13 Annotated Bibliography and References

Those who cannot remember the past
are condemned to repeat it.
GEORGE SANTAYANA

Annotated Bibliography: An Emerging Literature

What follows is a noncomprehensive bibliography of software security publications. This list is heavily biased toward recent publications. The references here can serve as a springboard to the wider literature. Below each reference is a brief description of the work and its place in the literature. All opinions are mine.

The bibliography is divided into three sections. First is a very short list of required reading (the top five list for software security). Second is a complete list of all references cited in this book. Third is a list of other important software security references not otherwise mentioned in this book. There are overlaps only between the required reading list and the other two lists.

Required Reading: The Top Five

This is a completely biased list of the top five publications to read in software security (presented in alphabetical order). If you have time to read only a handful of stuff, read everything on this list first.

1. [**Anderson 2001**] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons, New York, 2001. <http://www.cl.cam.ac.uk/~rja14/book.html>

   This is probably the best security book on the market. If you can buy only one other book relevant to software security, buy this one. *Security Engineering* is about building systems that remain dependable in the face of malicious attack, unintentional error, or accident. Anderson's treatment focuses on the tools, processes, and methods needed to design, implement, and test complete systems and to adapt existing systems as their environment evolves.

2. **[Hoglund and McGraw 2004]** Greg Hoglund and Gary McGraw. *Exploiting Software: How to Break Code*. Addison-Wesley, Boston, MA, 2004. <http://www.exploitingsoftware.com>

   One of my three software security books. *Exploiting Software* goes way beyond the script kiddie hacking basics by describing the software attacker's toolkit and how it is commonly used by bad guys. This book includes hard-core information on real attacks against real software. It also introduces the notion of attack patterns.

3. **[Howard and LeBlanc 2003]** Michael Howard and David LeBlanc. *Writing Secure Code,* 2nd edition. Microsoft Press, Redmond, WA, 2003. Mike Howard's blog serves as the de facto site for this book <http://blogs.msdn.com/michael_howard/>.

   Writing Secure Code is a very good treatment of software security with an emphasis on code and implementation problems (bugs). The introduction of the STRIDE model is particularly noteworthy. If you're serious about software security, you need to read this book.

4. **[Saltzer and Schroeder 1975]** Jerome Saltzer and Michael Schroeder. "The Protection of Information in Computer Systems," *Proceedings of the IEEE* 9(63), September 1975, pp. 1278–1308. <http://web.mit.edu/Saltzer/www/publications/protection/>

   An absolutely classic paper that everyone cites but few actually read. This paper introduces and discusses a number of central security principles. The paper itself is a pithy, short, essential read. (By the way, a treatment of the principles idea related to software security can be found in *Building Secure Software*.)

5. **[Viega and McGraw 2001]** John Viega and Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, Boston, MA, 2001. <http://www.buildingsecuresoftware.com/>

   One of my three software security books. *Building Secure Software* launched the field of software security. Though there is plenty of code in *BSS,* the book itself is really a philosophical treatment introducing the idea of building security in.

References Cited in *Software Security: Building Security In*

A complete alphabetical listing of all references in this book, including those references mentioned in footnotes.

[**Abbott et al. 1976**] Robert Abbott, Janet Chin, James Donnelley, William Konigsford, Shigeru Tokubo, and Douglas Webb. "Security Analysis and Enhancements of Computer Operating Systems," NBSIR 76-1041, National Bureau of Standards, ICST, Washington, DC, 1976.

Abbott introduces the RISOS taxonomy of computer security problems related to operating systems. Very early work in understanding security vulnerabilities.

[**Alexander 2003**] Ian Alexander. "Misuse Cases: Use Cases with Hostile Intent," *IEEE Software* 20(1), January/February 2003, pp. 58–66.

Alexander advocates using misuse and use cases together to conduct threat and hazard analysis during requirements analysis.

[**Anderson 2001**] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons, New York, 2001.

See entry in *Required Reading*.

[**Arkin, Stender, and McGraw 2005**] Brad Arkin, Scott Stender, and Gary McGraw. "Software Penetration Testing," *IEEE Security & Privacy* 3(1), 2005, pp. 84–87.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[**Ashcraft and Engler 2002**] Ken Ashcraft and Dawson Engler. "Using Programmer-Written Compiler Extensions to Catch Security Holes," *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA, IEEE Computer Society Press, 2002, pp. 131–147.

Engler's work on static analysis is now being commercialized by Coverity. This academic paper describes the bug-finding technology developed at UC Berkeley.

[**Aslam 1995**] Taimur Aslam. "A Taxonomy of Security Faults in the UNIX Operating System." Master's Thesis, Purdue University, 1995.

An early taxonomy focused on UNIX security problems.

[**Ball and Rajamani 2001**] Tom Ball and Sriram Rajamani. "Automatically Validating Temporal Safety Properties of Interfaces," *Proceedings of the 8th International SPIN Workshop on Model Checking of Software,* Springer Lecture Notes in Computer Science, vol. 2057, 2001, pp. 103–122.

