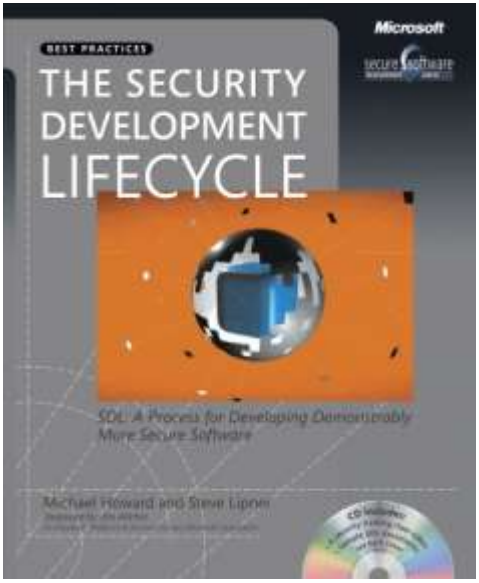

The Security Development Lifecycle

Michael Howard and Steve Lipner

To learn more about this book, visit Microsoft Learning at http://www.microsoft.com/MSPress/books/8753.aspx

9780735622142
Publication Date: May 2006

Microsoft Press

Table of Contents

Chapter 1

Enough Is Enough: The Threats Have Changed

In this chapter:

The adage "Necessity is the mother of invention" sums up the birth of the Security Development Lifecycle (SDL) at Microsoft. Under the banner of Trustworthy Computing (Microsoft 2002), Microsoft heard the call from customers requiring more secure software from their software vendors and changed its software development process to accommodate customers' pressing security needs and, frankly, to preserve the company's credibility. This book explains that process in detail with the simple goal of helping you update your present software development process to build more secure software.

The first question that probably comes to mind is, "Why bother with security?" The answer is simple: the world is more connected now than it has ever been, and no doubt it will become even more connected over time. This incredible level of interconnectedness has created a huge threat environment and, hence, hugely escalated risk for all software users. The halcyon days of defacing Web sites for fun and fame are still with us to an extent, but the major and most dangerous attacks are now upon us: cybercrime has arrived. What makes these attacks so dangerous is that the cybercriminal can attack and exploit his target system silently without creating any obvious sign of a break-in. Now, the criminal can access private or sensitive data or use a compromised system for further attacks on other users, as in the cases of phishing (APWG 2006) and extortion.

The cost-benefit ratio for a criminal is defined by Clark and Davis (Clark and Davis 1995) as

$$M_b + P_b > O_{cp} + O_{cm}P_aP_c$$

where

- M_b is the monetary benefit for the attacker.
- P_b is the psychological benefit for the attacker.
- O_{cp} is the cost of committing the crime.
- O_{cm} is the monetary costs of conviction for the attacker (future lost opportunities and legal costs).
- P_a is the probability of being apprehended and arrested.
- P_c is the probability of conviction for the attacker.

If the left side of the equation is greater than the right side, the benefit of an attack outweighs the costs and a crime could ensue. Of course, this does not imply that all people will commit a crime given enough opportunity! Remember the old model of 10:80:10: 10 percent of people would never commit a crime, no matter what; 80 percent are opportunists; and 10 percent can't be deterred, no matter what. By raising the probability of getting caught and lowering the chance of success, you deter the 80 percent and make the task harder for the "evil 10."

The software development industry cannot easily control P_a or P_c, although the industry can work with the law-enforcement community to provide information that helps apprehend criminals. However, some countries have no cybercrime laws.

Users and administrators of computer systems could control M_b a little by not storing data of value to the attacker, but this solution is infeasible because much of the benefit of using computers is that they allow businesses to operate more efficiently, and that means storing and manipulating data of value to both the customer and the attacker. A well-designed and secure system will increase O_{cp}, making it expensive for an attacker to successfully mount an attack and motivating the attacker to move on to softer targets at other IP addresses.

From an Internet attacker's perspective, the element that influences this equation the most is P_a because the chance of being found and apprehended is too often very small. Admittedly, some miscreants have been apprehended (FBI 2005, CNN 2003), but most attacks are anonymous and go unnoticed by users and system administrators alike. In fact, the most insidious form of attack is the one that goes unnoticed.

