

Reducing Internet-Based Intrusions: Effective Security Patch Management

Bill Brykczynski and Robert A. Small, *Software Productivity Consortium*

The Software Productivity Consortium (the Consortium) has been investigating methods for improving and measuring four essential defenses against Internet-based threats: security patch management, system and application hardening, network reconnaissance and enumeration, and tools against malicious software.¹ These defenses increasingly are critical to an organization's information security posture and should be implemented in an effective, systematic, and repeatable fashion.

Senior-level managers or executives should review process measurement data regularly to ensure that these defenses are being performed properly and to provide an objective basis for organizational improvement. This article focuses on lessons learned implementing improvements in the first of these defenses, security patch management, and is derived largely from pilot projects conducted in collaboration with Consortium members.² The need for improved security patch management figured prominently in the recent draft cyber security strategy issued by the White House.³ The practices examined in this article can assist organizations in substantially reducing the risk from Internet-based compromises.

Patch management

Vulnerability management describes the life-cycle issues in managing software application vulnerabilities that security patches can correct or minimize. Such vulnerabilities are due to design or implementation flaws. Patch

management is a part of vulnerability management and begins with a security patch that an organization applies to reduce or eliminate one or more vulnerabilities. In practice, therefore, an organization needs to focus on vulnerability management because the problem exists before the patch is published. The term *software application* includes network applications, user applications, operating systems, and any other applications that might periodically require security patches.

Background

Software applications commonly have security flaws that enable an unauthorized person to have access or capabilities that should not be allowed. Once a flaw becomes public, the vendor usually creates a patch to correct the flaw or prevent it from being exploited. The vendor might publish the patch on its Web site, post it to a security forum, or distribute it to a list of registered users. An organization must apply the patch before an attack to reduce or eliminate the risk associated with the vulnerability.

The authors examine eight key practices intrinsic to effective, systematic, and repeatable patch management and propose performance measures for evaluating it.

The Gartner Group reports that approximately 90 percent of successful Internet-based attacks exploit software applications that were not properly configured or patched.⁴ Successful attacks commonly exploit a vulnerability for which a patch had been available, possibly for months before the attack.⁵ Attacks might be conducted by humans or Internet-based worms and viruses (for example, Nimda and Code Red) that target unpatched systems. Effective patch management ensures that patches are fielded in a timely manner, thereby eliminating vulnerabilities and preventing successful attacks.⁶ However, judging by the extent of reports of compromised systems, patch management's current state of practice needs improvement. Moreover, the number of reported vulnerabilities has been doubling in recent years, as Figure 1 shows.

Despite patch management's importance, practitioners have few available resources to help them implement an effective process. *Patch Management for Dummies* has not been published yet. In addition, books related to information and system security do not address patch management adequately. For example, we have not found an entire chapter on patch management in any book and have found only modest discussion in a few books. Few papers cover this topic, and even fewer discuss process-related issues. Considering that patch management is one of the most important defenses against Internet-based attacks, it is surprising that so little process guidance is available to the practitioner.

The vulnerability timeline

Figure 2 shows a timeline of the generic activities and events in the life cycle of a typical vulnerability. Implementation details vary with each organization. Bruce Schneier refers to this cycle as the "window of vulnerability."⁸ Steve Beattie and his colleagues have analyzed the timing of when to apply patches based on a cost and risk minimization model, to maximize system availability.⁹

The timeline begins at T_0 , when the vendor creates an engineering flaw that eventually leads to the vulnerability. Until the vulnerability is discovered, the risk is insignificant. At T_1 , only the individuals who discovered the vulnerability know about it. The risk goes up as the number of people who know about it increases. At T_2 , as the vulnerability becomes

