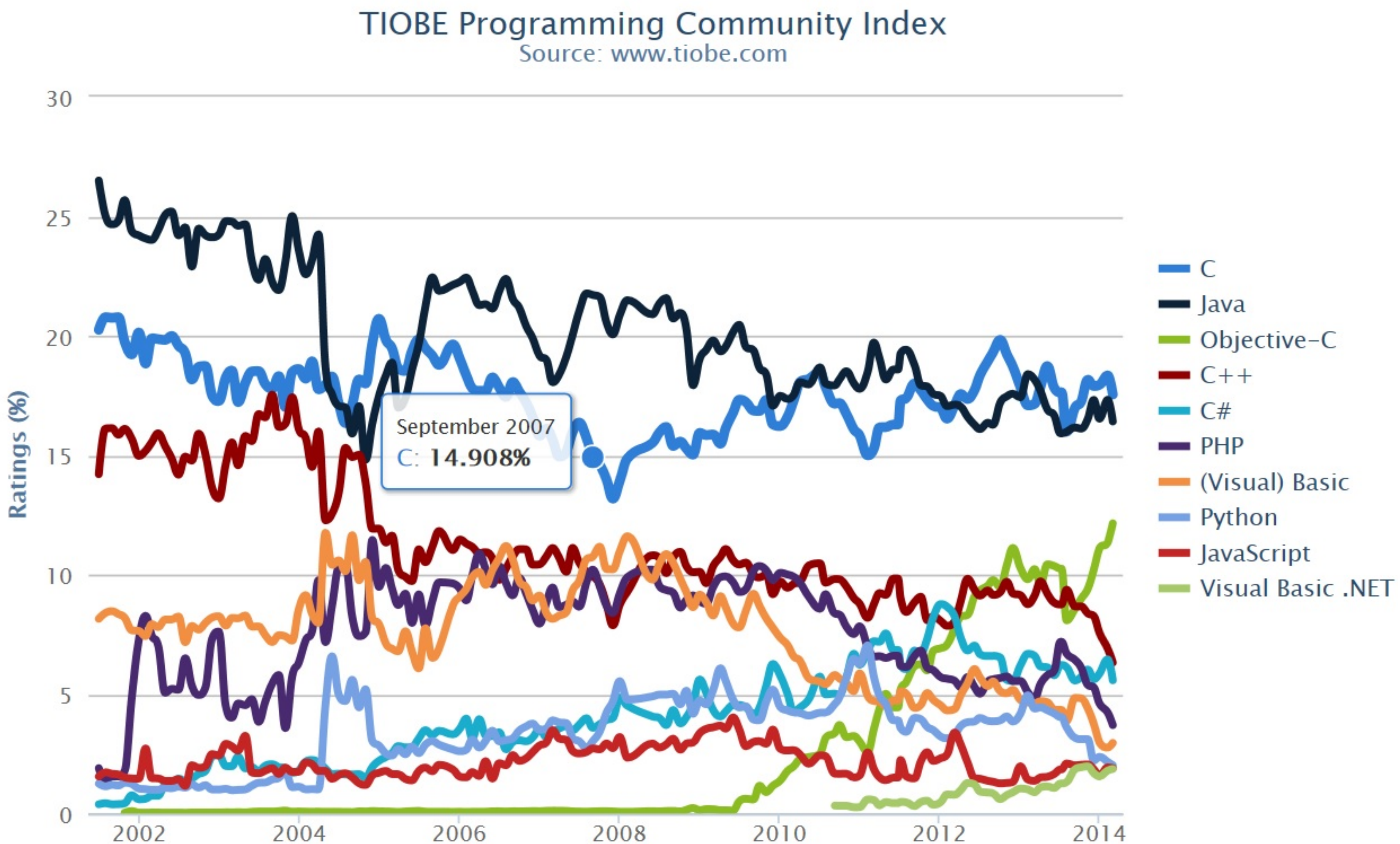
Programming is hard

Limitations of analysis and testing

Use and application of secure coding standards

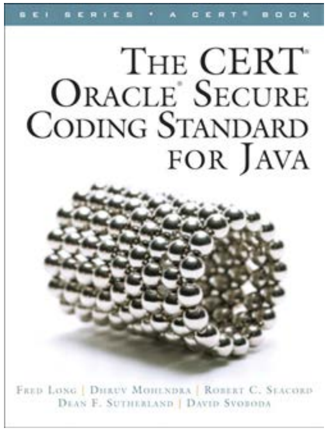
Conformance testing using SCALe (Source Code Analysis Laboratory)

