

Software Security:

Thought leadership in information security

Gary McGraw

Cigital

21351 Ridgetop Circle, Suite 400

Dulles, VA 20166

gem@cigital.com

A new security paradigm

Computer security takes on more importance as commerce becomes e-commerce and business embraces the Net. However, little progress has been made in the security field, especially when commercially-adopted technology is considered. Popular press coverage of computer security orbits around basic technology issues such as what firewalls are, when to use the AES encryption algorithm, which anti-virus product is best, or how the latest email-based attack works. The problem is, many security practitioners don't know what the problem is. Simply put, the problem is bad software.

Internet-enabled software applications, especially custom applications, present the most common security risk encountered today, and are the target of choice for real hackers. Software security is about understanding software induced security risks and how to manage them. Good software security practice leverages good software engineering practice and involves thinking about security early in the software lifecycle, knowing and understand common threats (including language-based flaws and pitfalls), designing for security, and subjecting all software artifacts to thorough objective risk analyses and testing.

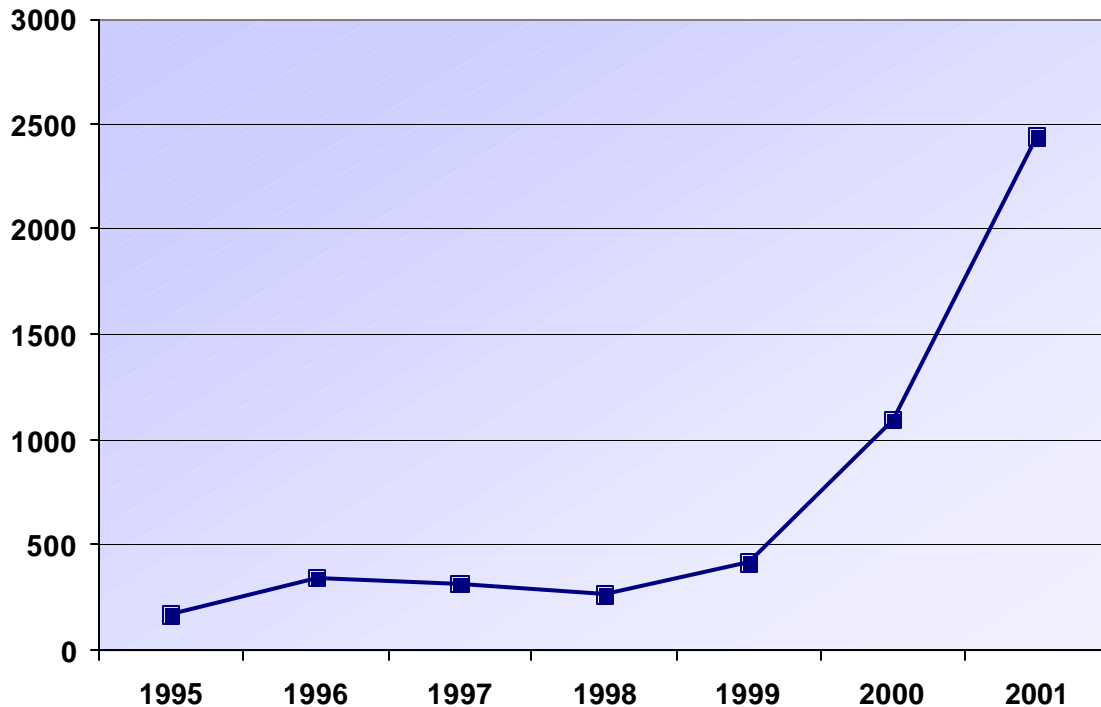


Figure 1: The number of security-related software vulnerabilities reported to CERT/CC over several years.

By any measure, security holes in software are common and the problem is growing. Figure one shows the number of security related software vulnerabilities reported to CERT/CC over the past several years. There is a clear and pressing need for a disciplined approach to software security.