As operating system vendors have focused on shoring up core operating system security, cybercriminals have simply moved to more fertile ground higher in the application stack (eWeek 2004)—such as databases (ZDNet 2006a), antivirus software (InformationWeek 2005), and backup software (ZDNet 2006b)—because there is a better chance of a successful attack and the reward is worth the effort. Attacking an operating system does not directly yield valuable data for a criminal, but attacking a database, a customer relationship management (CRM) tool, a health-care system, or a system management tool is like winning the lottery and is reflected in the O_{cm} variable in the equation previously mentioned. It doesn't matter how big or small your software or your company might appear to be; if the attacker

thinks it's worth the effort, and the gains are many and the risk is low, then any insecure application you use might very well come under attack (Computerworld 2006).

Microsoft products are not the only targets of attack. That's why we wrote this book. A cursory glance at any security-bug tracking database will show that every platform and every product includes security bugs (OSVDB 2006a, OSVDB 2006b, OSVDB 2006c, OSVDB 2006d).

Furthermore, the skill required to reverse-engineer security updates (Flake 2004) and build exploitations is easier than ever (Moore 2006). As Mary Ann Davidson, Oracle Corporation's chief security officer, points out:

> *You don't have to be technically sophisticated to be a hacker anymore. Hacking isn't just for bragging rights and chest thumping. There's real money in it. (eWeek 2005)*

The implications of attacks on applications rather than on operating systems cannot be underestimated. Most software vendors build business-productivity or end-user applications, not operating system components. And most security-savvy administrators have focused their limited time on securing their operating system installations and putting network-level defenses such as firewalls in place.

One could argue that the software industry focused almost exclusively on securing operating systems when it should have considered the security of applications with just as much effort. One could also argue that the reason attackers are targeting applications is because the operating system vendors have, on the whole, done a reasonable job of securing the base operating systems in common use. Remember, everything is relative—we said "reasonable," not "good"—but regarding application security, most operating systems are in a better security state than applications.

To compound the problem, many application vendors are dangerously unaware of the real security issues facing customers (CNN 2002), which has led to a false sense of security within the user community and a lack of urgency within the vendor community. Many users and vendors see security as an operating system problem or a network perimeter and firewall problem, but it has become obvious that this is simply untrue.

In short, if you build software, and your software can be accessed by potentially malicious users inside or outside the firewall, the application will come under attack. But this alone is not a sufficient reason to consider security in the development life cycle. The following sections address additional considerations.

Worlds of Security and Privacy Collide

For security to be accepted within an organization, and for software developers to take security seriously, security must accommodate, or at least acknowledge, business needs and business problems. To be successful in an organization, secure software development requires a business benefit. In the case of Microsoft, the business benefit was pretty obvious—our customers demanded more secure software.

But for some people, the decision to make software more secure appears to be not so simple. This is where privacy enters the picture. Trying to sell security to project managers and to upper management can be difficult because there is little, if any, demonstrable return on investment (ROI) data for employing secure development practices. Frankly, upper management is tired of nebulous "we could be attacked" stories that are used to gain budget for security. This is often not a productive way to sell security. But privacy is another matter altogether. People understand what privacy is and what it means when personal, confidential, or personally identifiable information is leaked to miscreants. When users think of security, most often they think about credit card information or online banking passwords being stolen. This, to be pedantic, is not security; it is privacy. Administrators, Chief Information Officers (CIOs), and Chief Information Security Officers (CISOs) should think in terms of risk to business-critical data. Privacy plays a big part in risk calculations.

Privacy and Security

Many people see privacy and security as different views of the same issue. However, privacy can be seen as a way of complying with policy and security as a way of enforcing policy. Restrooms are a good analogy of this concept. The sign on a restroom door indicates the policy for who should enter the restroom, but no security prevents anyone who might want to enter. Adding a lock to the door would provide security to help enforce the privacy policy.

Note Privacy's focus is compliance with regulatory requirements (Security Innovation 2006), corporate policy, and customer expectations.

Risk managers try to put a monetary value on risk. If, according to risk management, the value of protected data if exposed to attackers is, say, $10,000,000, it probably makes sense to spend the $200,000 needed by the development team to remove all known design and coding issues and to add other defenses to protect against such attacks.

Note Risk management can assign a monetary value to the risk of disclosing data.