# Popular Programming Languages



# Programming is Hard

Popular programming languages such as C (17.5%), Objective-C (12%), and C++ (6.3%) have undefined behaviors which do not need to be diagnosed and can result in errors and vulnerabilities.




I used to think Java was a “secure” language, then we wrote this book→ with 744 pages and 156 rules followed by this book with 304 pages and 75 additional recommendations →

# Undefined Behaviors

Undefined behaviors are identified in the C Standard:

- If a “shall” or “shall not” requirement is violated, and that requirement appears outside of a constraint, the behavior is undefined.
- Undefined behavior is otherwise indicated in this International Standard by the words “undefined behavior”
- by the omission of any explicit definition of behavior.

There is no difference in emphasis among these three; they all describe “behavior that is undefined”.

The C Standard Annex J.2, “Undefined behavior,” contains a list of explicit undefined behaviors in C.

# Undefined Behaviors

Behaviors are classified as "undefined" by standards committees to:

- give the implementer license not to catch certain program errors that are difficult to diagnose;
- avoid defining obscure corner cases which would favor one implementation strategy over another;
- identify areas of possible conforming language extension: the implementer may augment the language by providing a definition of the officially undefined behavior.

Implementations may

- ignore undefined behavior completely with unpredictable results
- behave in a documented manner characteristic of the environment (with or without issuing a diagnostic)
- terminate a translation or execution (with issuing a diagnostic).

# Code Example

```
char *copy(size_t n, const char *c_str) {
  if (n == 0) return NULL;
  char *p = (char *)malloc(n);
  if (p == NULL) return NULL;
  for (int i = 0; i < n; ++i) p[i] = *c_str++;
  return p;
}
```

The dynamically allocated buffer referenced by **`p`** overflows for values of **`n > INT_MAX`**

# Agenda

Education and assessment of programmers in major software markets

Programming is hard

Limitations of analysis and testing

Use and application of secure coding standards

Conformance testing using SCALe (Source Code Analysis Laboratory)

# Defect-removal Efficiency

The percentage of bugs eliminated by software reviews, inspections and tests. For example:

Total defect reports: 100

Development defects/total defects

= defect removal efficiency 90/100 = 0.9

Defect-removal efficiency: 90%

Jones, C., "Software defect-removal efficiency," *Computer* , vol.29, no.4, pp.94,95, Apr 1996
doi: 10.1109/2.488361

# Software Testing

Exhaustive testing (with all possible combinations of inputs or values for program variables) is impossible.

Some statistics:

- Most forms of testing are below 35% in defect removal efficiency or remove only about one bug out of three.
- All tests together seldom top 85% in defect removal efficiency.
- About 7% of bug repairs include new bugs.
- About 6% of test cases have bugs of their own.

Software testing can demonstrate the presence of bugs but cannot demonstrate their absence

- As we find problems and fix them, we raise our confidence that the software performs as it should
- But we can never guarantee that all bugs have been removed

# Formal Inspections

Formal inspections have been measured to top 85% in defect removal efficiency and have more than 40 years of empirical data from thousands of projects.

Inspections also raise testing defect removal efficiency by more than 5% for each major test stage.

# Static Analysis

A static analysis tool analyzes software without actually executing the software.

Many analyses which could be performed statically and would produce useful results are, unfortunately, NP-complete problems.

- the time required to solve the problem using any currently known algorithm increases quickly as the size of the problem grows.
- the time required to solve even moderately sized versions of many of these problems can easily reach into the billions or trillions of years, using any amount of computing power available today.

