Secure Coding Standards

2008 February 5th

Robert C. Seacord

Agenda

Software Security

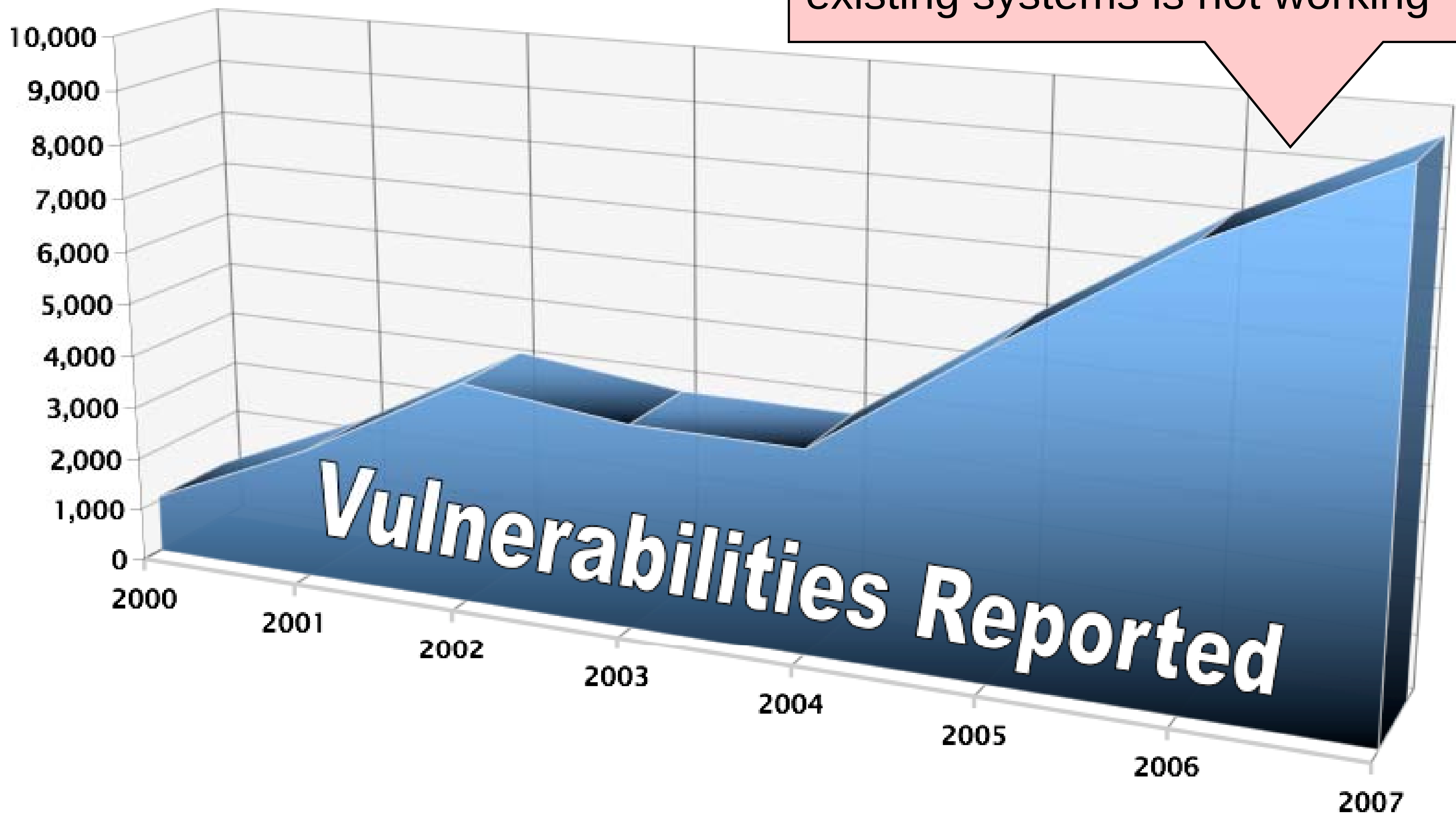
CERT Secure Coding Initiative

CERT Secure Coding Standards

Future Directions and Key Points

Increasing Vulnerabilities

Reacting to vulnerabilities in existing systems is not working



Application Security

Most Vulnerabilities caused by Programming Errors

According to a study of vulnerabilities in the National Vulnerabilty Database (NVD) in 2004 by Heffley/Meunier:

- 64% of the vulnerabilities are due to programming errors
- 36% of vulnerabilities are due to configuration or design problems.

Most of these programming errors are repeated basic mistakes.

- 20% buffer overflows
- 11% directory traversal attacks
- 9% format string vulnerabilities
- 4% symlink attacks
- 4% cross-site scripting vulnerabilities
- 3% shell metacharacter