The SLAM model checker uses predicate abstraction to examine program safety properties. Tom Ball now runs a research group at Microsoft.

[Barnum and McGraw 2005] Sean Barnum and Gary McGraw. "Knowledge for Software Security," *IEEE Security & Privacy* 3(2), 2005, pp. 74–78.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[Bisbey and Hollingworth 1978] Richard Bisbey and Dennis Hollingworth. "Protection Analysis Project Final Report," ISI/RR-78-13, DTIC AD A056816, USC/Information Sciences Institute, 1978.

A description of the Protection Analysis (PA) project meant to enable anybody (with or without any knowledge about computer security) to discover security errors in a system by using a pattern-directed approach. Formalized patterns were used to search for corresponding errors. The PA project was the first project to explore automation of security defect detection.

[Bishop 2003] Matt Bishop. *Computer Security: Art and Science*. Addison-Wesley, Boston, MA, 2003.

A decent though overly formal textbook on computer security. Matt Bishop is one of the pioneers of software security. Echoes of his philosophy of building security in are evident in this book.

[Bishop and Dilger 1996] Matt Bishop and Mike Dilger. "Checking for Race Conditions in File Accesses," *Computing Systems* 9(2), 1996, pp. 131–152.

Matt Bishop's seminal paper explains a simple static analysis tool for detecting time-of-check–time-of-use (TOCTOU) defects.

[Bush, Pincus, and Sielaff 2000] William Bush, Jonathan Pincus, and David Sielaff. "A Static Analyzer for Finding Dynamic Programming Errors," *Software Practice and Experience,* 30(7), June 2000, pp. 775–802.

The only paper published about Prefix, the complicated precursor to Prefast invented by Jon Pincus and used internally at Microsoft for many years.

[Cavusoglu, Mishra, and Raghunathan 2002] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. "The Effect of Internet Security Breach Announcements on Market Value of Breached Firms and Internet Security Developers," Technical Report from the University of Texas at Dallas School of Management, February 2002.

A minor academic study indicating a link between security events and negative stock price movements.

[Chen and Wagner 2002] Hao Chen and David Wagner. "MOPS: An Infrastructure for Examining Security Properties of Software," *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS2002),* Washington, DC, ACM Press, 2002, pp. 235–244.

MOPS takes a model-checking approach to look for violations of temporal safety properties. Developers can model their own safety properties, and some have used the tool to check for privilege management errors, incorrect construction of `chroot` jails, file access race conditions, and ill-conceived temporary file schemes.

[Chess 2002] Brian Chess. "Improving Computer Security Using Extended Static Checking," *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA, IEEE Computer Society Press, 2002, pp. 118–130.

The Eau Claire tool uses a theorem prover to create a general specification-checking framework for C programs. It can help find common security problems like buffer overflows, file access race conditions, and format string bugs. Developers can use specifications to ensure that function implementations behave as expected.

[Chess and McGraw 2004] Brian Chess and Gary McGraw. "Static Analysis for Security," *IEEE Security & Privacy* 2(6), 2004, pp. 76–79.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[Cheswick and Bellovin 1994] Bill Cheswick and Steve Bellovin. *Firewalls and Internet Security,* 1st edition. Addison-Wesley, Reading, MA, 1994.

The very first edition of a classic security tome. See the next entry for up-to-date information; especially note that the new edition is coauthored with Avi Rubin.

[Cheswick, Bellovin, and Rubin 2003] Bill Cheswick, Steve Bellovin, and Avi Rubin. *Firewalls and Internet Security,* 3rd edition. Addison-Wesley, Boston, MA, 2003.

A classic computer security book, now available in a revised and updated edition featuring Avi Rubin as coauthor.

[Christey 2005] Steven Christey. "PLOVER—Preliminary List of Vulnerability Examples for Researchers," NIST Draft, August 2005 (unpublished).

An unpublished attempt to categorize the CVE vulnerabilities into some kind of bottom-up taxonomy.

[Das, Lerner, and Seigle 2002] Manuvir Das, Sorin Lerner, and Mark Seigle. "ESP: Path-Sensitive Program Verification in Polynomial Time," *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI2002),* Berlin, Germany, ACM Press, 2002, pp. 57–68.

The static analysis tool ESP is a large-scale property verification approach.

[Davis et al. 2004] Noopur Davis, Samual Redwine, Gerlinde Zibuski, Gary McGraw, and Watts Humphrey. "Summary of National Cyber Security Summit Subgroup Report: Processes for Producing Secure Software." April 2004.

A committee-produced paper describing the software security problem used to set national policy. The touchpoints were prominently included in this paper. The complete report can be found here: <http://www.cyberpartnership.org>.

[Engler et al. 2000] Dawson Engler, Benjamin Chelf, Andy Chou, and Seth Hallem. "Checking System Rules Using System-Specific, Programmer-Written Compiler Extensions," *Proceedings of the Symposium on Operating System Design and Implementation (OSDI),* San Diego, CA, USENIX Association, October 2000.

This paper introduces a set of small extensions that were used to find roughly 500 bugs in Linux, OpenBSD, and Xok. The engine behind Coverity is described in this paper.