# Static Analysis

NP-complete problems are often addressed by using heuristic methods and approximation algorithms.

- static race detection tools provide an approximate identification.
- static analysis algorithms are prone to false negatives (vulnerabilities not identified) and false positives (incorrectly identified vulnerabilities).

Static analysis has a high defect removal efficiency, frequently topping 65%.

# Dynamic Analysis

Dynamic analysis tools integrates detection with the actual program's execution.

The advantage of this approach is that a real runtime environment is available to the tool.

Analyzing only the actual execution flow has the additional benefit of producing fewer false positives that the programmer must consider.

The main disadvantages of dynamic detection are

- fails to consider execution paths not taken
- significant runtime overhead associated with dynamic detection

# Why Can’t Johnny Program Securely?

> Inefficient programmers tend to experiment randomly until they find a combination that seems to work.
>
> — Steve McConnell, Code Complete

“Inefficient”, “inexperienced”, “under-educated”, etc.


Random experimentation will eventually produce code that works under optimal (tested) conditions but will not produce secure code.

# Why Can’t Suzie Program Securely?

Women's share in computer occupations declined to 27% in 2011 after reaching a high of 34% in 1990[1].

The notion of a gender divide in how men and women relate to computing is largely a result of cultural and environmental conditions[2].

The reasons for women entering, not entering, or not staying in the field of computer science have a lot to do with

- environment
- culture
- perception of the field

1) Census Bureau Reports Women's Employment in Science, Tech, Engineering and Math Jobs Slowing as Their Share of Computer Employment Falls.

2) Carol Frieze, Orit Hazzan, Lenore Blum, and M. Bernardine Dias. 2006. Culture and environment as determinants of women's participation in computing: revealing the "women-CS fit". *SIGCSE Bull.* 38, 1 (March 2006), 22-26. DOI=10.1145/1124706.1121351 http://doi.acm.org/10.1145/1124706.1121351

# Agenda

Education and assessment of programmers in major software markets

Programming is hard

Limitations of analysis and testing

Use and application of secure coding standards

Conformance testing using SCALe (Source Code Analysis Laboratory)

# CERT Secure Coding Standards

CERT C Secure Coding Standard

- Version 1.0 (C99) published in 2009
- Version 2.0 (C11) published in 2011
- ISO/IEC TS 17961 C Secure Coding Rules Technical Specification
- Conformance Test Suite

CERT C++ Secure Coding Standard

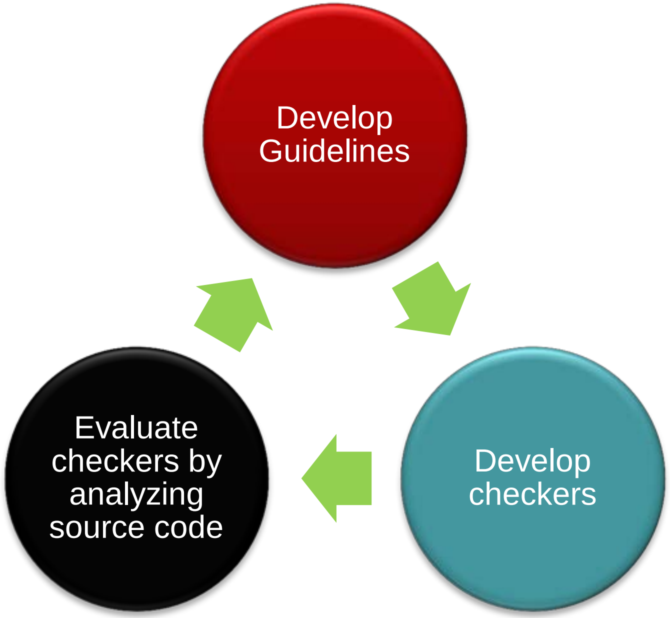
- Not completed/not funded

CERT Oracle Secure Coding Standard for Java

- Version 1.0 (Java 7) published in 2011
- Java Secure Coding Guidelines
- Identified Java rules applicable to Android development
- Planned: Android-specific version designed for the Android SDK