Agenda

Software Security

CERT Secure Coding Initiative

CERT Secure Coding Standards

Future Directions and Key Points

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

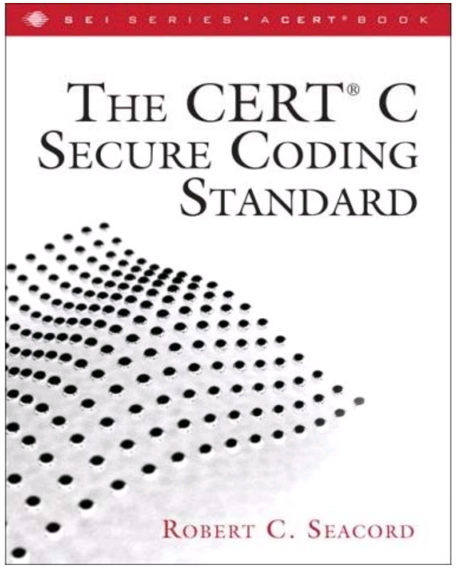
Secure coding standards

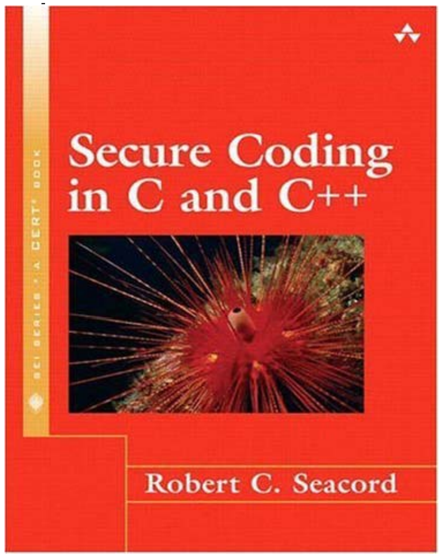
www.securecoding.cert.org

Source code analysis and certification

Training courses

Involved in international standards development.





EXPLORE | CREATE | APPLY | AMPLIFY | SUSTAIN

CERT C Secure Coding Standard

- Open Group Adoption
- WG23 Language Annex

- Wiki content

- Software Certification
- Tool Certification

CERT C++ Secure Coding Standard

The CERT Sun Microsystems Secure Coding Standard for Java

C Language Security Annex and Prototype

C Language Security Annex and Prototype

C language standard is undergoing a major revision “C1X”

- Originally about security, emphasis has shifted 180 degrees towards support for multithreading and performance optimizations that may actually hurt security.

This work would develop a

- An informative “Analyzability” annex to the C standard that defines a security “profile”
  - some enhancements would be implemented by simply recompiling
  - others may require source code modification
- a prototype implementation using gcc

Annex could be implemented by compiler vendors who

- want to provide a secure implementation
- do not have to compete head to head with other compilers only concerned with performance.

C Language Security Annex and Prototype

This project could have a greater positive impact on software security than anything I can imagine given that

- success of the C security annex would motivate C++ vendors to provide the same semantics for C++
- C and C++ account for ~26% of the market (according to the TIOBE index)
- ~65% of vulnerabilities in the US-CERT vulnerability database involve these languages
- standardization will result in broad adoption by compiler vendors

Agenda

Software Security

CERT Secure Coding Initiative

CERT Secure Coding Standards

Future Directions and Key Points

CERT Secure Coding Standards

Identify coding practices that can be used to improve the security of software systems under development

Coding practices are classified as either rules or recommendations

- Rules need to be followed to claim compliance.
- Recommendations are guidelines or suggestions.

Development of Secure Coding Standards is a community effort

Scope

The secure coding standards proposed by CERT are 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)

Applicable technical corrigenda and documented language extensions such as the ISO/IEC TR 24731 extensions to the C library are also included.

Secure Coding Wiki


www.securecoding.cert.org

CERT

Software Assurance | Secure Systems | Organizational Security | Coordinated Response

Dashboard > Secure Coding > CERT Secure Coding Standards   Search

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

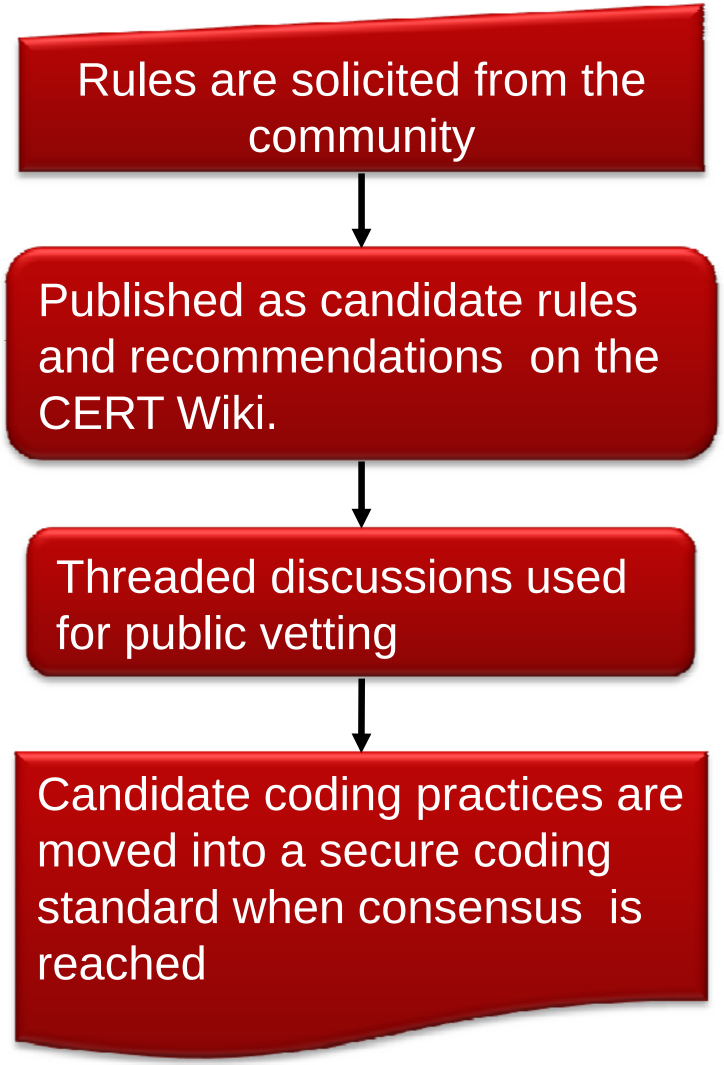
Added by Confluence Administrator , last edited by Robert Seacord on Sep 08, 2008 (view change)

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

Coding practices are defined as rules when

1. Violation of the coding practice is likely to result in a security flaw that may result in an exploitable vulnerability.
2. There is a denumerable set of conditions for which violating the coding practice is necessary to ensure correct behavior.
3. Conformance to the coding practice can be determined through automated analysis, formal methods, or manual inspection techniques.