A security bug such as a SQL injection bug (Howard, LeBlanc, and Viega 2005) is a serious problem to have in your Web-based service-oriented application, but potential privacy issues are what make this class of bug so grave. A SQL injection bug allows an attacker to wreak havoc on an underlying database, including corrupting information and viewing sensitive data. In some instances, a SQL injection attack can be a steppingstone to complete takeover of a network (Johansson 2005). SQL injection vulnerabilities are reasonably

common in database applications, but some have been discovered in database engines also (Red Database 2006).

SQL injection issues are not the only form of security bug that has privacy ramifications. Any bug that allows an attacker to run code of his bidding can potentially lead to privacy violations. Examples include some forms of buffer overflows, command-injection bugs, integer arithmetic issues, and cross-site scripting bugs. But more subtle issues that do not allow arbitrary code execution, such as cryptographic weaknesses and data leakage faults, can lead to privacy issues also.

Warning Much noise has been made about not running as an administrator or root account when operating a computer. We authors are vocal commentators about this issue, and this has helped force fundamental changes in Microsoft Windows Vista; users are, by default, ordinary users and not administrators. Even members of the local Administrators group are users until they are elevated to perform administrative tasks. Running as a normal user does indeed provide security benefits, but it may provide only a limited benefit for privacy protection. Malicious code running as the user can still access any sensitive data that can be read by the user.

The ability of an unauthorized person to view data, an information disclosure threat, can be a privacy issue. In some countries and United States, it could lead to legal action under U.S. state or federal or international privacy laws and industry-specific regulations.

In short, privacy is a huge driver for employing effective security measures and making applications secure from attack. Security is not the same as privacy, but effective security is a prerequisite for protecting the privacy of data about employees and customers.

It's also important to remember that, in some cases, security and privacy can be diametrically opposed to one another. For example, good authentication is a venerable and effective security defense, but it can also raise a privacy issue. Anytime you provide your identity to a computer system, any tasks you perform or any resources you access while you are logged on can be collected and used to model your computer habits. One way to defend against this is to not authenticate, but that's hardly secure.

A good example of privacy and security colliding is the design of Google Desktop version 3 beta. This product allows a user to upload his or her potentially personal or private documents to Google's servers. This design prompted a Gartner analyst to warn that the product posed an "unacceptable security risk" (ZDNet 2006c). It may seem like we're splitting hairs, but this is not a security risk; it's a privacy risk. It's very easy to mix the two concepts. Note that *Time* magazine ran a cover story on February 20, 2006, with the headline "Can We Trust Google with Our Secrets?" (Time 2006). Privacy issues can quickly yield negative headlines.

But wait, there's more!

Another Factor That Influences Security: Reliability

Additional aspects to consider are service-level agreements with your customers and maintaining uptime. Crashed or unresponsive software will probably not satisfy customers or meet their needs. Just as privacy and security are not the same, security is not the same as reliability. But like privacy and security, reliability and security share some goals. For example, any security mitigation that protects against denial of service (DoS) attacks is also a reliability feature.

However, like security and privacy, security and reliability can be at odds. Take a critical server on a protected network as an example. Once the computer's security audit log is full, the machine can no longer log security events, which means that an attacker has a window of opportunity to access sensitive resources on the computer without being audited. In this example, it is not unheard of to simply cause the computer to stop functioning *on purpose* when the audit log is full. For example, the U.S. government protection profiles (NIAP 2005) for evaluating the security of operating systems require the availability of the CrashOnAuditFail option in Microsoft Windows (Microsoft 2003). When this option is set, the computer will crash if the security log is full or if a security-related audit entry cannot be written successfully. Clearly, this is a reliability concern, but it's a valid security defense for some customers. In fact, in some legal-compliance scenarios, you might have no alternative but to crash a computer if auditing can no longer continue.

Another example of security working at odds with reliability is the ability of Windows to automatically restart a service if the service fails. This is an excellent reliability feature, but if it is configured incorrectly, it could be a security issue. Imagine that a service has a bad security vulnerability, like a buffer overrun, and an attacker attempts to compromise a system by exploiting the buffer overrun. If the attacker gets the attack wrong on the first attempt, the service crashes, and then, depending on the configuration, the service might restart. The restart would give the attacker another chance to get the attack right. Every time he gets it wrong, the service crashes and restarts!

Figure 1-1 shows the service recovery configuration dialog box for the print spooler. Configuration options present a tradeoff between security and reliability. The application can crash only twice in one day: if it crashes again, it will not restart. Also, there is a delay of one minute before the service starts up again. This will slow down an attacker substantially.