Why the problem is growing

Most modern computing systems are susceptible to software security problems, so why is software security a bigger problem now than in the past? Three trends—together making up the *trinity of trouble*—have a large influence on the growth and evolution of the problem.¹

Networks are Everywhere: The growing connectivity of computers through the Internet has increased both the number of attack vectors, and the ease with which an attack can be made. This puts software at greater risk. More and more computers, ranging from home PCs to systems that control critical infrastructures (e.g., the power grid), are being connected to enterprise networks and to the Internet. Furthermore, people, businesses, and governments are increasingly dependent upon network-enabled communication such as e-mail or Web pages provided by information systems. Unfortunately, as these systems are connected to the Internet, they become vulnerable to software-based attacks from distant sources. An attacker no longer needs physical access to a system to exploit vulnerable software.

Because access through a network does not require human intervention, launching automated attacks is relatively easy. The ubiquity of networking means that there are more software systems to attack, more attacks, and greater risks from poor software security practice than in the past.

Systems are Easily Extensible: A second trend negatively affecting software security is the degree to which systems have become extensible. An extensible host accepts updates or extensions, sometimes referred to as mobile code, so that the functionality of the system can be evolved in an incremental fashion [McGraw and Felten, 1999]. For example, the plug-in architecture of Web browsers makes it easy to install viewer extensions for new document types as needed. Today's operating systems support extensibility through dynamically-loadable device drivers and modules. Today's applications, such as word-processors, e-mail clients, spreadsheets, and Web-browsers support extensibility through scripting, controls, components, and applets. From an economic standpoint, extensible systems are attractive because they provide flexible interfaces that can be adapted through new components. In today's marketplace, it is crucial that software be deployed as rapidly as possible in order to gain market share. Yet the marketplace also demands that applications provide new features with each release. An extensible architecture makes it easy to satisfy both demands by allowing the base application code to be shipped early, and by later shipping feature extensions as needed.

Unfortunately, the very nature of extensible systems makes it hard to prevent software vulnerabilities from slipping in as an unwanted extension. Advanced languages and platforms including Sun Microsystem's Java and Microsoft's .NET Framework are making extensibility commonplace.

System Complexity is Rising: A third trend impacting software security is the unbridled growth in the size and complexity of modern information systems, especially software systems. A desktop system running Windows/NT and associated applications depends upon the proper functioning of the kernel as well as the applications to ensure that vulnerabilities cannot compromise the system. However, XP itself consists of at least forty million lines of code, and end user applications are becoming equally, if not more, complex. When systems become this large, bugs cannot be avoided. Figure 2 shows how the complexity of Windows (measured in lines of code) has grown over the years.

¹ Interestingly, these three general trends are also responsible for the alarming rise of malicious code [McGraw and Morrisett, 2000].