[Evans et al. 1994] David Evans, John Guttag, Jim Horning, and Yang Meng Tan. "LCLint: A Tool for Using Specifications to Check Code," *Proceedings of the SIGSOFT Symposium on the Foundations of Software Engineering,* New Orleans, LA, ACM Press, December 1994, pp. 87–96.

LCLint is introduced, a simple tool that accepts ANSI C programs and some annotations to find and report inconsistencies.

[Fagan 1976] Michael Fagan. "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal* 15(3), 1976, pp. 182–211.

The seminal work on manual code inspection.

[Farmer and Venema 2005] Dan Farmer and Wietse Venema. *Forensic Discovery.* Addison-Wesley, Boston, MA, 2005.

Dan Farmer and Wietse Venema (purveyors of SATAN and other great security stuff) recently released this long-awaited, excellent new tome on forensics.

[Foster, Terauchi, and Aiken 2002] Jeffrey Foster, Tachio Terauchi, and Alex Aiken. "Flow-Sensitive Type Qualifiers," *Proceedings of the ACM Conference on Programming Language Design and Implementation (PLDI2002),* Berlin, Germany, ACM Press, 2002, pp. 1–12.

One of the many papers on CQual. Inspired by Perl's taint mode, CQual uses type qualifiers to perform a taint analysis, which detects format string vulnerabilities in C programs. CQual requires a programmer to annotate a few variables as either tainted or untainted and then uses type inference rules (along with pre-annotated system libraries) to propagate the qualifiers. Once the qualifiers are propagated, the system can detect format string vulnerabilities by type checking.

[Geer 1998] Dan Geer. "Risk Management Is Where the Money Is," The Digital Commerce Society of Boston, Boston, MA, November 1998. This paper has been widely reprinted, including RISKS 20.06 <http://catless.ncl.ac.uk/Risks/20.06.html>.

An early discussion of the criticality of risk management to security. This paper provides a reasonable overview and history.

[Geer et al. 2003] Dan Geer, Rebecca Bace, Peter Gutmann, Perry Metzger, Charles Pfleeger, John Quarterman, and Bruce Schneier. "Cyber*In*security: The Cost of Monopoly, How the Dominance of Microsoft's Products Poses a Risk to Security." Published on the Web by the Computer & Communications Industry Association (CCIA), September 2003. <http://www.ccianet.org/papers/cyberinsecurity.pdf>

The famous "monoculture" paper that caused Dan Geer to be fired from @stake. Computer security is so important that it is becoming political. This paper argues that by dominating the software market so completely, Microsoft is putting security at risk.

[Ghosh, O'Connor, and McGraw 1998] Anup Ghosh, Tom O'Connor, and Gary McGraw. "An Automated Approach for Identifying Potential Vulnerabilities in Software," *Proceedings of the IEEE Symposium on Security and Privacy,* Oakland, CA, IEEE Computer Society Press, May 1998, pp. 104–114.

FIST is a tool for software fault injection for security. This work inspired a number of commercial dynamic testing tools.

[Gilb and Graham 1993] Tom Gilb and Dorothy Graham. *Software Inspection*. Addison-Wesley, Reading, MA, 1993.

After Fagan [1976], this book is the classic text on code review.

[Graff and van Wyk 2003] Mark Graff and Kenneth van Wyk. *Secure Coding: Principles and Practices*. O'Reilly and Associates, Sebastopol, CA, 2003.

One of the key books in software security, aimed at network and operations security types. This book explains the importance of software security to computer security people.

[Gutmann 2004] Peter Gutmann. "Simplifying Public Key Management," *IEEE Computer* 37(2), February 2004, pp. 101–103.

A paper explaining why many security errors exist because of user problems caused by overly complicated technology (ever try to use early versions of `pgp`?). Simplicity for users and consumers of software and software security technology is essential.

[Henzinger et al. 2003] Thomas Henzinger, Ranjit Jhala, Rupak Majumdar, and Gregoire Sutre. "Software Verification with BLAST," *Proceedings of the 10th International Workshop on Model Checking of Software,* Springer Lecture Notes in Computer Science, vol. 2648, 2003, pp. 235–239.

A paper explaining the BLAST model checker, which uses predicate abstraction to examine program safety properties.

[Hoglund and Butler 2005] Greg Hoglund and James Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, Boston, MA, 2005.

The first complete book on the important topic of rootkits. Rootkits are the apex of the attacker's toolkit, and understanding how they really work is essential for today's software security professionals. Better get this book.

[Hoglund and McGraw 2004] Greg Hoglund and Gary McGraw. *Exploiting Software: How to Break Code*. Addison-Wesley, Boston, MA, 2004. <http://www.exploitingsoftware.com>

See entry in *Required Reading*.

[Hope, McGraw, and Anton 2004] Paco Hope, Gary McGraw, and Annie Anton. "Misuse and Abuse Cases: Getting Past the Positive," *IEEE Security & Privacy* 2(3), 2004, pp. 32–34.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[Hovemeyer and Pugh 2004] Dave Hovemeyer and William Pugh. "Finding Bugs Is Easy," *Companion of the 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA),* Vancouver, Canada, ACM Press, 2004.