Many common security coding bugs and design errors can lead to reliability issues such as some forms of buffer overrun, integer arithmetic bugs, memory exhaustion, referencing invalid memory, or array bounds errors. All of these issues have forced software developers to create security updates, but they are reliability issues, too. In fact, the OpenBSD project refers to some of its security bugs as reliability bugs, although other vendors would call the fix a security fix. One such example is a bug fixed by OpenBSD in the BIND DNS daemon in late 2004 (OpenBSD 2004). This is clearly a DoS bug that most vendors would treat as a security fix, but OpenBSD treats it as a reliability fix. Technically, the OpenBSD team is correct, but no major OS vendor differentiates between reliability and security fixes.

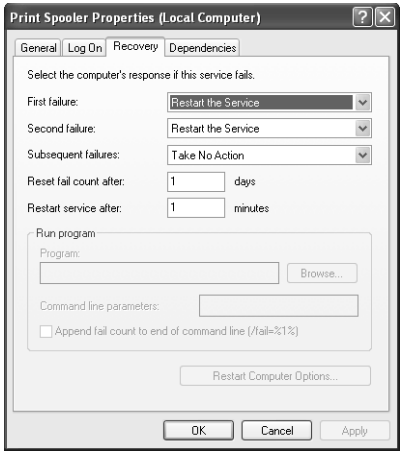

Figure 1-1 Microsoft Windows XP service recovery configuration dialog box.

Note Microsoft's Trustworthy Computing initiative has four pillars. Three of them are technical, addressing the issues we have discussed so far: Security, Privacy, and Reliability. The selection of these three technical pillars is not accidental. (For completeness, the fourth pillar is Business Practices.)

Figure 1-2 shows the results of an analysis of security bugs that were assigned a CVE number by Common Vulnerabilities and Exposures (CVE 2006) between 2002 and 2004. The authors analyzed the CVE bug categories (CVE 2005) to determine whether they had security, privacy, or reliability ramifications. Over this three-year period, CVE created entries for 3,595 security bugs from all corners of the software industry. Notice that the sum is greater than 3,595 because some bugs are both privacy and reliability issues.

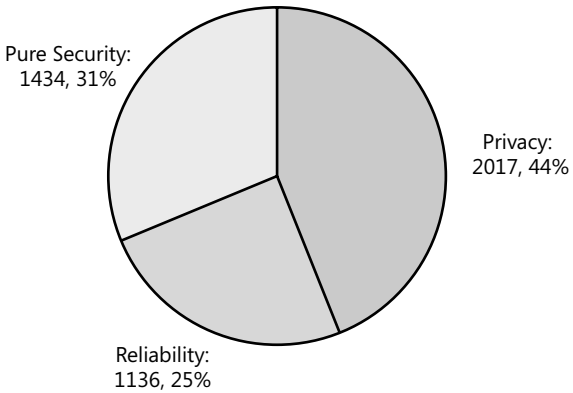

Figure 1-2 Analysis of CVE statistics showing a breakdown of security, privacy, and reliability issues. All the bugs are security bugs, but some also have privacy or reliability consequences, or both.

It's Really About Quality

Ultimately, all the issues we have mentioned are quality bugs. Figure 1-3 shows the relationship among quality, security, privacy, and reliability.

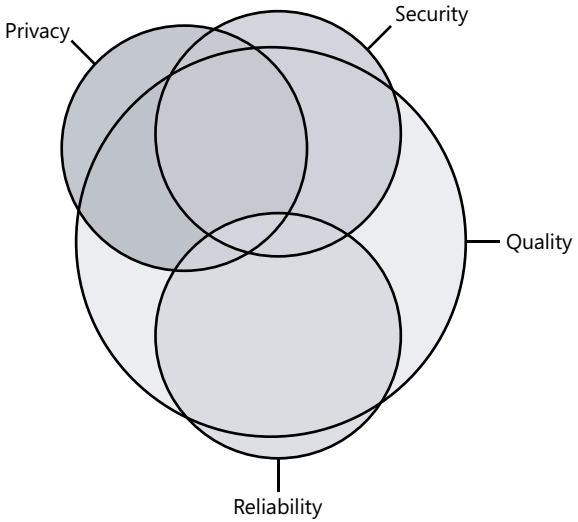

Figure 1-3 The relationship among quality, privacy, security, and reliability.