The CERT Perl Secure Coding Standard

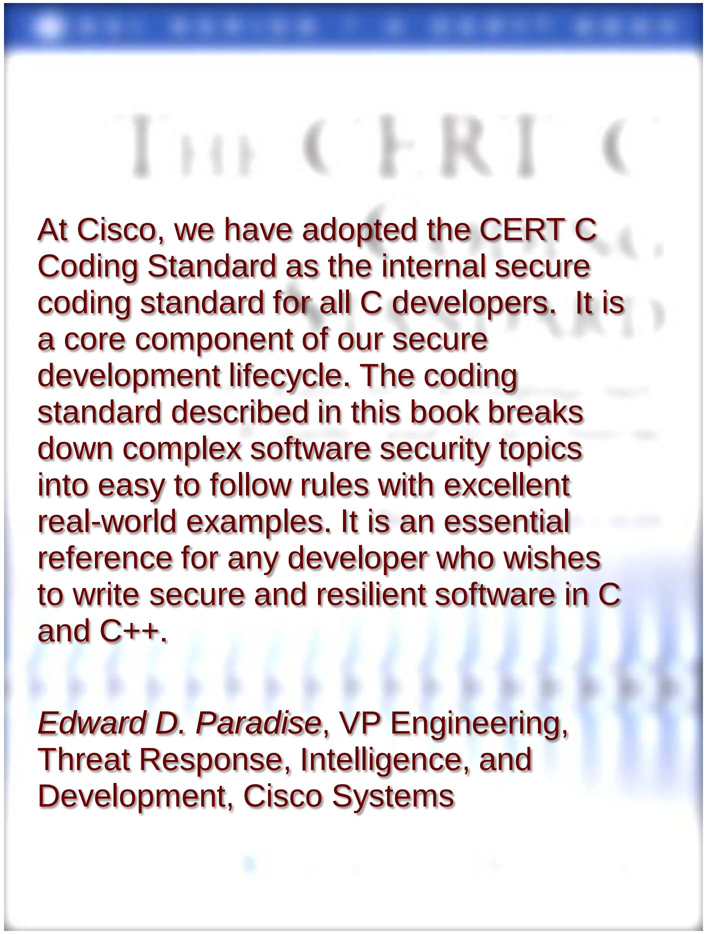
- Version 1.0 under development



# The CERT C Coding Standard

## Standards

- ISO/IEC TS 17961 C Secure Coding Rules establishes a baseline set of requirements for static analysis tools and C language compilers.
- The CERT C Coding Standard was updated for C11 and compatibility with ISO/IEC TS 17961.

At Cisco, we have adopted the CERT C Coding Standard as the internal secure coding standard for all C developers. It is a core component of our secure development lifecycle. The coding standard described in this book breaks down complex software security topics into easy to follow rules with excellent real-world examples. It is an essential reference for any developer who wishes to write secure and resilient software in C and C++.

*Edward D. Paradise*, VP Engineering, Threat Response, Intelligence, and Development, Cisco Systems

# Rules and Recommendations

Rules and recommendations in the secure coding standards include

- Concise but not necessarily precise title
- Precise definition of the rule
- Noncompliant code examples or antipatterns in a pink frame—do not copy and paste into your code
- Compliant solutions in a blue frame that conform with all rules and can be reused in your code
- Risk Assessment

# Risk Assessment

Risk assessment is performed using failure mode, effects, and criticality analysis.