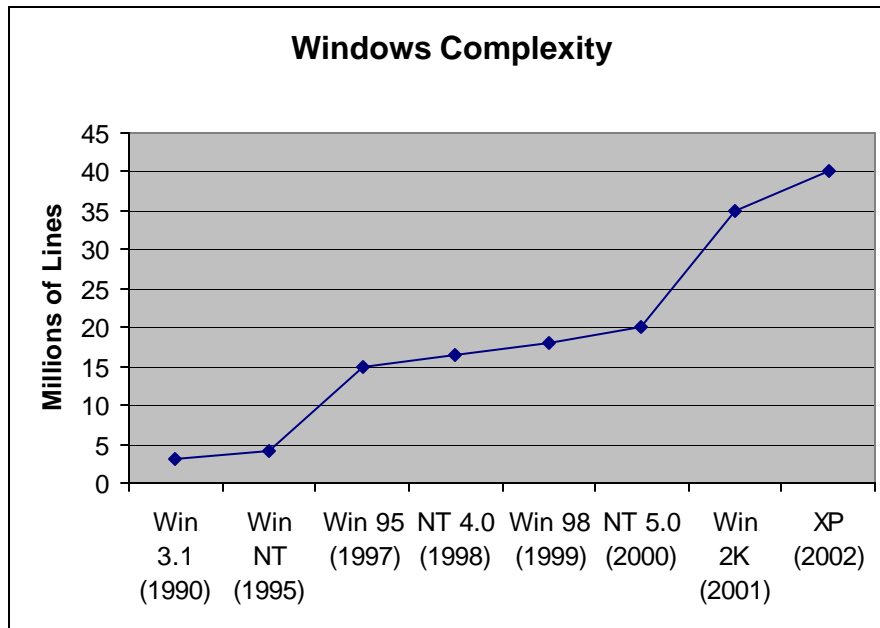


Figure 2: Growth of the Microsoft Operating System code base from 1990 to 2002.

The complexity problem is exacerbated by the use of unsafe programming languages (e.g., C or C++) that do not protect against simple kinds of attacks, such as buffer overflows. In theory, we could analyze and prove that a small program was free of problems, but this task is impossible for even the simplest desktop systems today, much less the enterprise-wide systems used by businesses or governments.

Fleshing out a security discipline

Though some aspects of software security have been discussed for decades in the more general computer security literature, only recently has software security been recognized as a clearly defined sub-discipline. Software security is still in its infancy. The first books in the world devoted solely to the study of software security, *Building Secure Software* (2001) and *Writing Secure Code* (2002), were only recently published. These books have been well received by practitioners, but they only begin to scratch the surface and provide only the roughest outline of a discipline.

A workshop in software security is very likely to have a broad impact on the information security field, helping to define a nascent, but increasingly-important discipline. Results of a workshop are likely to be relevant to academics, software practitioners, software architects, languages researchers, and others interested in proactively avoiding security problems by producing more secure Internet-based code.

Application security versus software security

The software/application security space can be cleanly divided into two distinct subfields. Identifying these subfields is important since the commercial market appears to be blending them together.

Software security is about building secure software. Issues critical to this subfield include: software risk management, programming languages and platforms, auditing software, design for security, security flaws (buffer overflows, race conditions, access control and password problems, randomness, cryptographic errors, and so on), and testing for security. Software security is mostly concerned with designing software to be secure, making sure that software is secure, and educating software developers, architects and users.

Application security is about protecting software and the systems that software runs in a post facto way, after development is complete. Issues critical to this subfield include: sandboxing code, protection against malicious code, locking down executables, monitoring programs (especially their input) as they run, enforcing software use policy with technology, dealing with extensible systems.

Application security follows naturally from a network-centric approach to security, embracing standard approaches such as penetrate and patch [McGraw, 1998] and input filtering, and providing value in a reactive way. Put succinctly, application security is primarily based on finding and/or fixing known security problems only after they have been exploited in fielded systems. Note that this approach addresses security symptoms in a reactive way, ignoring the root cause of the problem. In general, application security takes the same approach to security as firewalls. In fact, application security vendors such as Sanctum, Stratum8, SPI dynamics, Entercept, and Kavado often refer to their products as “application firewalls”.

Software security, the process of designing, building, and testing software for security gets to the heart of the matter by identifying and expunging problems in the software itself. In this way, software security attempts to build software that can withstand attack proactively. Software security follows naturally from software engineering, programming languages, and security engineering.

Areas of Interest in Software Security

The following areas of interest are focal points in the field of software security:

- Reconciling security goals and software goals: software risk management in commercial practice
- Security requirements engineering
- Design for security, software architecture, architectural analysis
- Security analysis, security testing, the role 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, etc)
- Common implementation risks: buffer overflows, race conditions, randomness, authentication systems, access control, applied cryptography, trust management
- Application security: protecting code post production, 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
- Code obfuscation and digital content protection
- Malicious code detection and analysis

Workshop Administration

The areas of interest listed above can be used in a call for participation in a workshop and can help in the creation of an Agenda. Prospective workshop attendees who wish to present or lead a discussion on a particular topic will be asked to submit a short position paper. Workshop attendance will be open to the public.