It is worth mentioning that some elements overlap, as noted in our description of the CVE analysis. Overlap can occur in the following combinations:

- **Security and privacy** Examples include mitigation of privacy issues using encryption, which is a security technology.
- **Security and reliability** For example, a DoS threat is also a reliability issue.
- **Reliability and privacy** For example, an application might crash or otherwise fail, yielding sensitive information in an error message. This is also a security issue.

You'll also notice that portions of the privacy, security, and reliability elements extend beyond the quality circle:

- **Security** If a user invites malicious software onto the computer, this is a security problem but not a security-quality issue.
- **Privacy** If a user willingly divulges personal data to an untrustworthy attacker, through a phishing attack for example, this is not a privacy-quality issue.
- **Reliability** If a person trips over and pulls out a computer's power cable, this is not a software reliability-quality issue.

What we're trying to say is that security should not be considered an isolated endeavor. Only when you start to think about security holistically—as the intersection of privacy, reliability, and quality—does it start to make business-value sense. At that point, you can better sell secure-software improvements to upper management.

Important Security bugs that lead to disclosure of sensitive, confidential, or personally identifiable data are privacy issues and can have legal ramifications. Security bugs that lead to reliability issues could mean reduced uptime and failure to meet service-level agreements.

Why Major Software Vendors Should Create More Secure Software

Improving software security should be an easy sell if your software has a significant number of users; the sheer cost of applying security updates makes it worth getting security, privacy, and reliability right early in the process rather than putting the burden on your customers to apply updates. And frankly, if you have a large number of users, every security vulnerability in your product puts many customers at risk of attack—or worse, exploitation—because you will never have 100-percent patch deployment, and a deployment of less than 100 percent means that a large number of users are put at risk.

If your software is a business-critical application, improved security should again be an easy sell because of the business impact of a failed system.

The goal of creating more secure software and reducing customer pain is why Microsoft has adopted SDL. SDL is not free; it costs time, money, and effort to implement. But the upfront benefits far outweigh the cost of revisions, developing and testing security updates, and having customers deploy the updates. Microsoft has received a lot of criticism in the past about the insecurity of some of its products, and this criticism was a major factor in the company's commitment to improve its software development processes. A vocal critic of Microsoft's security problems was John Pescatore of Gartner. In September 2001, Pescatore advised Gartner clients to evaluate the cost of ownership of using Microsoft Internet Information Services (IIS) 5.0 Web server on Internet-facing computers and to seek alternatives if the costs were justified (Gartner 2001). After seeing the progress Microsoft has made since that date, Pescatore has stated, "We actually consider Microsoft to be leading the software [industry] now in improvements in their security development life cycle [SDL]," and "Microsoft is not the punching bag for security anymore" (CRN 2006).

In an interesting (almost perverse) turnaround, the main IIS competitor, Apache on Linux, is now, and has been for some time, the most frequently attacked Web server on the Internet. Not only does Apache on Linux (Secunia 2006a) have more security bugs than IIS 6.0 on Windows (Secunia 2006b), it is attacked and *compromised* more than IIS on Windows (Zone-H 2006). Admittedly, many attacks result from poor server administration and insecure configuration, but system management is a critical part of the security equation. We discuss this issue in more detail in Chapter 10, "Stage 5: Creating Security Documents, Tools, and Best Practices for Customers."

A Challenge to Large ISVs

We challenge *all* independent software vendors, especially those who have more than 100,000 customers, to change their software development processes. Pay close attention to what we say next: If you are not implementing a process similar to SDL, the processes you have now simply do not create more secure products. It's time to admit this and do something about it. Your customers demand it.

At Microsoft, our customers have benefited from a vulnerability reduction of more than 50 percent because of SDL. Admittedly, we still have a great deal of work ahead of us, and we are under no illusion that we're "done" with security. Jim Allchin, copresident of the Platforms and Services Division at Microsoft, stated, "At no time am I saying this system is unbreakable" (CNET 2006).

That said, Microsoft has taken on the challenge, and SDL has galvanized the company to deliver more secure products to customers. You must do likewise, or attackers will smell blood and the competition that offers products that are more secure than yours will take sales from you. Rebuilding customer trust and goodwill will be difficult at best. We say this from painful experience.

Numerous consumers are starting to ask what their vendors are doing to secure their products from attack. What will your answer be?