**Severity**—How serious are the consequences of the rule being ignored?

| Value | Meaning | Examples of Vulnerability |
|---|---|---|
| 1 | low | denial-of-service attack, abnormal termination |
| 2 | medium | data integrity violation, unintentional information disclosure |
| 3 | high | run arbitrary code |

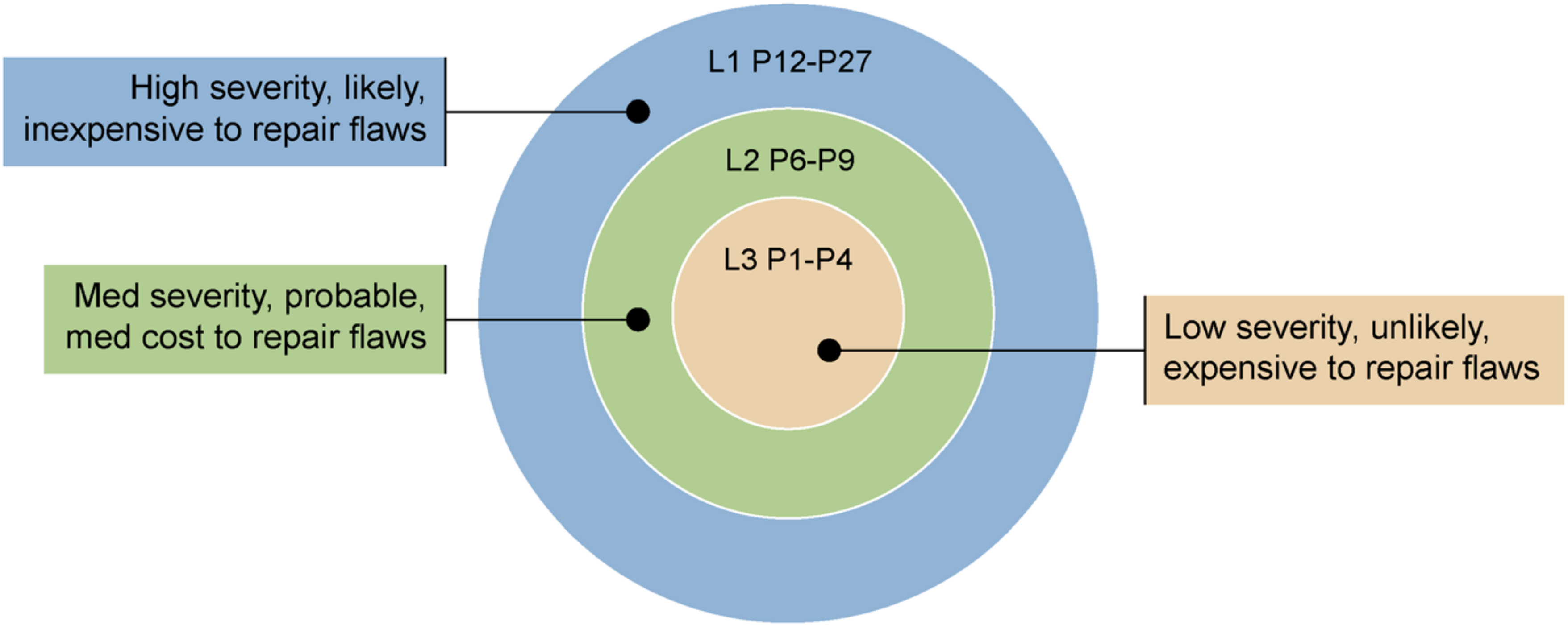
**Likelihood**—How likely is it that a flaw introduced by ignoring the rule can lead to an exploitable vulnerability?