Bill Pugh's FindBugs program is a very popular open source code analysis system for Java bytecode.

[Howard and LeBlanc 2002] Michael Howard and David LeBlanc. *Writing Secure Code,* 1st edition. Microsoft Press, Redmond, WA, 2002.

See entry in *Required Reading*.

[Howard and LeBlanc 2003] Michael Howard and David LeBlanc. *Writing Secure Code,* 2nd edition. Microsoft Press, Redmond, WA, 2003.

See entry in *Required Reading*.

[Howard, LeBlanc, and Viega 2005] Michael Howard, David LeBlanc, and John Viega. *19 Deadly Sins of Software Security*. McGraw-Hill Osborne Media, New York, 2005.

This book discusses in detail 19 serious software security problems. The 19 sins are not presented in a hierarchy.

[Howard and Lipner 2003] Michael Howard and Steve Lipner. "Inside the Windows Security Push," *IEEE Security & Privacy* 1(1), 2003, pp. 57–61.

A description of Microsoft's Trustworthy Computing Initiative one year after the effort began. Microsoft's work provides a critical case study for the adoption of software security best practices in a large enterprise.

[Jurjens 2001] Jan Jurjens. "Towards Secure Systems Development with UMLsec," *Proceedings of FASE'01*. Springer Lecture Notes in Computer Science, 2001.

UMLsec is one way of thinking about security at the design level. This work is overly focused on security features.

[Kernighan and Ritchie 1988] Brian Kernighan and Dennis Ritchie. *The C Programming Language,* 2nd edition. Prentice Hall, New York, 1988.

The C bible. Unfortunately, this language has serious security problems. The string functions are particularly notorious for introducing buffer overflow conditions. And `gets()`? Ouch. The best software security advice about C is "don't use it."

[Koziol et al. 2004] Jack Koziol, David Litchfield, Dave Aitel, Chris Anley, Sinan "noir" Eren, Neel Mehta, and Riley Hassell. *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*. John Wiley & Sons, New York, 2004.

One of the books helping to describe (in great technical detail) how software attacks work. This book makes an excellent companion to *Exploiting Software*. This is a black hat must-read.

[Landwehr, Bull, and McDermott 1993] Carl Landwehr, Alan Bull, and John McDermott. "A Taxonomy of Computer Program Security Flaws, with Examples," Technical Report NRL/FR/5542—93/9591, United States Navy, Naval Research Laboratory, November 1993.

An important early taxonomy of computer security problems. This work set the stage for an escalation of excellent computer security research in the mid-1990s.

[Larochelle and Evans 2001] David Larochelle and David Evans. "Statically Detecting Likely Buffer Overflow Vulnerabilities," *Proceedings of the 10th*

Usenix Security Symposium (USENIX'01), Washington, DC, USENIX Association, 2001.

Splint extends the lint concept into the security realm. By adding code annotations, developers can enable splint to find abstraction violations, unannounced modifications to global variables, and possible use-before-initialization errors. Splint can also reason about minimum and maximum array bounds accesses if it is provided with function pre- and postconditions.

[Leveson 1995] Nancy Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, Reading, MA, 1995.

The classic book on software safety. Safety has a number of critical lessons to teach software security, only a few of which have been absorbed so far.

[McDermott and Fox 1999] John McDermott and Chris Fox. "Using Abuse Case Models for Security Requirements Analysis," *Proceedings of the 15th Annual Computer Security Applications Conference,* Scottsdale, AZ, IEEE Computer Society Press, 1999, p. 55.

The first paper on record about abuse cases.

[McGraw 1998] Gary McGraw. "Testing for Security During Development: Why We Should Scrap Penetrate-and-Patch," *IEEE Aerospace and Electronic Systems* 13(4), 1998, pp. 13–15.

A paper describing why penetrate-and-patch is a failed approach. This paper represents some of my earliest thinking about software security. Note that it was published in a journal devoted to very high assurance systems (those that control aircraft).

[McGraw 2003] Gary McGraw. "From the Ground Up: The DIMACS Software Security Workshop," *IEEE Security & Privacy* 1(2), 2003, pp. 59–66.

The results of the first conference devoted entirely to software security. This intimate workshop of around 50 people helped to crystallize and define the emerging field of software security. Presentations and notes from the workshop are here <http://www.cigital.com/ssw/>.

[McGraw 2004] Gary McGraw. "Software Security," *IEEE Security & Privacy* 2(2), 2004, pp. 80–83.

The first of the BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[McGraw 2005] Gary McGraw. "The 7 Touchpoints of Secure Software," *Software Development,* September 2005, pp. 42–43.

A popular press treatment of the touchpoints that appeared in *Software Development* magazine.

[**McGraw and Felten 1996**] Gary McGraw and Edward Felten. *Java Security: Hostile Applets, Holes, and Antidotes*. John Wiley & Sons, New York, 1996.

The first book on Java security, written with Ed Felten, leader of the Princeton Team. This book made quite a splash when it appeared. See the next entry for *Securing Java,* the second edition.