Why In-House Software Developers Should Create More Secure Software

The main benefits of SDL for in-house developers are reduced privacy and reliability exposure. Yes, there is a pure security benefit, but as we mentioned earlier, the benefits of security to in-house applications are hard to quantify. Privacy has a risk component that senior managers and risk managers understand, and reliability has an uptime and service-level agreement component that managers also understand. Sell security as privacy and reliability, with a security bonus!

Customer-facing e-commerce applications are, of course, high-risk components and should be developed with utmost care.

Why Small Software Developers Should Create More Secure Software

Creating more secure software is a harder sell for smaller companies because even a small amount of security work up front costs time and money. Although "hacking the code" is effective at creating code rapidly, it is also extremely effective at creating bugs.

Smaller development houses often have a lot of personal pride and ego tied up in their code; so look at security as a measure of quality. Most importantly, if you get it right up front, the cost of fixing bugs later diminishes rapidly. Many sources outline the benefits of building better-quality and more secure software early. One such example is in Chapter 9, "Stage 4: Risk Analysis."

It's fair to say that most people don't mind doing hard work; they just hate reworking. Fixing security bugs can be difficult and time consuming. You can pay now and increase the odds that you'll get it right, or you can pay much more later. As a small development house or an individual developer, you probably have little spare time, and implementing more secure software up front saves you time in the long run. Better-quality software means less reworking, which translates into more time to ski, work out, play with the kids, read a good book (not about software!), or go on a date with your significant other. You get the picture. We have observed at Microsoft that having fewer security vulnerabilities also means that there is more time to add useful features that customers want to our products, and this translates into more customers.

Summary

Selling security process improvements to upper management is not easy because security professionals have often focused on vague although troubling potential threats. Security experts are often seen as alarmists in the boardroom. Selling security as a means to mitigate risk—most notably privacy issues that could lead to legal action from affected customers and reliability issues that could lead to violation of service-level agreements and system downtime—is much more plausible and can be assigned monetary value by managers. Risks and potential costs are associated with the privacy issue and with downtime.

Threats have changed, and the security and privacy landscape is not what it was in 2001. Everything is connected today, and criminals are being lured to the online community because that's "where the money is." There is no indication that this trend will abate any time soon.

The software industry's past is littered with security bugs from all software vendors. If our industry is to protect the future and deliver on the vision of Trustworthy Computing, we need to update our processes to provide products that are more secure, more private, and more reliable for customers.

Microsoft has learned from and has adopted the SDL to remedy its past mistakes. You should, too. Microsoft has seen vulnerabilities reduced more than 50 percent because of the SDL. You will, too.

References

(Microsoft 2002) Trustworthy Computing site, *http://www.microsoft.com/mscorp/twc/default.mspx*.

(APWG 2006) Anti-Phishing Working Group, *http://www.antiphishing.org/*.

(Clark and Davis 1995) Clark, J. R., and W. L. Davis. "A Human Capital Perspective on Criminal Careers," *Journal of Applied Business Research*, volume 11, no 3. 1995, pp. 58–64.

(FBI 2005) "FBI Announces Two Arrests in Mytob and Zotob Computer Worm Investigation," *http://www.fbi.gov/pressrel/pressrel05/zotob_release082605.htm*. August 2005.

(CNN 2003) "Teenager arrested in 'Blaster' Internet attack," *http://www.cnn.com/2003/TECH/internet/08/29/worm.arrest/*. August 2003.

(eWeek 2004) "App Developers Need to Redouble Security Efforts," *http://www.eweek.com/article2/0,1759,1663716,00.asp*. September 2004.

(ZDNet 2006a) Ou, George. "Oracle from unbreakable to unpatchable," *http://blogs.zdnet.com/Ou/?p=151&tag=nl.e622*. January 2006.

(InformationWeek 2005) Keizer, Gregg. "Bug Bites McAfee Antivirus," *http://www.informationweek.com/showArticle.jhtml?articleID=175007526*. December 2005.

(ZDNet 2006b) Evers, Joris. "Backup software flaws pose risk," *http://news.zdnet.com/2100-1009_22-6028515.html*. January 2006.

(Computerworld 2006) Vijayan, Jimkumar. "Targeted attacks expected to rise in '06, IBM study says," *http://www.computerworld.com/securitytopics/security/story/0,10801,107992,00.html*. January 2006.