| Value | Meaning |
|---|---|
| 1 | unlikely |
| 2 | probable |
| 3 | likely |

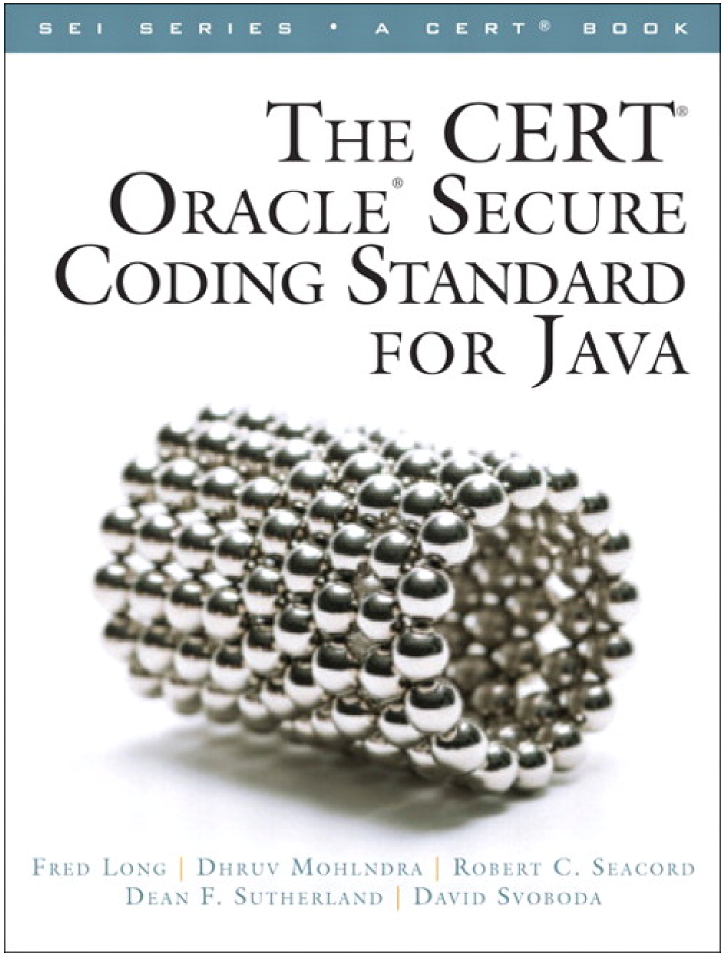
**Cost**—The cost of mitigating the vulnerability.

| Value | Meaning | Detection | Correction |
|---|---|---|---|
| 1 | high | manual | manual |
| 2 | medium | automatic | manual |
| 3 | low | automatic | automatic |

# Priorities and Levels



# Secure Coding Standard for Java




"In the Java world, security is not viewed as an add-on a feature. It is a pervasive way of thinking. Those who forget to think in a secure mindset end up in trouble. But just because the facilities are there doesn't mean that security is assured automatically. A set of standard practices has evolved over the years. ***The Secure® Coding® Standard for Java™*** is a compendium of these practices. These are not theoretical research papers or product marketing blurbs. This is all serious, mission-critical, battle-tested, enterprise-scale stuff."

—**James A. Gosling,** Father of the Java Programming Language

# Scope

The *CERT® Oracle® Secure Coding Standard for Java™* focuses on the Java Standard Edition 6 (Java SE 6) Platform environment and includes rules for secure coding using the Java programming language and libraries.

*The Java Language Specification,* third edition [JLS 2005], prescribes the behavior of the Java programming language and served as the primary reference for the development of this standard.

This coding standard also addresses new features of the Java SE 7 Platform, primarily as alternative compliant solutions to secure coding problems that exist in both the Java SE 6 and Java SE 7 platforms.

# CERT Perl Secure Coding Standard

Provides a core of well-documented and enforceable coding rules and recommendations for Perl

Developed specifically for versions 5.12 and later of the Perl programming language

Contains just over 30 guidelines in eight sections:

- Input Validation and Data Sanitization
- Declarations and Initialization
- Expressions
- Integers
- Strings
- Object-Oriented Programming (OOP)
- File Input and Output
- Miscellaneous

# Agenda

Education and assessment of programmers in major software markets

Undefined behaviors in popular programming languages

Limitations of analysis and testing

Use and application of secure coding standards

Conformance testing using SCALe (Source Code Analysis Laboratory)

# Source Code Analysis Laboratory

Source Code Analysis Laboratory (SCALe)

- Consists of commercial, open source, and experimental analysis
- Is used to analyze various code bases including those from the DoD, energy delivery systems, medical devices, and more
- Provides value to the customer but is also being instrumented to research the effectiveness of coding rules and analysis

SCALe customer-focused process:

1. Customer submits source code to CERT for analysis.
2. Source is analyzed in SCALe using various analyzers.
3. Results are analyzed, validated, and summarized.
4. Detailed report of findings is provided to guide repairs.
5. The developer addresses violations and resubmits repaired code.
6. The code is reassessed to ensure all violations have been properly mitigated.
7. The certification for the product version is published in a registry of certified systems.