[**McGraw and Felten 1999**] Gary McGraw and Edward Felten. *Securing Java: Getting Down to Business with Mobile Code*. John Wiley & Sons, New York, 1999. <http://www.securingjava.com/>

The second edition of my book *Java Security,* updated with new attacks and advice. The complete book is available for free on the Web. Ed Felten ran the Princeton team of security researchers who consistently challenged assumptions about Java.

[**McGraw and Morrisett 2000**] Gary McGraw and Greg Morrisett. "Attacking Malicious Code: A Report to the Infosec Research Council," *IEEE Software* 17(5), September/October 2000, pp. 33–41.

Malicious code is a side effect of bad software. This paper introduced the *trinity of trouble*. This paper describes a U.S. government-sponsored set of workshops (which I set up and chaired) meant to dig deeply into the root causes of viruses, worms, and other nasty beasties.

[**Mead and McGraw 2005**] Nancy R. Mead and Gary McGraw. "A Portal for Software Security," *IEEE Security & Privacy* 3(4), 2005, pp. 75–79.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[**Miller et al. 1995**] Barton Miller, David Koski, Cjin Lee, Vivekananda Maganty, Ravi Murphy, Ajitkumar Natarajan, and Jeff Steidl. "Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services," Technical Report CS-TR-95-1268, University of Wisconsin, April 1995.

An excellent description of the fuzz tool, five years after it was first introduced (by the same authors). The simple idea of sending random input to UNIX commands and seeing what happens helped to spark dynamic testing approaches offered on the commercial market today.

[**Pincus and Baker 2004**] Jon Pincus and Brandon Baker. "Beyond Stack Smashing: Recent Advances in Exploiting Buffer Overruns," *IEEE Security & Privacy* 2(4), 2004, pp. 20–27.

An in-depth description of new buffer overflow attacks not yet commonly encountered in the wild. You know what that means—coming soon to software near you. This was the best paper in a special issue of

IEEE Security & Privacy magazine, which I edited with Ivan Arce, devoted to attacking systems.

[Potter and McGraw 2004] Bruce Potter and Gary McGraw. "Software Security Testing," *IEEE Security & Privacy* 2(5), 2004, pp. 81–85.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[Saltzer and Schroeder 1975] Jerome Saltzer and Michael Schroeder. "The Protection of Information in Computer Systems," *Proceedings of the IEEE* 9(63), September 1975, pp. 1278–1308.

See entry in *Required Reading*.

[Sindre and Opdahl 2000] Guttorm Sindre and Andreas Opdahl. "Eliciting Security Requirements by Misuse Cases," *Proceedings of the 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-37'00),* Sydney, Australia, IEEE Press, 2000, pp. 120–131.

Sindre and Opdahl explain how to extend use case diagrams with misuse cases. Their basic idea is to represent the actions that systems should prevent in tandem with those that systems should support so that security analysis of requirements is easier.

[Stubblefield, Ioannides, and Rubin 2004] Adam Stubblefield, John Ioannides, and Avi Rubin. "A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)," *ACM Transactions on Information and System Security,* May 2004, pp. 319–332.

WEP is a prime example of the widespread security risk brought about by architectural security flaws.

[Swiderski and Snyder 2004] Frank Swiderski and Window Snyder. *Threat Modeling*. Microsoft Press, Redmond, WA, 2004.

The unfortunately titled book explaining how Microsoft approaches security risk analysis. This book is worth a quick glance.

[Taylor and McGraw 2005] Dan Taylor and Gary McGraw. "Adopting a Software Security Improvement Program," *IEEE Security & Privacy* 3(3), 2005, pp. 88–91.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[Tsipenyuk, Chess, and McGraw 2005] Katrina Tsipenyuk, Brian Chess, and Gary McGraw. "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors," *Proceedings of the NIST Workshop on Software Security*

Assurance Tools, Techniques, and Metrics (SSATTM), Los Angeles, CA, 2005.

A paper introducing the seven pernicious kingdoms and associated phyla expounded in this book. A related BSI article from *IEEE Security & Privacy* magazine is also in the works but was not published at the time of this writing. See <http://www.computer.org/security> for subscription information.

[**van Wyk and McGraw 2005**] Kenneth R. van Wyk and Gary McGraw. "Bridging the Gap between Software Development and Information Security," *IEEE Security & Privacy* 3(4), 2005, pp. 64–68.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[**Verdon and McGraw 2004**] Denis Verdon and Gary McGraw. "Risk Analysis in Software Design," *IEEE Security & Privacy* 2(4), 2004, pp. 79–84.

One of the original BSI articles from *IEEE Security & Privacy* magazine that sparked this book. See <http://www.computer.org/security> for subscription information.

[**Viega et al. 2000a**] John Viega, J. T. Bloch, Tadyoshi Kohno, and Gary McGraw. "ITS4: A Static Vulnerability Scanner for C and C++ Code," *Proceedings of Annual Computer Security Applications Conference,* New Orleans, LA, December 2000, pp. 257–267.