Organizing Committee

Gary McGraw, CTO of Cigital (a leader in the commercial software security space) will chair the organizing committee and the workshop. Joining McGraw in organizing the workshop are a committee of three: Ed Felten, Professor of Computer Science at Princeton University; Virgil Gligor, Professor of Computer Science at the University of Maryland; and Dave Wagner, Professor of Computer Science at University of California at Berkeley. All committee members have agreed to serve and have participated in the creation of this document. The organizing committee will decide the final Agenda.

Timing: January 2003

The organizing committee would like to schedule the workshop for the days of January 6th and 7th 2003. The software security community (loose affiliation though it is) has been spontaneously discussing the formation of a workshop. DIMACS would do well to sponsor and enable the first workshop in the world devoted entirely to software security. The time to do this is now. Unless this workshop is held soon, someone else will organize a workshop along similar lines, co-opting current momentum in the field.

Potential Attendees

A number of scientists from academia and commercial labs have either worked directly in software security or have expressed direct interest in software security. We provide the following list of potential attendees who we will invite to participate in the workshop.

Martin Abadi, UC Santa Cruz	Trent Jaeger, IBM Research	Jim Roskind, AOL
Andrew Appel, Princeton	Dick Kemmerer, UCSB	Avi Rubin, JHU
Bill Arbaugh, UMD	Brian Kernighan, Princeton	David Sands, Chalmers
Dirk Balfanz, Xerox Parc	Larry Koved, IBM Research	Fred Schneider, Cornell
Steve Bellovin, AT&T	Carl Landwehr, NSF	Bruce Schneier, Counterpane
Matt Bishop, UC Davis	Ben Laurie, openssl	Jonathan Shapiro, Johns Hopkins
Sekar Chandrasekaran, IDA	Peter Lee, CMU	Mike Schroeder, Microsoft
Bill Cheswick, Lumeta	Steve Lipner, Microsoft	Christian Skalka, Johns Hopkins
Cripsin Cowan, wirex	Spiros Mancoridis, Drexel	Geoffrey Smith, Florida International University
Manuvir Das, Microsoft Research	John McHugh, CERT	Scott Smith, Johns Hopkins
Drew Dean, SRI	Cathy Meadows, NRL	Gene Spafford, Purdue
Theo DeRaadt, OpenBSD	John Mitchell, Stanford	John Steven, Cigital
Dawson Engler, Stanford	Greg Morrisett, Cornell	Dennis Volpano, Naval Postgraduate School
Dave Evans, UVa	Andrew Myers, Cornell	David Walker, Princeton
Dan Geer, @stake	George Nacula, UC Berkeley	Dan Wallach, Rice
Anup Ghosh, DARPA	Jens Palberg, Purdue	
Dave Hanson, Microsoft	Rob Pike, Lucent	
Greg Hoglund, Cenizc	John Pincus, Microsoft	
Mike Howard, Microsoft	Francois Pottier, INRIA	
	Marcus Ranum, indy	

A Call to Arms

Computer security is a vast topic that is becoming more important because the world is becoming highly interconnected, with networks being used to carry out critical transactions. The environment in which machines must survive has changed radically since the popularization of the Internet. The root of most security problems is software that fails in unexpected ways. Though software security as a field has much maturing to do, it has much to offer to those practitioners interested in striking at the heart of security problems.

Software practitioners are only now becoming aware of software security as an issue. Plenty of work remains to be done in software security. The most pressing current need involves understanding architectural-level risks and flaws (something the workshop must address). Organizing the nascent software security field with a seminal workshop will help combat the rampant growth of the software security problem by suggest potential solutions and research directions. Such work will deeply impact information security.