Recommendations

Coding practices are defined as recommendations when

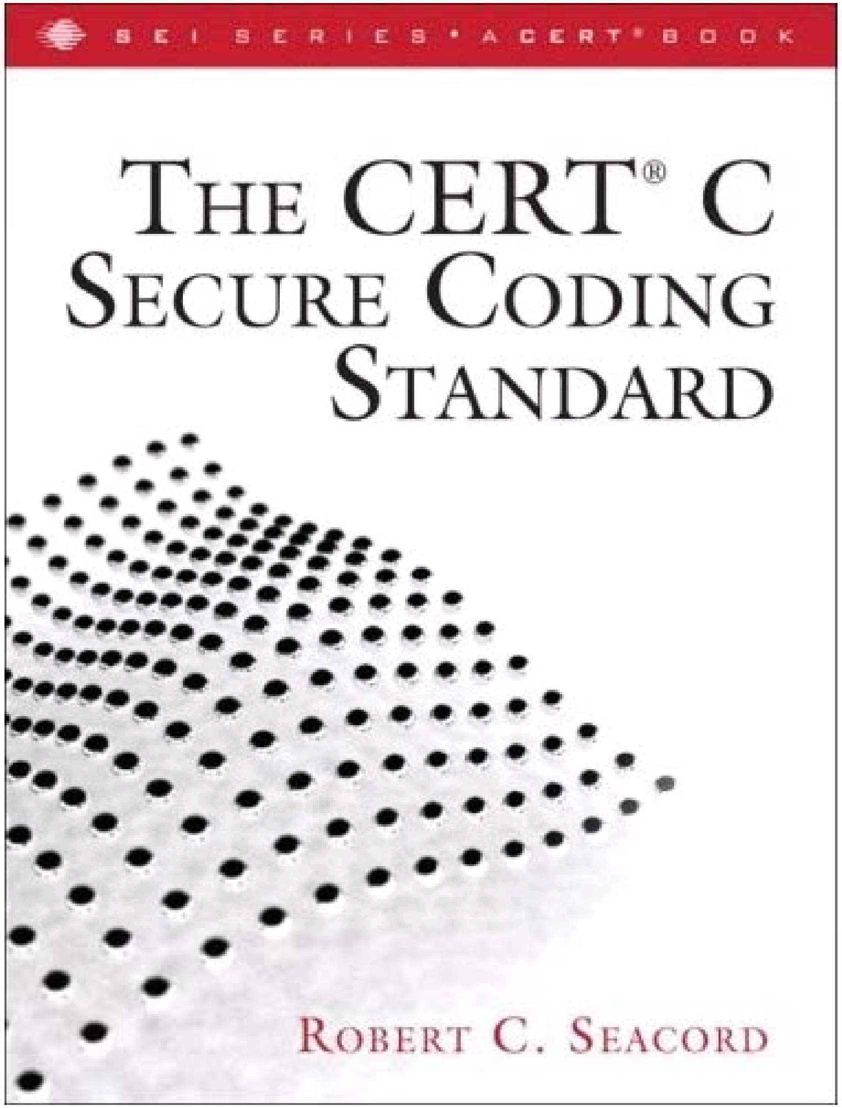
1. Application of the coding practice is likely to improve system security.
2. One or more of the requirements necessary for a coding practice to be considered a rule cannot be met.

The CERT C Secure Coding Standard

Developed with community involvement, including over 320 registered participants on the wiki.

Version 1.0 published by Addison-Wesley in September, 2008.

- 134 Recommendations
- 89 Rules



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.

