Secure Coding Standards

Robert C. Seacord
CERT/CC
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+1-412-228-7608
rcs@cert.org

ABSTRACT

Secure coding standards define rules and recommendations to guide the development of secure software systems. Establishing secure coding standards provides a basis for secure system development as well as a common set of criteria that can be used to measure and evaluate software development efforts and software development tools and processes. This paper describes plans by the CERT/Coordination Center at the Software Engineering Institute at Carnegie Mellon University to establish, through a coordinated community effort, a set of secure coding standards for commonly used programming languages.

Keywords
Security, Standardization, Programming languages.

1. INTRODUCTION

Society's increased dependency on networked software systems has been matched by an increase in the number of attacks aimed at these systems. These attacks—directed at governments, corporations, educational institutions, and individuals—have resulted in loss and compromise of sensitive data, system damage, lost productivity, and financial loss [18].

Software vulnerability reports continue to grow at an alarming rate [1] and a significant number of them result in technical alerts [2]. To address this growing threat, the introduction of software vulnerabilities during software development and ongoing maintenance must be significantly curtailed.

An essential element of secure software development is well documented and enforceable coding standards. Coding standards encourage programmers to follow a uniform set of rules and guidelines determined by the requirements of the project and organization, rather than by the programmer's familiarity or preference. Once established, these standards can be used as a metric to evaluate source code (using manual or automated processes) to determine compliance with the standard.

There are numerous available sources, both online and in print, containing coding guidelines, best practices, suggestions, and tips. For example, the following books have been published containing C/C++ programming languages rules and guidelines:

- C++ Coding Standards: 101 Rules, Guidelines, and Best Practices [20]
- Effective C++ : 55 Specific Ways to Improve Your Programs and Designs (3rd Edition) [10]
- More Effective C++: 35 New Ways to Improve Your Programs and Designs [11]
- Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library [12]
- C++ Programming Guidelines [15]
- C Programming Guidelines [16]

Industry-specific standards such as the Motor Industry Software Reliability Association (MISRA) Guidelines for the use of the C language in critical systems [13] have also been published. Additionally, many companies have internal coding standards. An example of a publicly released coding standard is the Joint Strike Fighter Air Vehicle C++ Coding Standards [9].

Many online sources of coding practices and coding rules also exist, including the Build Security In web site [4] sponsored by the U.S. Department of Homeland Security (DHS) National Cyber Security Division. Other sources provide databases of known security flaws, such as the SAMATE Reference Dataset (SRD) maintained by NIST [14].

With all these sources of information, it might seem that a secure coding standard for these languages would be unnecessary. However, none of these sources provides a prescriptive set of secure coding standards that can be uniformly applied in the development of a software system. This conclusion is reinforced by the Secure Software Assurance Common Body of Knowledge [17] published by the U.S. Department of Homeland Security, which laments the "lack of public standards as such for secure programming."

2. SCOPE

At one extreme, a secure coding standard can be developed for a particular release of a compiler from a particular vendor. At the other extreme, the standards can be designed to be not only compiler independent but also language independent.

A coding standard for a particular compiler release has the largest possible benefit to the smallest group of users. Targeting a particular compiler allows for the definition of rules and guidelines that deal specifically with the peculiarities of that implementation, including defects in the implementation and non-standard extensions. At the other extreme, a language-independent coding standard has the least possible benefit to the largest possible group of users, as the rules and guidelines specified at this level of abstraction are largely notional.

The secure coding standards proposed by CERT are based on documented standard language versions as defined by official or *de facto* standards organizations. For example, secure coding standards are planned for the following languages:

- C programming language (ISO/IEC 9899:1999) [5]
- C++ programming language (ISO/IEC 9899:1999) [6]
- Sun Microsystems' Java2 Platform Standard Edition 5.0 API Specification [19]
- C# programming language (ISO/IEC 23270:2003) [7]

Applicable technical corrigenda and documented language extensions such as the ISO/IEC TR 24731 extensions to the C library [8] will also be considered.

The scope allows specific guidance to be provided to broad classes of users. Programming language standards, like those created by ISO/IEC, are primarily intended for compiler implementers. Secure coding standards are ancillary documents that provide rules and guidance directly to developers who program languages defined by these standards.

3. GOALS

The goal of each coding standard is to define a set of rules that are necessary (but not sufficient) to ensure the security of software systems developing in the respective programming languages.

A secure coding standard consists of *rules* and *recommendations*. Coding practices are defined to be rules when all of the following conditions are met

1. Violation of the coding practice will result in a security flaw that may result in an exploitable vulnerability.
2. There is an enumerable set of exceptional conditions (or no such conditions) where violating the coding practice is necessary to ensure the correct behavior for the program.
3. Conformance to the coding practice can be verified.

Rules must be followed to claim compliance with a standard unless an exceptional condition exists. If an exceptional condition is claimed, the exception must correspond to a pre-defined exceptional condition and the application of this exception must be documented in the source code.

Recommendations are guidelines or suggestions. Coding practices are defined to be recommendations when all of the following conditions are met

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.

Compliance with recommendations is not necessary to claim compliance with a coding standard. It is possible, however, to claim compliance with one or more verifiable guidelines. The set of recommendations that a particular development effort adopts depends on the security requirements of the final software product. Projects with high-security requirements can dedicate more resources to security, and are thus likely to adopt a larger set of recommendations.