An early ITS4 publication describing a simple source code security analysis tool. The paper includes a couple of case studies showcasing how to use ITS4. This paper won the best paper award at ACSAC in 2000 even though it is not really all that good.

[**Viega et al. 2000b**] John Viega, Gary McGraw, Tom Mutdosch, and Ed Felten. "Statically Scanning Java Code: Finding Security Vulnerabilities," *IEEE Software* 17(5), September/October 2000, pp. 68–74.

A paper describing a very simple static analysis tool written by Tom during a summer internship at Cigital. The Jscan prototype captured the guidelines from *Securing Java* [McGraw and Felten 1999] in a simple tool.

[**Viega and McGraw 2001**] John Viega and Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, Boston, MA, 2001. <http://www.buildingsecuresoftware.com/>

See entry in *Required Reading*.

[Voas and McGraw 1998] Jeff Voas and Gary McGraw. *Software Fault Injection: Inoculating Programs against Errors*. John Wiley & Sons, New York, 1998.

The first book in the world on software fault injection, a technology pioneered by Jeff Voas, cofounder of Cigital.

[Wagner et al. 2000] David Wagner, Jeffrey Foster, Eric Brewer, and Alexander Aiken. "A First Step Towards Automated Detection of Buffer Over-run Vulnerabilities," *Proceedings of the Year 2000 Network and Distributed System Security Symposium* (NDSS), San Diego, CA, 2000, pp. 3–17.

Wagner describes a tool that uses constraints to scan for buffer overflows in C code. The paper has an excellent analysis of the buffer overflow problem itself. This paper won the best paper award at ISOC NDSS in 2000 and most certainly deserved it.

[Walsh 2003] Larry Walsh. "Trustworthy Yet?" *Information Security Magazine,* February 2003. <http://infosecuritymag.techtarget.com/2003/feb/cover.shtml>

A skeptical look at Microsoft's Trustworthy Computing Initiative, one year into the program.

[Whittaker and Thompson 2003] James Whittaker and Herbert Thompson. *How to Break Software Security*. Addison-Wesley, Boston, MA, 2003.

A good, simple, black-hat-related book about probing software security through input. Whittaker is a master of compelling tools that are easy to understand and useful at the same time.

[Wing 2003] Jeannette Wing. "A Call to Action: Look Beyond the Horizon," *IEEE Security & Privacy* 1(6), 2003, pp. 62–67.

Jeannette wrote this interesting paper after a summer at Microsoft being exposed to software security in a large corporate software environment. Software security is listed among the top three major issues to work on in computer security.

Government and Standards Publications Cited

[IEC 61508] International Standards Organization, "IEC 61508"; Version 4.0 (1997). <http://www.iee.org>

[NIST 800-30] U.S. Federal Government, NIST Special Publication 800-30, "Risk Management Guide for Information Technology Systems." <http://csrc.nist.gov/publications/nistpubs/>

[NIST 800-37] U.S. Federal Government, NIST Special Publication 800-37,

"Guide for the Security Certification and Accreditation of Federal Information Systems." <http://csrc.nist.gov/publications/nistpubs/>

[**NIST 800-53**] U.S. Federal Government, NIST Special Publication 800-53, "Recommended Security Controls for Federal Information Systems." <http://csrc.nist.gov/publications/nistpubs/>

Other Important References

There are plenty of other references not directly cited in this book that are worth a look. Though this list is by no means complete, it can serve as a springboard into the wider software security literature.

[**Aleph1 1996**] Aleph One. "Smashing the Stack for Fun and Profit," *Phrack* 49, November 1996.

A comprehensive study of classic stack-smashing attacks. This is among the earliest papers dedicated to software security. *Phrack* is an excellent black hat resource that is well worth checking out.

[**Amoroso 1994**] Ed Amoroso. *Fundamentals of Computer Security Technology*. Prentice Hall, Englewood Cliffs, NJ, 1994.

Introduction of threat trees, the Bell-LaPadula model, Biba integrity, and other basic models. An oldie, but a goodie.

[**Anderson and Kuhn 1996**] Ross Anderson and Marcus Kuhn. "Tamper Resistance—A Cautionary Note," *Proceedings of the Second Usenix Workshop on Electronic Commerce,* Oakland, CA, USENIX Association, November 1996, pp. 1–11. <http://www.cl.cam.ac.uk/users/rja14/tamper.html>

Attacking smart cards with interesting, surprising attacks. This great article shows how to think like an attacker (with your black hat on).

[**Anderson and Needham 1995**] Ross Anderson and Roger Needham. "Programming Satan's Computer," *Computer Science Today,* Springer Lecture Notes in Computer Science, vol. 1000, 1995, pp. 426–441. <http://www.cl.cam.ac.uk/ftp/users/rja14/satan.ps.gz>

Why programming distributed systems is really hard.

[**Arbaugh, Fithen, and McHugh 2000**] Bill Arbaugh, Bill Fithen, and John McHugh. "Windows of Vulnerability: A Case Study Analysis," *IEEE Computer* 33(12), December 2000, pp. 52–59.

Ever wonder whether patching works? This paper shows conclusively that it doesn't work very well at all. The most surprising result describes how attack scripts appear to be developed well after patches are released.