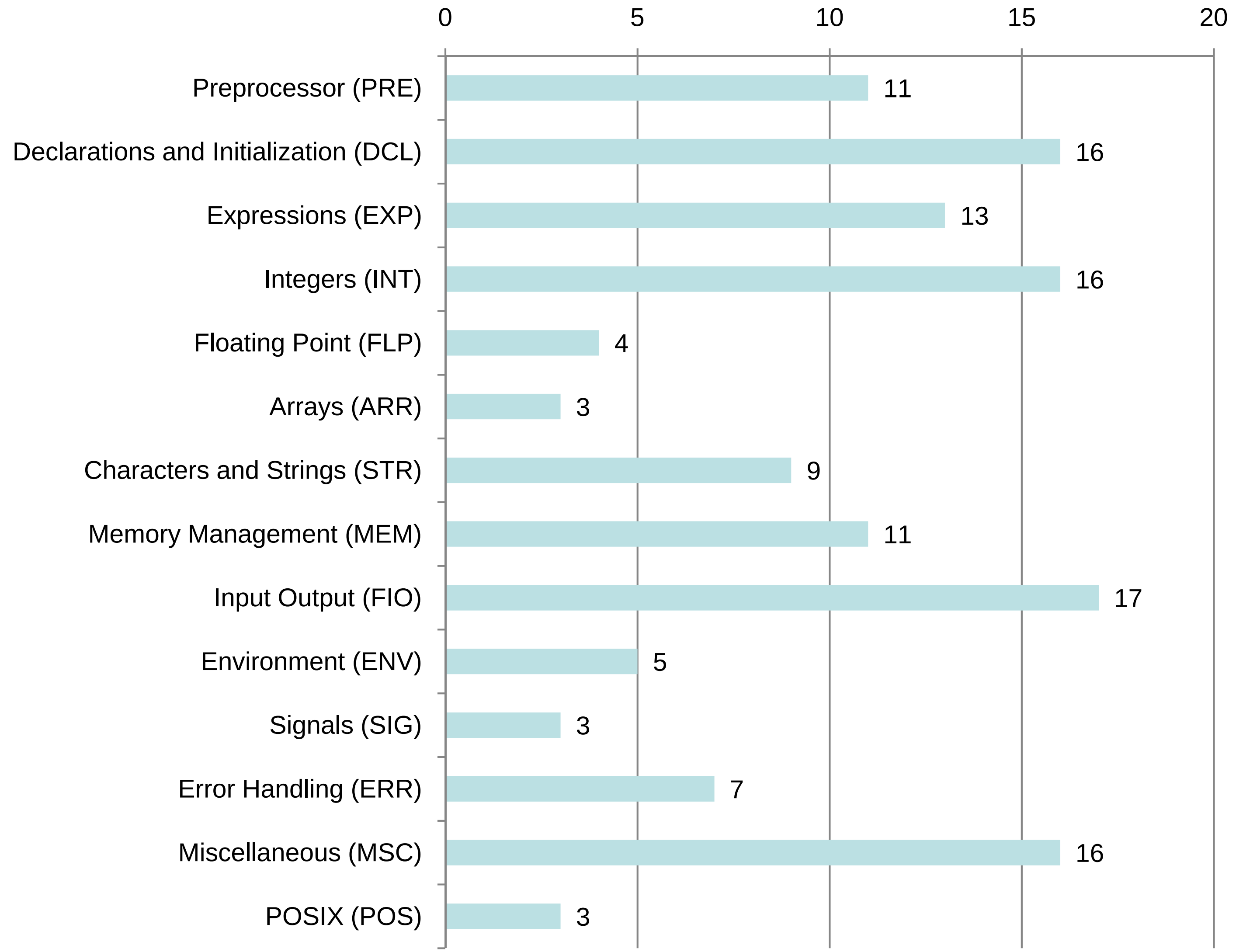
```
char *p = "string literal"; p[0] = 'S';
```

Compliant Solution

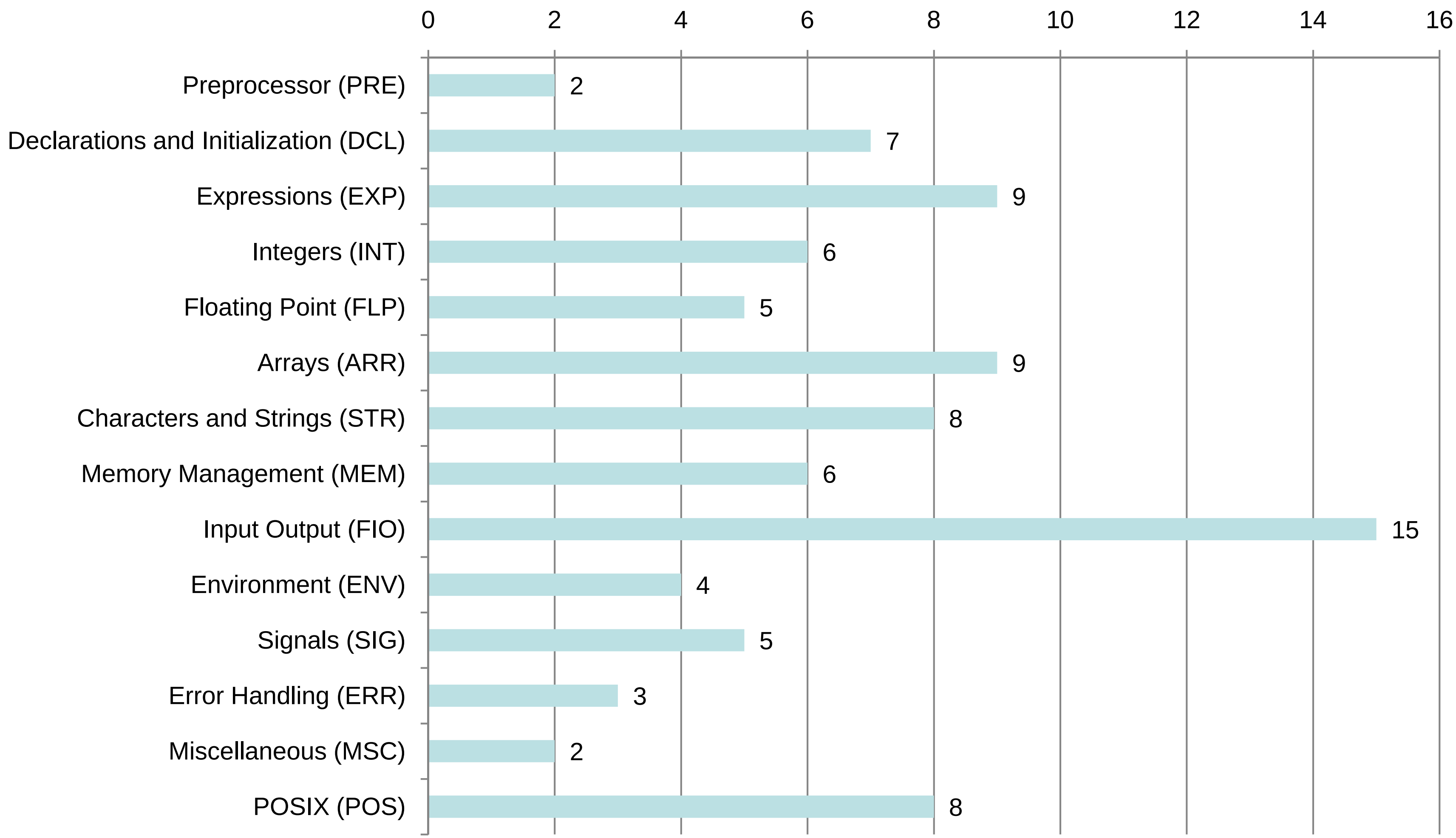
As an array initializer, a string literal specifies the initial values of characters in an array as well as the size of the array. This code creates a copy of the string literal in the space allocated to the character array **`a`**. The string stored in a can be safely modified.