4. DEVELOPMENT PROCESS

The development of a secure coding standard for any programming language is a difficult undertaking that requires significant community involvement. To produce standards of the highest possible quality, CERT is implementing the following development process:

1. Rules and recommendations for a coding standard are solicited from the communities involved in the development and application of each programming language, including the formal or de facto standard bodies responsible for the documented standard.
2. These rules and recommendations are edited by senior members of the CERT technical staff for content and style and placed in the Secure Coding area of CERT web site for comment and review [3].
3. The user community may then comment on the publically posted content using threaded discussions and other communication tools. Once a consensus develops that the rule or recommendation is appropriate and correct the final rule is incorporated into the coding standard.

Various groups, including the ISO/IEC JTC1/SC22/WG14 international standardization working group for the C programming language have expressed an interest in supporting this model.

5. USAGE

These rules may be extended with organization-specific rules. However, the rules contained in a standard must be obeyed to claim compliance with the standard.

Once established, tools can be developed or modified to determine compliance with the standard. Compliant software systems may then be certified as compliant by a properly authorized certification body.

Training may also be developed to educate software professionals regarding the appropriate application of secure coding standards. After passing an examination, these trained programmers may also be certified as secure coding professionals.

6. SYSTEM QUALITIES

Security is one of many system attributes that must be considered in the selection and application of a coding standard. Other attributes of interest include safety, portability, reliability, availability, maintainability, readability, and performance.

Many of these attributes are interrelated in interesting ways. For example, readability is an attribute of maintainability; both are important for limiting the introduction of defects during maintenance that could result in security flaws or reliability issues. Reliability and availability require proper resources management, which contributes also to the safety and security of the system. System attributes such as performance and security are often in conflict requiring tradeoffs to be considered.

The purpose of the secure coding standard is to promote software security. However, because of the relationship between security and other system attributes, the coding standards may provide recommendations that deal primarily with some other system attribute that also has a significant impact on security. The dual nature of these recommendations will be noted in the standard.

7. CONCLUSIONS

The development of secure coding standards is a necessary step to stem the ever-increasing threat from software vulnerabilities. Establishing secure coding standards allows for a common set of criteria that can be used to measure and evaluate software development efforts and software development tools and processes. Once established, secure coding standards can be incrementally improved, as a common understanding of existing problems and solutions allows for the development of more advanced security solutions.

8. ACKNOWLEDGMENTS

Thanks to Thomas Plum for suggesting this idea, John Benito for supporting this effort, and Hal Burch for his insights. Thanks to Jason Rafail, Jeff Gennari, Allen Householder, Chad Dougherty, and Claire Dixon for their review and thoughtful comments.

9. REFERENCES

[1] CERT/CC. See http://www.cert.org/stats/cert_stats.html for current statistics.

[2] CERT/CC. US-CERT's Technical Cyber Security Alerts. http://www.us-cert.gov/cas/techalerts/index.html

[3] CERT/CC. Secure Coding web site. http://www.cert.org/secure-coding/

[4] DHS. Build Security In web site. See https://buildsecurityin.us-cert.gov/

[5] INCITS/ISO/IEC 9899-1999. Programming Languages — C, Second Edition, 1999.

[6] INCITS/ISO/IEC 14882-2003. Programming Languages — C++, Second Edition, 2003.

[7] INCITS/ISO/IEC 23270-2003. Information technology - C# Language Specification ,2003.

[8] ISO/IEC WDTR 24731. Specification for Secure C Library Functions, 2004.

[9] Lockheed Martin. Joint Strike Fighter Air Vehicle C++ Coding Standards for the System Development and Demonstration Program. Document Number 2RDU00001 Rev C. December 2005.

[10] Meyers, Scott. Effective C++ : 55 Specific Ways to Improve Your Programs and Designs (3rd Edition). Addison-Wesley Professional. (September 2, 1997)

[11] Meyers, Scott. More Effective C++: 35 New Ways to Improve Your Programs and Designs. Addison-Wesley Professional. (December 29, 1995)

[12] Meyers, Scott. Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library. Addison-Wesley Professional. (June 6, 2001)

[13] MISRA C: 2004 Guidelines for the use of the C language in critical systems. MIRA Limited. Warwickshire, UK. October 2004. ISBN 0 9524156 4

[14] NIST. SAMATE Reference Dataset (SRD). See http://samate.nist.gov/SRD/srdFiles/

[15] Plum, Thomas. C Programming Guidelines. Plum Hall; 2nd edition (June 1989). ISBN: 0911537074.

[16] Plum, Thomas. C++ Programming. Plum Hall (November 1991) ISBN: 0911537104.

[17] Redwine, Jr. Samuel T, Editor. Secure Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software Draft Version 0.9. January 2006.

[18] Seacord, R. *Secure Coding in C and C++*. Addison-Wesley, 2005. See http://www.cert.org/books/secure-coding for news and errata.

[19] Sun Microsystems. Java2 Platform Standard Edition 5.0 API Specification, 2004. http://java.sun.com/j2se/1.5.0/docs/api/index.html

[20] Sutter, Herb. Alexandrescu, Andrei. C++ Coding Standards: 101 Rules, Guidelines, and Best Practices. Addison-Wesley Professional (October 25, 2004). ISBN: 0321113586.