[Bell and LaPadula 1974] David Bell and Len LaPadula. "Secure Computer Systems," ESD-TR-73-278, Mitre Corporation; vols. I and II (November 1973), vol. III (April 1974).

A classic paper describing principals (actors) and objects in a matrix of permissions. This is the seminal work behind access control lists and role-based access control.

[Brooks 1995] Frederick Brooks, Jr. *The Mythical Man-Month: Essays on Software Engineering,* 2nd edition. Addison-Wesley, Reading, MA, 1995.

Ever wonder why throwing more programmers at a software project only makes things take longer? Read this great book and find out why.

[Brown 2000] Keith Brown. *Programming Windows Security*. Addison-Wesley, Boston, MA, 2000.

Windows security APIs. Security features are important, too.

[Cowan et al. 1998] Crispin Cowan, Calton Pu, David Maier, Heather Hinton, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. "Automatic Detection and Prevention of Buffer-Overflow Attacks," *Proceedings of the 7th USENIX Security Symposium,* San Antonio, TX, USENIX Association, January 1998, pp. 63–78.

Stackguard was the clear inspiration for Microsoft's maligned /GS flag. Though I am not a fan of detecting or stopping buffer overflows dynamically, this is a great paper.

[Denning 1998] Dorothy Denning. *Information Warfare and Security,* Addison-Wesley, Reading, MA, 1998.

Possibly the ultimate black hat technique—war. This is a scary and relevant book well worth comprehending.

[DOD 1985] *Trusted Computer System Evaluation Criteria* ("The Orange Book"). U.S. Department of Defense, 1985.

A failed attempt, but a valiant attempt to codify security assurance. The problem with this approach to security is that computer systems are extensible, networked, and way more complicated than ever.

[Ford 1994] Warwick Ford. *Computer Communications Security: Principles, Standard Protocols, and Techniques*. Prentice Hall, Englewood Cliffs, NJ, 1994.

Network and communications security. Basic coverage of crypto, CIA, and some aspects of privacy. The Open Systems Interconnection (OSI) security architecture explained.

[Forrest, Hofmeyr, and Somayaji 1997] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. "Computer Immunology," *Communications of the ACM* 40(10), October 1997, pp. 88–96.

Why computer security might benefit by analogy with biology.

[**Gamma et al. 1995**] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley, Reading, MA, 1995.

This is an instrumental software architecture book. This book led to the idea of attack patterns.

[**Garfinkel and Spafford 1996**] Simson Garfinkel and Gene Spafford. *Practical UNIX and Internet Security,* 2nd edition. O'Reilly, Sebastopol, CA, 1996.

A classic tome on UNIX security. UNIX root must-read, but applicable widely to other operating systems.

[**Gasser 1988**] Morrie Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold, New York, 1988.

A very old but interesting read that anticipates the philosophy of building security in some twenty years earlier.

[**Goldberg and Wagner 1996**] Ian Goldberg and Dave Wagner. "Randomness and the Netscape Browser," *Dr. Dobbs Journal,* no. 243, January 1996, pp. 66–70.

A great case study in broken software and the resulting attacks.

[**Gollmann 1999**] Dieter Gollmann. *Computer Security*. John Wiley & Sons, New York, 1999.

Probably the best basic security book (textbook style) out there. Use this to enhance Amoroso [1994]. By the way, we need a better basic computer security book.

[**Kahn 1996**] David Kahn. *The Code-Breakers* (revised edition). Scribner, New York, 1996.

A historically accurate treatment of cryptography. Long, interesting, and worth slogging through.

[**Kaner and Pels 1998**] Cem Kaner and David Pels. *Bad Software: What to Do When Software Fails*. John Wiley & Sons, New York, 1998.

Ever wonder whether those software licenses that you click on stand up in court? This lawyer tells why they don't.

[**Knuth 1997**] Donald Knuth. *The Art of Computer Programming: Seminumerical Algorithms,* 3rd Edition. Addison-Wesley, Reading, MA 1997.

Knuth; alpha geek. What, you don't own this book and its two companions? For shame.

[**Kocher 1999**] Paul Kocher. "Differential Power Analysis," *Advances in Cryptology—Crypto 99,* Springer Lecture Notes in Computer Science, vol. 1666, 1999, pp. 388–397.

How smart cards leak critical security information through their power consumption. This is a great study in thinking outside the box to break a system.

[**Krusl 1998**] Ivan Krsul. *Software Vulnerability Analysis*. Ph.D. Thesis, COAST TR 98-09, Department of Computer Sciences, Purdue University, 1998.

This thesis is one of the first modern attempts at a computer security vulnerability taxonomy.

[**LaMacchia et al. 2002**] Brian LaMacchia, Sebastian Lang, Matther Lyons, Rui Martin, and Kevin Price. *.NET Framework Security*. Addison-Wesley, Boston, MA, 2002.

From the guy who brought you .NET security. Good, but not very clear.

[**Maguire 1993**] Steve Maguire. *Writing Solid Code*. Microsoft Press, Redmond, WA, 1993.