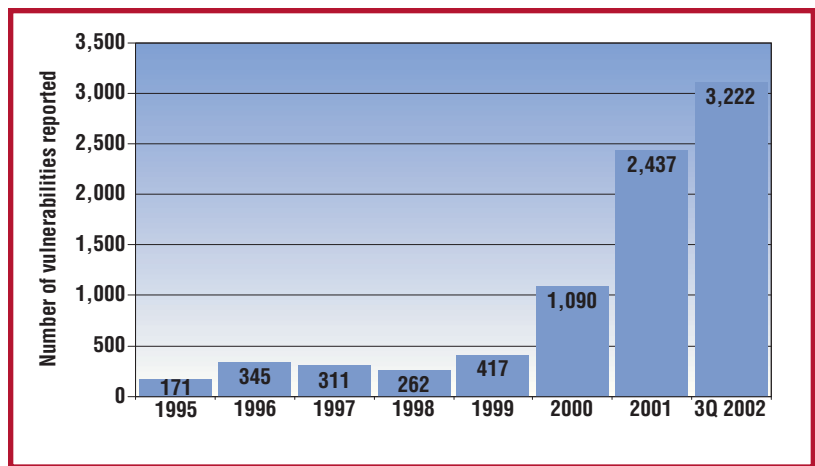


Figure 1. Vulnerabilities from 1995 through the third quarter of 2002.⁷

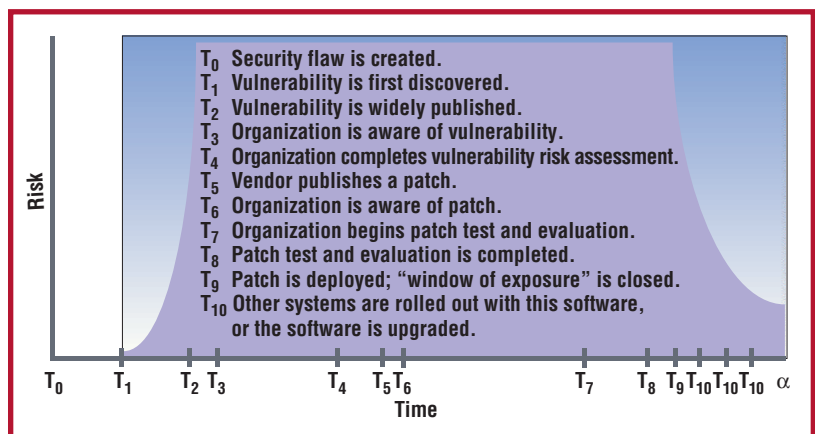


Figure 2. The vulnerability life cycle.

widely known (for example, as user groups, the hacker underground, and vulnerability alert forums publish it on the Internet), the risk increases exponentially.

Soon after this point, hopefully, an organization notes the vulnerability as part of its ongoing threat awareness process. T_3 denotes when the organization becomes aware of the vulnerability. The organization conducts a preliminary risk assessment to determine whether the vulnerability applies to its environment and whether any immediate action is required; this is completed at T_4 . At T_5 , the vendor publishes a patch. T_6 denotes when the organization learns of the patch via its vulnerability alert monitoring process. If the patch is pushed and the organization immediately reads it, T_5 and T_6 are the same. Until this point, the organization can do little against the threat, other than turning off the vulnerable service or application. Once the organization receives the patch at T_6 , more control and options are available.

The detailed test and evaluation of the patch begin at T_7 . The first step in evaluating

We hope the current revision to ISO 17799 will improve upon current patch management requirements.

the patch is to confirm that it applies to the organization's hardware and software environment. Then, the organization tests the patch to ensure that it effectively closes the vulnerability and does not introduce intolerable side effects. T_8 marks the end of patch evaluation.

At T_9 , the organization installs the patch into the production environment, and the immediate window of vulnerability exposure closes. The risk does not go to zero, however, because new systems might be installed without this patch, or future upgrades might not include the patch. This is indicated by the multiple occurrences of T_{10} .

As we show in the section "Measure and improve the process," capturing appropriate data associated with the events and activities for all applicable vulnerabilities provides visibility into patch management's effectiveness.