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

Distribution of C Recommendations



Distribution of C Rules



POSIX

Many of the core guidelines demonstrate compliant solutions that rely for POSIX-compliant systems.

The CERT C Secure Coding Standard also contains an appendix with guidelines (3 recommendations and 8 rules) for using functions that are defined as part of the POSIX family of standards but are not included in ISO/IEC 9899-1999.

These rules and recommendations are not part of the core standard because they do not apply in all C language applications and because they represent an incomplete set.

The intent of providing these guidelines is to demonstrate how rules and recommendations for other standards or specific implementations may be integrated with the core C99 recommendations.

Contributors and Reviewers

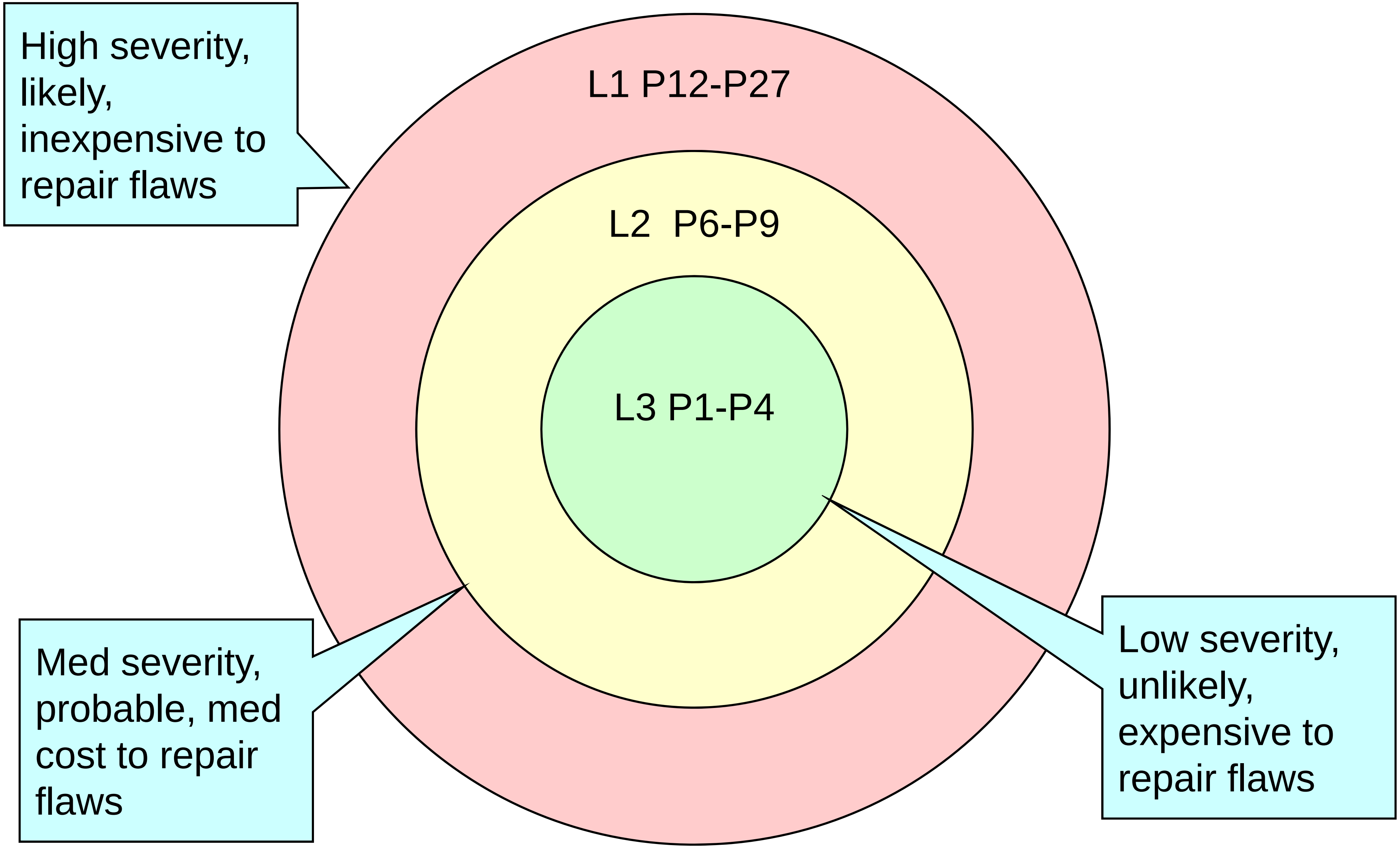
Contributors

Arbob Ahmad, Juan Alvarado, Dave Aronson, Abhishek Arya, BJ Bayha, Levi Broderick, Hal Burch, Steven Christey, Ciera Christopher, Geoff Clare, Joe Damato, Stephen C. Dewhurst, Susan Ditmore, Chad Dougherty, Mark Dowd, Xiaoyi Fei, William Fithen, Hallvard Furuseth, Jeffrey Gennari, Douglas A. Gwyn, Shaun Hedrick, Christina Johns, David Keaton, Takuya Kondo, Masaki Kubo, Richard Lane, Stephanie Wan-Ruey Lee, Jonathan Leffler, Fred Long, Gregory K. Look, Nat Lyle, Larry Maccherone, John McDonald, Dhruv Mohindra, Bhaswanth Nalabothula, Justin Pincar, Randy Meyers, David M. Pickett, Thomas Plum, Dan Saks, Robert C. Seacord, David Svoboda, Chris Taschner, Ben Tucker, Fred J. Tydeman, Nick Stoughton, Wietse Venema, Alex Volkovitsky, Grant Watters, and Gary Yuan.

Reviewers