# Government Demand

SEC. 933 of the National Defense Authorization Act for Fiscal Year 2013 requires evidence that government software development and maintenance organizations and contractors are conforming in computer software coding to approved secure coding standards of the Department during software development, upgrade, and maintenance activities, including through the use of inspection and appraisals.

The Application Security and Development Security Technical Implementation Guide (STIG)

- is being specified in the DoD acquisition programs' Request for Proposals (RFPs).
- provides security guidance for use throughout an application's development lifecycle.

Section 2.1.5, "Coding Standards," of the Application Security and Development STIG identifies the following requirement:

*(APP2060.1: CAT II) "The Program Manager will ensure the development team follows a set of coding standards."*

# Industry Demand

Conformance with CERT secure coding standards can represent a significant investment by a software developer, particularly when it is necessary to refactor or modernize existing software systems.

However, it is not always possible for a software developer to benefit from this investment, because it is not always easy to market code quality.

A goal of conformance testing is to provide an incentive for industry to invest in developing conforming systems:

- Perform conformance testing against CERT secure coding standards.
- Verify that a software system conforms with a CERT secure coding standard.
- Use CERT seal when marketing products.
- Maintain a certificate registry with the certificates of conforming systems.

# CERT SCALe Seal 1

Developers of software that has been determined by CERT to conform to a secure coding standard may use the CERT SCALe seal to describe the conforming software on the developer’s website.

The seal must be specifically tied to the software passing conformance testing and not applied to untested products, the company, or the organization.

Use of the CERT SCALe seal is contingent upon the organization entering into a service agreement with Carnegie Mellon University and upon the software being designated by CERT as conforming.

# CERT SCALe Seal 2

Except for patches that meet the following criteria, any modification of software after it is designated as conforming voids the conformance designation. Until such software is retested and determined to be conforming, the new software cannot be associated with the CERT SCALe seal.

Patches that meet all three of the following criteria do not void the conformance designation:

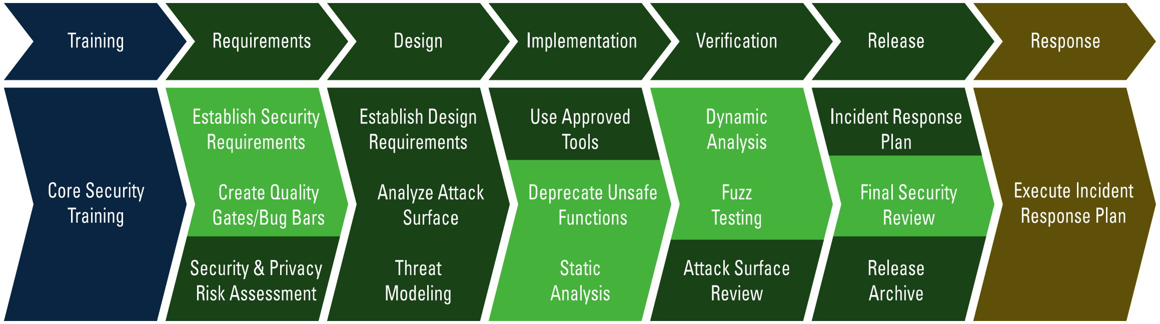
- The patch is necessary to fix a vulnerability in the code or is necessary for the maintenance of the software.
- The patch does not introduce new features or functionality.
- The patch does not introduce a violation of any of the rules in the secure coding standard to which the software has been determined to conform.

# Source Code Analysis Laboratory

Microsoft Simplified Security Development Lifecycle has been instantiated using CERT tools and methods*.

SCALe supports the following SDL Security Activities:

- *Establish Security Requirements*
- *Create Quality Gates/Bug Bars*
- *Static Analysis*
- *Dynamic Analysis*
- *Fuzz Testing*
- *Final Security Review*




* See www.cert.org/archive/pdf/MS_CERT_SDL.pdf

# For More Information

**Visit CERT® websites:**

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