ISO 17799 and patch management

ISO 17799, *Code of Practice for Information Security Management*, is becoming a widely accepted international standard for best practices and guidance associated with information security management systems.¹⁰ Unfortunately, the standard does not provide adequate guidance for patch management. Control 10.5.2, "Technical Review of Operating System Changes," states that

*Periodically it is necessary to change the operating system, for example, to install a newly supplied software release or patches. When changes occur, the application systems should be reviewed and tested to ensure that there is no adverse impact on operation or security.*¹⁰

This paragraph has two problems. First, it applies only to operating systems and not to other types of software applications in an organization. Second, there are more activities associated with patches than simply reviewing and testing them. For example, an organization should have an effective process for learning about the patches in the first place. This guideline appears to be written from the perspective of regular software maintenance updates, not critical security patches that can prevent Internet-based attacks.

It is unfortunate that ISO 17799 does not adequately address one of the most critical controls to safeguard information assets from Internet-based attacks. We hope the current revision to ISO 17799 will improve upon current patch management requirements.

Key practices

We now examine eight key patch management practices. An organization can institute patch management in a number of different infrastructure models (for example, centralized versus decentralized system administration or heterogeneous versus homogeneous computing environments) and use automated tools to assist with several of the practices. The National Institute of Standards and Technology recommends a vulnerability and patch management process using a hybrid approach of a centralized patch and vulnerability group (PVG) in a facilitator role for the systems administrators.¹¹ Additionally, an organization can perform patch management entirely in house, or it can outsource some or all patch management activities. For US federal agencies, a patch notification and distribution capability is available.¹²

Each key practice begins with a succinct control statement, modeled after controls found in ISO 17799, summarizing the practice's goal. Where applicable, we identify related controls from ISO 17799.

Establish policies, procedures, and responsibilities

Control: The organization should establish objectives, roles, procedures, and responsibilities that relate to patch management.

The objective of establishing policies, procedures, and responsibilities is to ensure that an organization defines, understands, and carries out patch management in a manner that supports its business requirements. Patch management should be a formal part of the organization's information security management system. The organization should define and establish the roles and responsibilities associated with patch management, including roles associated with vulnerability monitoring, alert response, and asset tracking. Additionally, the organization should define the duties associated with the vulnerability-monitoring roles. Who has patch management responsibilities? How often are the vulnerability alert resources monitored? Are the duties transferable so that when someone goes on vacation or is out of the office for the day, someone properly carries out the responsibilities? The organization should establish a measurement system to monitor the effectiveness and efficiency of the patch management process.²

Maintain awareness of IT infrastructure

Control: The organization should maintain a current and complete inventory of the information technology infrastructure.

An inventory of the organization's IT infrastructure is essential for effective patch management. Many organizations have inventory management systems to support financial management requirements. However, these systems might not provide the necessary information or permit the access needed to support patch management without modification.

The organization should classify IT assets by platform hardware type and location (for example, outward-facing server, internal network, desktop, or laptop), and software application records should include vendor, version, and current state of updates (for example, patches and upgrades). It also might be useful to maintain a source for vulnerability information associated with each class of application and identify the staff primarily responsible for administering the application.

Risk potential is another consideration for each asset. The organization should develop a method for determining asset risk, taking into account the asset's relative value. Applications on an outward-facing server that host proprietary data are far more vulnerable than a print server on an internal local area network. The value of the laptop of a "road warrior" is far greater than its undepreciated value; it includes any intellectual property on the computer and the potential for lost productivity. The vendor's market share is another factor to consider, because niche applications tend to have fewer exploited vulnerabilities than those with significant market share. Patch management resources should be allocated partly on the basis of asset risk potential.

A number of automated tools are available to assist with asset tracking, including stand-alone, service-based, and server-based tools.^{13,14} ISO 17799 provides excellent guidance for inventory management; see Control 5.1.1, "Inventory of Assets."

Maintain vulnerability alert resources

Control: The organization should establish and maintain a set of vulnerability alert resources.