Kevin Bagust, Greg Beeley, Arjun Bijanki, John Bode, Stewart Brodie, G Bulmer, Kyle Comer, Sean Connelly, Ale Contenti, Tom Danielsen, Török Edwin, Brian Ewins, Justin Ferguson, Stephen Friedl, Samium Gromoff, Kowsik Guruswamy, Peter Gutmann, Richard Heathfield, Darryl Hill, Paul Hsieh, Ivan Jager, Steven G. Johnson, Anders Kaseorg, Jerry Leichter, Nicholas Marriott, Scott Meyers, Eric Miller, Ron Natalie, Heikki Orsila, Dan Plakosh, P.J. Plauger, Michel Schinz, Eric Sosman, Chris Tapp, Andrey Tarasevich, Josh Triplett, Pavel Vasilyev, Ivan Vecerina, Zeljko Vrba, David Wagner, Henry S. Warren, Colin Watson, Zhenyu Wu, Drew Yao, and Christopher Yeleighton.

Priorities and Levels



FIO30-C. Exclude user input from format strings

Risk Assessment

Failing to exclude user input from format specifiers may allow an attacker to execute arbitrary code.

Rule	Severity	Likelihood	Remediation Cost	Priority	Level
FIO30-C	**3** (high)	**3** (probable)	**3** (low)	**P27**	**L1**

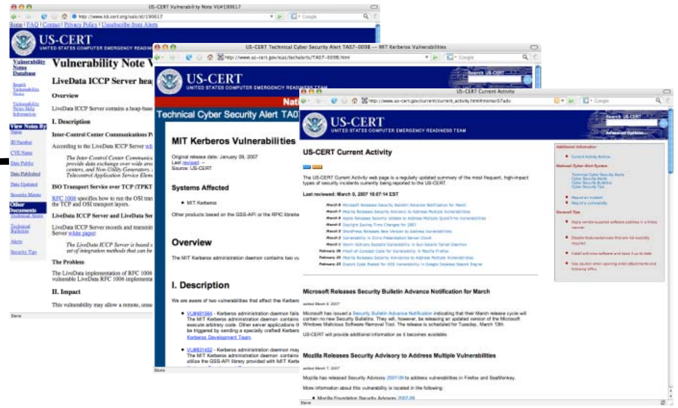
Two recent examples of format string vulnerabilities resulting from a violation of this rule include Ettercap and Samba. In Ettercap v.NG-0.7.2, the ncurses user interface suffers from a format string defect. The `curses_msg()` function in `ec_curses.c` calls `wdg_scroll_print()`, which takes a format string and its parameters and passes it to `vw_printw()`. The `curses_msg()` function uses one of its parameters as the format string. This input can include user-data, allowing for a format string vulnerability [VU#286468]. The Samba AFS ACL mapping VFS plug-in fails to properly sanitize user-controlled filenames that are used in a format specifier supplied to `snprintf()`. This security flaw becomes exploitable when a user is able to write to a share that uses Samba's `afsacl.so` library for setting Windows NT access control lists on files residing on an AFS file system.

