Annotated Bibliography

Daconta, Machael C. (2000). "When Runtime.exec() won't." *JavaWorld*, December 2000. http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps_p.html.

This article fills in the gaps in the documentation for how to use the Runtime and Process objects in the Java runtime environment.

Gilliam, David P., John C. Kelly, John D. Powell and Matt Bishop. (2000). "Reducing Software Security Risk through an Integrated Approach." *IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WET ICE'00).

This paper describes work into tool sets and methodologies that can be used to analyze software systems for vulnerabilities. The tools include ways of analyzing designs as well as implementation code.

Graff, Mark G. and Kenneth R. van Wyk. (2003). *Secure Coding Principles & Practices*. O'Reilly & Associates, Inc.

This book covers techniques that the authors have found useful for designing and implementing secure software systems. The main interest of this project in the work is what it says concerning implementation issues.

Hoglund, Greg and Gary McGraw. (2004). *Exploiting Software: How to Break Code*. Addison-Wesley.

There are many ways to break the security of a software system. This book illustrates how software is attacked using the idea of *attack patterns*. An attack pattern is a blueprint for an exploit.

Koziol, Jack, David Litchfield, et. al. (2004) *The Shellcoders's Handbook.* (Wiley).

Another "how-to" guide. This one covers attacks on the operating systems Linux, Windows, Solaris, and HP/UX. It also illustrates attacks on products such as Oracle.

McGraw, Gary and Edward Felten. (1998). "Twelve rules for developing more secure Java code." Java World. http://www.javaworld.com/javaworld/jw-12-1998/jw-12-securityrules_p.html.

This article from the December 1998 issue of the magazine JavaWorld provides a set of guidelines for developing secure code. McGraw is also a co-author of *Exploiting Software*.

Oualline, Steve. (2003). *How Not to Program in C++*. No Starch Press.

This book illustrates its points about the hazards of programming in C++ using illustrations of mistakes. Although the book does contain humor, the errors illustrated are serious. The reader is presented with the source code of a broken program and asked to find the error. Hints and answers are numbered and spread out in an appendix. A truly humbling book.

Peikari, Cyrus and Anton Chuvakin. (2004). *Security Warrior.* O'Reilly & Associates, Inc.

This book is about how software systems are attacked. It covers many attacks on Windows operating systems and on networks.

Pincus, Jonathan, and Brandon Baker. (2004) "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns." *IEEE Security & Privacy*, July/August 2004.

The article describes three general-purpose types of exploits for buffer overruns and argues that these new techniques invalidate much of the traditional assumptions about stack overruns.

"Security Code Guidelines." (2000). Sun Microsystems, Inc. http://java.sun.com/security/seccodeguide.html.

This is a set of guidelines published by Sun Microsystems for secure programming in Java and C with an emphasis on Java.

Schneier, Bruce. (2000). *Secrets & Lies: Digital Security in a Networked World.* Wiley Publishing Inc.

Schneier is a well-known security expert. While this book is published for a general readership, it does provide a good high-level overview of the issues regarding security in software systems. It also provides a general introduction to attack trees and their use in security analysis.

Teer, Rich. (2001). "Secure C Programming." Sun Developer Network Community. http://developers.sun.com/solaris/articles/secure.html.

This is one of many technical articles describing the most common security programming errors that C programmers make.

Tevis, Jay-Evan J. and John A Hamilton, Jr. (2004). "Methods for the Prevention, Detection, and Removal of Software Security Vulnerabilities." *ACM Southeast Conference '04*.

Tevis and Hamilton present an argument that much of the current problems found in software code can be removed using a functional programming approach. Rather than an outright abandonment of imperative programming, the paper describes work that uses functional programming techniques to build better code analyzers for imperative languages.

Thompson, Ken. (1984). "Reflections on Trusting Trust." *Communications of the ACM*, August 1984.

This is a classic description of the problems of software security. It is based on the remarks made by Mr. Thompson when he accepting the ACM Turing Award.

Viega, John, Matt Messier. (2003). *Secure Programming Cookbook for C and C++*. O'Reilly & Associates.

This is a "cookbook" of coding examples for different problems encountered during development of secure systems when programming in C or C++.

Viega, John and Gary McGraw. (2001). *Building Secure Software*. Addison-Wesley.

This book shows how to avoid the kinds of vulnerabilities described in [Hoglund and McGraw]. The main interest of this project in the work is what it says concerning implementation issues.

Walder, Philip. (1997). "Functional programming: An angry half-dozen." *SIGPLAN Notices*, 33(2):25-30.

This essay describes some examples of using functional programming beyond just academic research. The examples, in Walder's terminology, show functional programming "used in anger" in the real world.

Wall, Larry, Tom Christiansen, and Jon Orwant. (2000). *Programming Perl 3rd Edition*. O'Reilly & Associates, Inc.

The authors of this work include the creator of the Perl language, Larry Wall. It covers the fundamentals of Perl programming. For this research paper, special emphasis will be put on the chapter covering the software security features of the language.

Wheeler, David A. (2003). *Secure Programming for Linux and Unix HOWTO*. Self published at http://www.dwheeler.com/secure-programs/.

This book provides a set of design and implementation guidelines for writing secure programs for Linux and Unix systems. Such programs include application programs used as viewers of remote data, web applications (including CGI scripts), network servers, and setuid/setgid programs. Specific guidelines for C, C++, Java, Perl, PHP, Python, Tcl, and Ada95 are included. For a current version of the book, see http://www.dwheeler.com/secure-programs.

Ullman, Jeffrey D. (1998). *Elements of ML Programming*. Prentice Hall.

This is a basic book on programming using Standard ML. It covers the 1997 version of ML.