Once an organization has identified the software applications to monitor for vulnerabilities, it should develop a *vulnerability*

alert resource list. This list identifies the information resources the organization will use to maintain awareness of vulnerabilities. The list should be based on the inventory of the IT infrastructure and is updated on the basis of changes in the inventory or by discovering new and useful vulnerability alert resources. Resources might include

- Third-party vulnerability alert Web sites such as Bugtraq (<http://online.securityfocus.com/archive/1>), the CERT Coordination Center (www.cert.org), and the SANS (System Administration, Networking, and Security) Institute (www.sans.org/sac)
- Vendor Web sites such as Microsoft, Hewlett-Packard, and Sun
- Fee-based vulnerability alert subscription services such as Counterpane Internet Security and Conexion
- Automated vulnerability-scanning tools such as CNET's Catchup and Symantec's NetRecon

To aid vulnerability alert resources, the Common Vulnerability and Exposure initiative provides an industry-standard list of common names for publicly known vulnerabilities.¹⁵ CVE names make it easier for vulnerability databases and tools to interoperate and for end users to evaluate coverage of these tools.

Additional support for managing vulnerabilities is available from vendors. For example, Microsoft has developed a variety of tools to scan, assess, and update their products. Sun Microsystems has developed the Patch Manager utility to scan Solaris 9 operating systems, identify vulnerabilities, and automatically download and install needed patches. Care must be taken with all such tools, because side effects might result in other applications that interoperate with the newly patched application.

Monitor vulnerability alerts

Control: The organization should obtain and analyze vulnerability alerts in a timely fashion.

Vulnerability alert monitoring aims to review published alerts regularly and identify those that might apply to the organization. Alerts also might include recommended risk reduction measures or vendor patches. Two criteria are particularly important:

Vulnerability alert monitoring aims to review published alerts regularly and identify those that might apply to the organization.

**Ultimately,
someone should
accept the
responsibility
for the residual
risk associated
with addressing
both the
vulnerability
and the patch.**

- Does the alert apply to our environment? If so, at what locations, and who is responsible?
- Does the alert provide new information? If the vulnerability or patch applies to a widely used software application (for example, Microsoft Internet Information Services), this alert will likely be reported in many resources and might be redundant.
- Are exploit tools and scripts available for the vulnerability?
- What steps can you take/have you taken to mitigate the risk?

If no patch is available, preventive measures might include turning off certain services or capabilities, increasing monitoring to detect or thwart actual attacks (for example, via an intrusion detection system), and performing activities designed to raise awareness and reduce an attack's potential damage. If a patch is available, the process moves to the next step (test and evaluate patches).

Differences of opinion might exist about the vulnerability's risk compared to the cost of temporarily removing the service or application. Organizations should develop a protocol for mediating these disagreements. Later, after the organization receives and evaluates the patch, it might decide to not apply the patch because it considers the vulnerability's risk to be low. This is an acceptable result of a rational patch management process. Ultimately, someone should accept the responsibility for the residual risk associated with addressing both the vulnerability and the patch.

For alerts that are sent via email, establishing a dedicated mailbox for their receipt is generally a good idea instead of having them sent to a specific individual. By having alerts sent to a separate mailbox, the organization can easily transfer the responsibility for monitoring them when a staff member is not in the office. Vulnerability-scanning tools can provide supplemental monitoring. The organization should define responsibilities associated with monitoring, including how frequently monitoring occurs (for example, daily or weekly), what actions are necessary, who will review alerts, and what coordination responsibilities are required. Having an accurate software application inventory is essential to ensure identification of potentially relevant alerts.

Assess and respond to vulnerability alerts

Control: The organization should take appropriate and timely action in response to potential vulnerabilities.