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

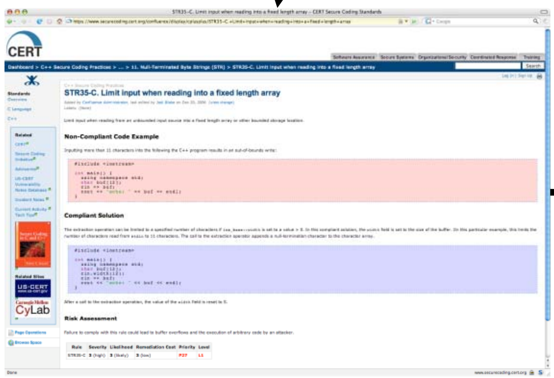
CERT Mitigation Information

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 Technical Alerts



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

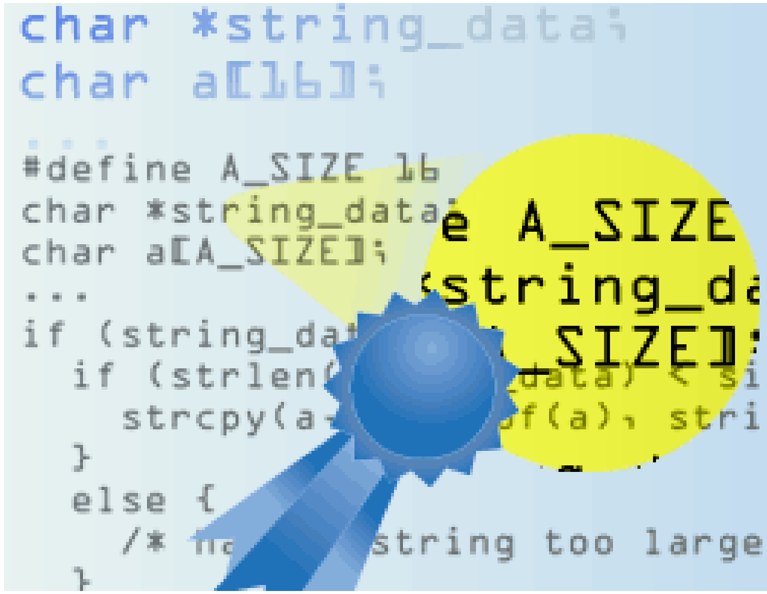
- may be extended with organization-specific rules
- cannot replace or remove existing rules

Train software professionals

Certify programmers in secure coding

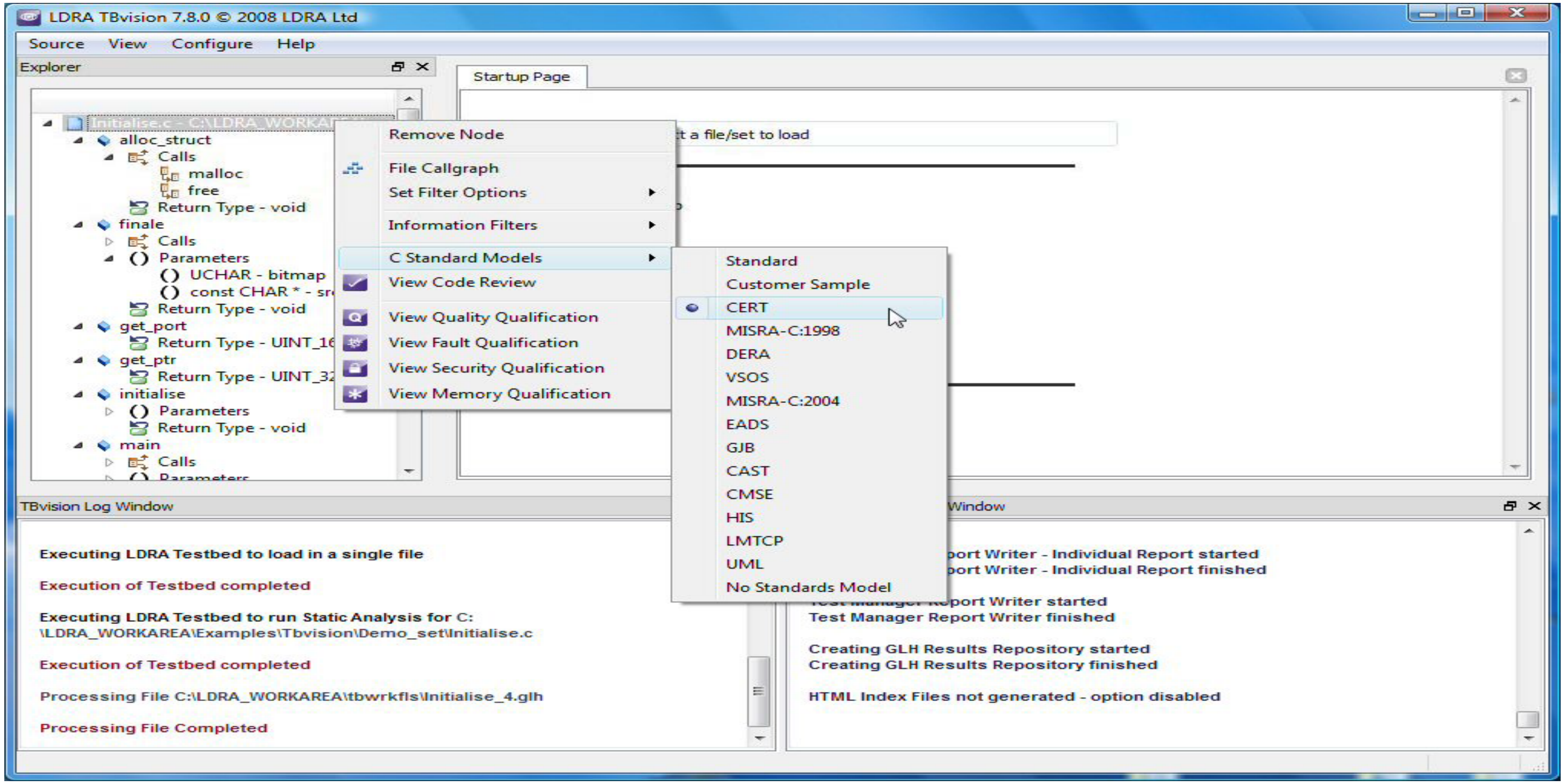
Establish requirements for software analysis tools

Software Certification



Major Software Vendor LDRA Adopts CERT C Secure Coding Standard

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

LDRA Press Release

Boston, MA – October 26, 2008. LDRA (Booth 1017), provider of the most complete automated software verification, source code analysis and test tools covering the full development lifecycle, has released its new TBsecure plug-in complete with the Carnegie Mellon Software Engineering Institute (SEI) CERT C secure coding standard.