(OSVBD 2006a) Open Source Vulnerability Database. Oracle, *http://www.osvdb.org/searchdb.php?action=search_title&vuln_title=oracle*.

(OSVDB 2006b) Open Source Vulnerability Database. CRM software, *http://www.osvdb.org/searchdb.php?action=search_title&vuln_title=crm*.

(OSVDB 2006c) Open Source Vulnerability Database. Lotus Domino, *http://www.osvdb.org/searchdb.php?action=search_title&vuln_title=lotus+domino*.

(OSVDB 2006d) Open Source Vulnerability Database. Firewalls, *http://www.osvdb.org/searchdb.php?action=search_title&vuln_title=firewall*.

(Flake 2004) Flake, Halvar. "Structural Comparison of Executable Objects," *http://www.sabre-security.com/files/dimva_paper2.pdf*.

(Moore 2006) Moore, H. D. Metasploit Project, *http://www.metasploit.com*.

(eWeek 2005) Fisher, Dennis, and Brian Fonseca. "Data Thefts Reveal Storage Flaws," *http://www.eweek.com/article2/0,1759,1772598,00.asp*. March 2005.

(CNN 2002) Evers, Joris. "Ellison: Oracle remains unbreakable," *http://archives.cnn.com/2002/TECH/industry/01/21/oracle.unbreakable.idg/index.html*. January 2002.

(Security Innovation 2006) Security Innovation, Inc. "Regulatory Compliance Demystified: An Introduction to Compliance for Developers," *http://msdn.microsoft.com/security/default.aspx?pull=/library/en-us/dnsecure/html/regcompliance_demystified.asp*. MSDN, March 2006.

(Howard, LeBlanc, and Viega 2005) Howard, Michael, David LeBlanc, and John Viega. *19 Deadly Sins of Software Development*. New York, NY: McGraw-Hill, 2005. Chapter 4, "SQL Injection."

(Johansson 2005) Johansson, Jesper. "Anatomy of a Hack," *http://www.microsoft.com/australia/events/teched2005/mediacast.aspx*. Microsoft Tech.Ed, 2005).

(Red Database 2006) Red Database Security. "Published Oracle Security Alerts," *http://www.red-database-security.com/advisory/published_alerts.html*.

(ZDNet 2006c) Espiner, Tom. "Google admits Desktop security risk," *http://news.zdnet.co.uk/0,39020330,39253447,00.htm*. February 2006.

(Time 2006) "Can We Trust Google with Our Secrets?" *Time*, February 20, 2006.

(NIAP 2005) National Information Assurance Partnership, National Security Agency. "Protection Profiles," *http://niap.nist.gov/pp/index.html*.

(Microsoft 2003) Microsoft Help and Support. "How To Prevent Auditable Activities When Security Log Is Full," *http://support.microsoft.com/kb/140058/*. Last Review: May 2003.

(OpenBSD 2004) OpenBSD 3.6 release errata & patch list. "002: Reliability Fix," *http://www.openbsd.org/errata36.html*. November 2004.

(CVE 2006) Common Vulnerabilities and Exposures. *http://cve.mitre.org*.

(CVE 2005) Christey, Steven M. "Re: Vulnerability Statistics," *http://seclists.org/lists/webappsec/2005/Jan-Mar/0056.html*. January 2005.

(Gartner 2001) Pescatore, John. "Nimda Worm Shows You Can't Always Patch Fast Enough," *http://www.gartner.com/DisplayDocument?doc_cd=101034*. September 2001.

(CRN 2006) Rooney, Paula. "Is Windows Safer?" *http://www.crn.com/sections/coverstory/coverstory.jhtml;jsessionid=VV1Q351RM5A1YQSNDBOCKH0CJUMEKJVN?articleId=179103240*. February 2006.

(Secunia 2006a) "Vulnerability Report: Apache 2.0.x," *http://secunia.com/product/73/*.

(Secunia 2006b) "Vulnerability Report: Microsoft IIS 6.0," *http://secunia.com/product/1438/*.

(Zone-H 2006) Zone-H, the Internet Thermometer. *http://www.zone-h.org*.

(CNET 2006) Evers, Joris. "Allchin: Buy Vista for the security," *http://news.com.com/Allchin+Buy+Vista+for+the+security/2100-1012_3-6032344.html?tag=st.prev*. January 2006.