Once the organization has identified a potentially applicable vulnerability, it must determine whether that vulnerability is applicable and, if so, what actions (if any) to take to prevent an intrusion. Al Berg recently posed a series of questions that help determine a vulnerability's priority:¹³

- Does the problem affect a technology that's in use in your organization?
- If the vulnerable technology is in use on your network, are you running the problem version (or component)?
- What business resources are at risk?
- Can the vulnerability be exploited remotely?
- What's the potential result of a successful attack?
- How common is the platform being threatened?

Test and evaluate patches

Control: The organization should test and evaluate patches to ensure that they are effective and do not result in intolerable side effects.

Vendors are often under significant pressure to release patches as soon as possible. So, a patch has some likelihood of not addressing the problem adequately and a greater likelihood of having its own side effects. This is especially true when the patch modifies the application environment, not just the application itself. An organization should evaluate a patch in a nonproduction environment that simulates the production environment to test its effectiveness at closing the vulnerability and to assess any side effects. The alternative is to install the patch in a production environment and hope for the best! Testing for side effects is especially important if the patched software interacts with custom software or off-the-shelf software that does not have a large installed base. In that case, there might be nonstandard modes of interaction, or the patch's vendor most likely has not tested these particular interfaces.

If the testing for effectiveness proves satisfactory and the residual risk (after the patch is installed) is sufficiently low, the patch is ready for deployment in the production environment.

Although the conventional wisdom and the literature recommend evaluating patches in nonproduction environments, we've noticed that organizations routinely deploy patches without testing them. Moreover, organizations often consider the resources required for patch testing prohibitive. They often implicitly perform a risk assessment and conclude that

- The risk of installing a patch is low.
- The cost of being able to duplicate their production environment is high.
- The time to configure the environment for each patch evaluation is too long.

As a low-cost alternative, some organizations wait a day or two before installing patches to see what users are reporting in discussion groups. If early reports indicate serious problems, they wait for the patch to be rereleased; otherwise, they proceed.

Install patches

Control: The organization should install patches in a timely fashion, balancing the need for security, resources, and business management.

The two issues in scheduling patch deployment are the urgency of correcting the problem and the degree of interruption to network services that the deployment will create. On the basis of these factors, the organization might install the patch immediately, during the first available time slot that does not interfere with business, or during regularly scheduled system maintenance.

Enterprise information management tools (for example, IBM's Tivoli, Hewlett-Packard's OpenView, and Computer Associates' Unicenter) can aid patch installation.

Measure and improve the process

Control: The organization should instrument the process and establish review mechanisms to ensure that vulnerability management is effective and efficient.

Because the health of the patch management process is frequently critical to organi-

zations, executives or senior-level managers should regularly monitor it. Without measurement, process stakeholders cannot know how well their process is being performed. In patch management, this translates into not knowing whether the organization is effectively managing its resources to protect against Internet-based compromises. Managers should use measurement to set the IT support staff's performance expectations. Executives or senior managers should monitor the status of the vulnerability management process to address questions such as these:

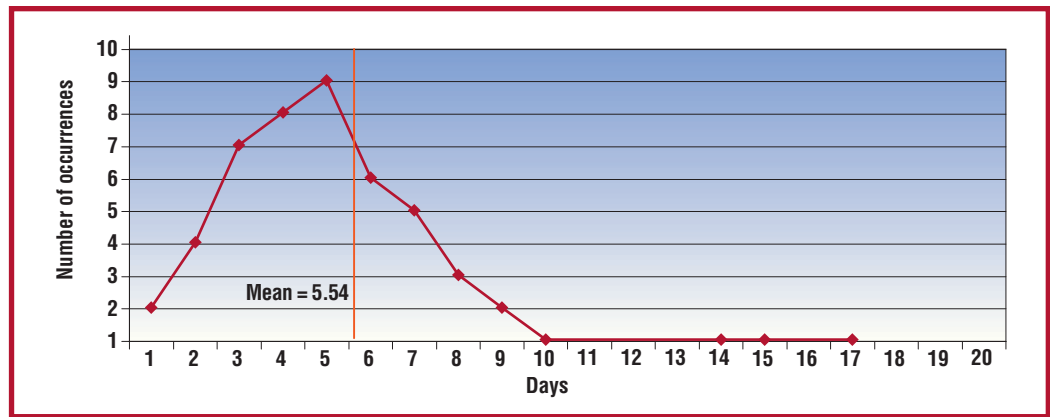
- How healthy is the process? Is the organization responding in a timely fashion to alerts and patches? If not, why not, and what risks does this pose for the organization?
- Are we doing better now than before? By measuring the vulnerability management process, executives and managers can assess the costs and benefits of adjusting the resources.
- What parts of the company are doing better than other parts, and why? This question applies to larger organizations and can help identify and promulgate leading performers' practices.