TBsecure identifies security vulnerabilities and enables implementation of the just released CERT C Secure Coding Standard version 1.0.

Software Validation & Verification

Implementing checkers for various software analysis tools to verify compliance with CERT secure coding standards

- LDRA
- Fortify SCA
- Lawrence Livermore National Laboratory (LLNL) Compass / ROSE
- Coverity Prevent

CMU/SEI-2008-TR-014

"Evaluation of CERT Secure Coding Rules through Integration with Source Code Analysis Tools"

Study to evaluate the effectiveness of secure coding practices, including the use of static analysis tools coupled with secure coding rule sets

- the CERT C Programming Language Secure Coding Standard
- CERT C++ Programming Language Secure Coding Standard

This study was a joint effort between the CERT Secure Coding Initiative and JPCERT/CC.

The objectives of the study were to evaluate the efficacy of the CERT Secure Coding Standards and source code analysis tools in improving the quality and security of commercial software projects.

Study Design

Two static analysis tools were selected for their extensibility as well as overall effectiveness:

- Fortify Source Code Analysis (SCA) from Fortify Software and
- Compass/ROSE from Lawrence Livermore National Laboratory.

Checkers were developed for each tool to check code for violations of the CERT C and C++ Secure Coding Standards.

These tools were provided to Software Research Associates, Inc. (SRA), a Japanese software development firm.

SRA evaluated the extended versions of Fortify SCA and Compass/ROSE on two existing projects:

- a toll collection system-related GUI application written in C++
- a Video Service Communication Protocol written in the C programming language.

Study Conclusions

The project successfully extended source code analysis tools to discover a number of software defects in both projects evaluated, demonstrating the effectiveness of both the CERT Secure Coding Standards and the static analysis tools evaluated in improving software quality.

The project was also successful in identifying ways in which both the CERT Secure Coding Standards and the static analysis tools could be further improved.

CERT SCALe (Source Code Analysis Lab)

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

Enable buyers and developers of software to ensure that software is correct, secure, and fault resistant, even when source code and design information is not fully available.

Secure Coding in C/C++ Training

Secure Coding in C and C++ 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

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

Software Assurance Education

CMU CS 15392 Secure Programming offered as an undergraduate elective in the School of Computer Science in S07, S08, S09

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

CMU INI Graduate Class in Secure Software Engineering14735

Courses based on this material currently being offered at several universities

Software Assurance Education

The SEI is organizing a small group of universities around the theme of secure coding.

The SEI to host a workshop that brings professors together to design parallel efforts to

1. promulgate secure coding to their students and
2. measure the impact of #1 on student abilities to develop software that is not vulnerable to known attacks.

Agenda

Software Security

CERT Secure Coding Initiative

CERT Secure Coding Standards

Future Directions and Key Points

CERT C and ISO/IEC WG14

The idea for a CERT C Secure Coding standard arose at the ISO/IEC WG14 (the international standardization working group for the programming language C) meeting in Berlin in March 2006.

The CERT C guideline has been twice reviewed by WG14, at the London and Kona meetings.

During the Delft 2008 meeting, PL22.11 discussed if it should submit the CERT Secure Coding Standard to WG14 as a candidate for publication as a Type 2 or Type 3 technical report.

The next revision of the C Secure Coding Guideline is being prepared in proper format for approval as an ISO C Technical Report and should be available by the end of this month.

ISO/IEC C Secure Coding Guideline

Goal is to publish as a Type II technical report.

Target audience would include source code analysis tool vendors.

The secure coding guidelines would focus on rules and “analyzable” recommendations.

The CERT C Secure Coding Standard is being used as a base document.

Cut down to eliminate non-normative text such as

- compliant solutions
- risk analysis

Publication as an Open Standard

Publishing secure coding guidelines as an open standard will allow the content to be controlled through an open standards process.

Particularly important In the case where government or other industry bodies begin to require conformance with this document.

- The corollary is that not publishing as an open standard may not prevent the US government or other software consumers from requiring conformance to the existing document.
- For example, procurement language under development by the State of New York and other state governments already is being adjusted to use the CWE/SANS TOP 25 Programming Errors which is mapped to CERT Secure Coding Standards.

Roadmap

Create	Apply	Amplify
CERT C Secure Coding Standard	Training, Professional Certification, Application Certification	Adoption by source code analysis vendors and developers
CERT C++ Secure Coding Standard	Training, Professional Certification, Application Certification	Adoption by source code analysis vendors and developers
CERT Sun Microsystems Secure Coding Standard for Java	Training, Professional Certification, Application Certification	Adoption by source code analysis vendors and developers

Future Directions

Continue to develop and enhance existing secure coding standards and associated checkers

Develop secure coding standards for other languages and programming environments

- Web Development
- Language independent
- Ada, SPARK

Develop secure coding design patterns

Key Points

Everyday software defects cause the majority of software vulnerabilities.

Software developers are not always properly trained and equipped to program securely.

The result is numerous delivered defects, some of which can lead to vulnerabilities.

Understanding the sources of vulnerabilities and learning to program securely is imperative to protecting the Internet and ourselves from attack.

Questions

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

Contact CERT Coordination Center:

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh PA 15213-3890

USA

Hotline: **+1 412 268 7090**

CERT/CC personnel answer 24x7, 365 days per year

Fax: **+1 412 268 6989**

Mailto: **cert@cert.org**