Bibliography

What follows is a non-comprehensive bibliography of software security publications. One task from the workshop can be to flesh out this bibliography and gather all relevant papers in electronic form.

[Ashcraft and Engler, 2002] K. Ashcraft and D. Engler. Using programmer-written compiler extensions to catch security holes. In *Proceedings of the IEEE Security and Privacy Conference*, May 2002.

[Arbaugh, Fithen and McHugh, 2000] Bill Arbaugh, Bill Fithen, and John McHugh. Windows of Vulnerability: A Case Study Analysis. *IEEE Computer*, December 2000.

[Bisbey and Hollignsworth, 1978] R. Bisbey II and D. Hollignsworth. Protection analysis project final report. Technical Report ISI/RR-78-13, DTIC AD A056816, USC/Information Sciences Institute, May 1978.

[Bishop and Dilger, 1996] Matt Bishop and Mike Dilger. Checking for Race Conditions in File Access, *Computing Systems*, 9(2):131-152. 1996.

[Bush, et al., 2000] William R. Bush, Jonathan D. Pincus, David J. Sielaff. A static analyzer for finding dynamic programming errors. In *Software Practice and Experience*, 30(7), June 2000.

[Brooks, 1995] Frederick Brooks, Jr. *The Mythical Man-Month: Essays On Software Engineering* (2nd Edition). Addison-Wesley. 1995.

[Cowan, 1998] C. Cowan, C. Pu, D. Maier, H. Hinton, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proceedings of the Seventh USENIX Security Symposium*, pages 63-77, San Antonio, TX, 1998.

[Engler, et al, 2000] Dawson Engler, Benjamin Chelf, Andy Chou, and Seth Hallem. Checking system rules using system-specific, programmer-written compiler extensions. In *Proceedings of the Symposium on Operating System Design & Implementation (OSDI)*, October 2000.

[Howard and Lablanc, 2002] Michael Howard and David Lablanc. *Writing Secure Code*. Microsoft Press. Redmond, WA, 2002.

[Larochelle and Evans, 2001] David Larochelle and David Evans. Statically detecting likely buffer overflow vulnerabilities. In *Proceedings of the USENIX Security Symposium*. August 2001.

[McGraw, 1998] Gary McGraw. Testing for Security During Development: Why we should scrap penetrate-and-patch. *IEEE Aerospace and Electronic Systems*, 13(4), pages 13-15, April 1998.

[McGraw, 1999] Gary McGraw, Software Assurance for Security. *IEEE Computer*, 32(4), April 1999.

[McGraw and Felten, 1999] Gary McGraw and Edward Felten. *Securing Java: Getting down to business with mobile code*. John Wiley & Sons. New York, NY, 1999. See <http://www.securingsjava.com/>.

[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.

[Miller et al, 1990] B.P. Miller, L. Fredricksen, and B. So. An empirical study of the reliability of Unix utilities. *Communications of the ACM*, 33(12):32-44, December 1990.

[Miller, et al, 1995] B.P. Miller, D. Koski, C.P. Lee, V. Maganty, R. Murphy, A. Natarajan, and J. Steidl. Fuzz revisited: a re-examination of the reliability of Unix utilities and services. Technical Report Tech. report CS-TR-95-1268, U. Wisconsin, April 1995.

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

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

[Viega and McGraw, 2001] John Viega and Gary McGraw, *Building Secure Software*. Addison-Wesley. New York, 2001. See <http://www.buildingsecuresoftware.com/>.

[Wagner, et al., 2000] D. Wagner, J. Foster, E. Brewer, and A. Aiken. A first step towards automated detection of buffer over-run vulnerabilities. In *Proceedings of the Year 2000 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, 2000.

[Weber, Shah, and Ren, 2001] M. Weber, V. Shah, and C. Ren. A Case Study in Detecting Security Vulnerabilities using Constraint Optimization. *Proceedings of SCAM*. Venice, Italy, 2001.