The following list summarizes several vulnerability management measures, using the timeline in Figure 2:²

1. *Vulnerability awareness delay*, $T_3 - T_2$. This interval measures the effectiveness of the organization's vulnerability awareness process. Failure to perform this key practice could mean the lost chance to take critical risk-mitigating actions before the vendor publishes the patch.
2. *Vulnerability evaluation duration*, $T_4 - T_3$. This interval measures the responsiveness of the organization's evaluation of a vulnerability alert. In some cases, the organization might accept the risk of compromise, in which case no action is required. In other cases, it might either discontinue the effected service until a patch is available or take other risk-mitigating actions.
3. *Vulnerability response time*, $T_4 - T_2$. This interval is the sum of the vulnerability awareness delay and the vulnerability evaluation delay. It is important because it measures the time between when

Because the health of the patch management process is frequently critical to organizations, executives or senior-level managers should regularly monitor it.

Figure 3.
Hypothetical data
for vulnerability
response time.



the vulnerability alert is published (and widely known) and when the organization has decided on a risk-mitigating action (if any) and implemented it.

4. *Patch awareness delay*, $T_6 - T_5$. This interval measures the duration between the vendor publishing the patch and the organization becoming aware of it. This measure is significant because it involves minimum resources and is relatively easy to do well. It would be unfortunate if an application were compromised after the patch was available but before the organization got around to installing it.
5. *Patch evaluation latency*, $T_7 - T_6$. This interval measures the delay, if any, between the organization becoming aware of a patch and beginning the evaluation. If this interval is significant, it suggests an easily solved problem.
6. *Patch evaluation duration*, $T_8 - T_7$. This interval measures the time the organization spends ensuring that the patch effectively closes or reduces the vulnerability and evaluating the patch for side effects. More complex patches or patches to more complex applications will likely require more testing and a longer evaluation. As process experience accumulates, tracking the patch's or application's complexity, as well as the duration, will be important, so that the organization can consider the duration measure in the appropriate context.
7. *Patch implementation duration*, $T_9 - T_5$. This interval measures the period between the organization becoming aware of the patch and implementing the patch fully. This summary measure captures the total time that a solution was available before it was deployed. When reviewing the process performance measures, the organization should investigate outlying data to ensure that it does not need to address any systemic problems.

Our hypothesis is that the vulnerability response time will naturally approximate a Poisson distribution. Figure 3 illustrates this. This hypothetical set of 50 data points illustrates the frequency of occurrence for the vulnerability response time being anywhere from 1 to 17 days.

With measurement data in hand, an organization can assess the performance of its patch management process. In this hypothetical example, the organization is experiencing an average of 5.5 working days between a vulnerability alert being published on the Internet and the organization's decision to protect itself against the threat or accept the risk. An initial heuristic is that cases to the right of the curve's symmetric part bear further investigation. Assume that the organization's chief security officer establishes the goal of reducing the mean vulnerability response time to three days. What process changes does this imply? What additional resources might be required?

Consider these additional questions:


- In absolute terms, is the organization satisfied that its average exposure to vulnerabilities is more than five working days?
- Is the organization satisfied that a third of the cases are handled within 6 to 11 working days?
- Why did the three worst cases take three weeks to resolve?
- What changes (in process, training, and resources) are required to reduce the average to three days?