Too bad the Microsoft guys didn't eat their own dog food in 1993! Get this book. Internalize.

[**McClure, Scambray, and Kurtz 1999**] Stuart McClure, Joel Scambray, and George Kurtz. *Hacking Exposed: Network Security Secrets and Solutions*. Osborne, New York, 1999.

The now-classic script kiddie book explaining black hat computer security to the masses. Not much software security in here, but an important book nonetheless.

[**McGraw 1999**] Gary McGraw. "Software Assurance for Security," *IEEE Computer* 32(4), April 1999, pp. 103–105.

My first real paper on software security. This short article introduces the idea of software risk management for security.

[**Menezes, van Oorschot, and Vanstone 1997**] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. <http://www.cacr.math.uwaterloo.ca/hac/>

The best applied cryptography book. Written by hard-core crypto guys.

[**Miller, Fredricksen, and So 1990**] Barton Miller, Lars Fredricksen, and Bryan So. "An Empirical Study of the Reliability of UNIX Utilities," *Communications of the ACM* 33(12), December 1990, pp. 32–44.

The first fuzz paper. See the second entry in the references cited for *Software Security* earlier in this chapter.

[**Necula and Lee 1998**] George Necula and Peter Lee. "Safe, Untrusted Agents Using Proof-Carrying Code," *Mobile Agents and Security,* Springer Lecture Notes in Computer Science, vol. 1419, 1998, pp. 61–91.

The seminal paper on proof-carrying code (also known as certified code). This paper describes a system very much likely to be fielded in the future.

[**Neumann 1995**] Peter Neumann. *Computer-Related Risks*. Addison-Wesley, Reading, MA, 1995.

From the comp.risks mailing list. This book explains (through a huge number of examples) just how dependent we are on computer technology and what can happen when it fails.

[Rivest, Shamir, and Adleman 1978] Ron Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM* 21(2), February 1978, pp. 120–126.

RSA.

[Rubin 1999] Avi Rubin. *The Whitehat Security Arsenal: Tackling the Threats*. Addison-Wesley, Reading, MA, 1999.

A good-guy book describing computer security basics. This book even has white hats on its cover.

[Schmid and Ghosh 1999] Matt Schmid and Anup Ghosh. "An Approach to Testing COTS Software for Robustness to Operating System Exceptions and Errors," *1999 International Symposium on Software Reliability Engineering,* Boca Raton, FL, IEEE Reliability Society, November 1–4, 1999.

Software fault injection for COTS software.

[Schneider 1998] Fred Schneider, ed. *Trust in Cyberspace*. National Academy Press, Washington, DC, 1998.

Why computer security is essential.

[Schneier 1996] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1996.

Applied cryptography explained in layman's terms.

[Schneier 2000] Bruce Schneier. *Secrets and Lies*. John Wiley & Sons, New York, 2000.

A great read, this book is pithy and fun. Need some stories to scare the pants off of upper management? Try this book.

[Thompson 1984] Ken Thompson. "Reflections on Trusting Trust," *Communications of the ACM* 27(8), August 1984, pp. 761–763. <http://www.acm.org/classics/sep95/>

This classic paper goes well with Saltzer and Schroeder's work on security principles. Once again, a paper that everyone cites and all too few read. Should you trust your C compiler? Probably not.

[Whittaker 2002] James Whittaker. *How to Break Software: A Practical Guide to Testing*. Addison-Wesley, Boston, MA, 2002.

Whittaker's first simple book on software testing. A good short read filled with compelling ideas.

[Whitten 1999] Alma Whitten. "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Eighth USENIX Security Symposium,* Washington, DC, USENIX Association, 1999, pp. 169–183.

A great paper on usability (and un-usability) in computer security technology.

[Winkler 1997] Ira Winkler. *Corporate Espionage*. Prima Publishing, 1997.

Winkler's excellent treatment of the insider problem makes this book worth a read.

[Zuse 1991] Horst Zuse. *Software Complexity: Measures and Methods* (Programming Complex Systems, No. 4). Walter de Gruyter, Inc., Berlin, 1991.

The ultimate software metrics tome. Also useful as a doorstopper for the heaviest of doors.

Software Security Puzzle Pieces

As you can see by perusing the annotated references, software security exists at the intersection of several disciplines. The following areas of interest are focal points in the field of software security, both among practitioners and among scientists.

- Reconciling security goals and software goals: software quality management in commercial practice
- Security requirements engineering
- Design for security, software architecture, architectural analysis
- Security analysis, security testing, use of the Common Criteria
- Guiding principles for software security, case studies in design and analysis, pedagogical approaches to teaching security architecture
- Software security education: educating students and commercial developers
- Auditing software: implementation risks, architectural risks, automated tools, technology developments (code scanning, information flow, and so on)
- Common implementation risks: buffer overflows, race conditions, randomness, authentication systems, access control, applied cryptography, trust management
- Application security: protecting code postproduction, commercial technologies
- Survivability and penetration resistance, type safety, dynamic policy enforcement
- Denial-of-service protection for concurrent software
- Penetrate-and-patch as an approach to securing software