Ideally, these measurements will be part of an enterprise-wide measurement system, such as a Balanced Scorecard, for example.¹⁶ All measurements have collection costs, so organizations must ensure a direct linkage from the decision-making process to the measurements they make and ensure that they maintain a healthy cost-benefit ratio.

Continuing research

Most security-related vulnerabilities in Figure 1 were due to a handful of common design and implementation mistakes. Avoiding these mistakes requires the application developer's commitment to solving the problem, using education, process discipline, and tools.^{17,18} Research is needed, however, into better understanding the economics of security patch management. Of immediate concern to us are these questions:

- How do you best quantify a particular vulnerability's risk?
- How do you optimize resource allocation throughout patch management?
- How do you rationally determine whether to test security patches for effectiveness and side effects?
- As the infrastructure's size increases, how well does patch management scale? Are there scale thresholds that will require changing the process?

By taking a life-cycle view, we offer specific guidance to organizations on the features of well-engineered patch management. Although the implementation details will vary for each organization, the principles are the same. By incorporating patch management requirements into an information security management system, an organization will establish performance expectations of the staff. Establishing a measurement system will give the stakeholders objective insight into how well the process works. Direct executive involvement shows the organization's commitment to excellence in patch management. 

Acknowledgments

This article is based partly on work sponsored by DARPA Grant MDA972-01-1-0006. The article benefited from technical review by Doron Becker, Mike Polen, and Lou Rose.

References

1. B. Brykczynski and B. Small, *Establishing and Measuring Information Security Policy Conformance*, SPC-2001054-CMC, Software Productivity Consortium, Herndon, Va., Jan. 2002; www.software.org/catalog/products/product.asp?pfid=385.
2. B. Brykczynski and B. Small, *Improving the Security Patch Management Process*, SPC-2002015-CMC, Software Productivity Consortium, Herndon, Va., June 2002; www.software.org/catalog/products/product.asp?pfid=410.
3. *The National Strategy to Secure Cyberspace* (draft), President's Critical Infrastructure Protection Board, Washington, D.C., Sept. 2002; www.whitehouse.gov/pcipb.
4. J.C. Perez, "Gartner: Most IT Security Problems Self-Inflicted," *Computerworld*, 9 Oct. 2001. www.computerworld.com/securitytopics/security/story/0,10801,64605,00.html.
5. W. Arbaugh, W. Fithen, and J. McHugh, "Windows of Vulnerability: A Case Study Analysis," *Computer*, vol. 33, no. 12, Dec. 2000, pp. 52–59.
6. *Keep Operating Systems and Applications Software Up to Date*, CERT Coordination Center, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 2001, www.cert.org/security-improvement/practices/p067.html.
7. "CERT/CC Statistics 1988–2002," CERT Coordination Center, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 2002, www.cert.org/stats/#vulnerabilities.
8. B. Schneier, "Closing the Window of Exposure," Counterpane Internet Security, Cupertino, Calif., 2002, www.counterpane.com/window.html.
9. S. Beattie et. al., "Timing the Application of Security Patches for Optimal Uptime," *Proc. 16th Ann. USENIX Systems Administration Conf. (LISA 02)*, USENIX Assoc., Berkeley, Calif., 2002; www.nxnw.org/~stevel/papers/lisa2002-time-to-patch.pdf.
10. *Code of Practice for Information Security Management*, ISO/IEC 17799:2000, Int'l Organization for Standardization, Geneva, Dec. 2000.
11. P. Mell and M. Tracy, *Procedures for Handling Security Patches—Recommendations of the National Institute of Standards and Technology*, NIST Special Publication 800-40, Nat'l Inst. of Standards and Technology, Gaithersburg, Md., Aug. 2002.
12. E. Messmer, "GSA to Keep Vigilant with Patch-Notification Service," *Network World*, 2 Dec. 2002, www.nwfusion.com/news/2002/1202fedcirc.html.
13. A. Berg, "Feeling Vulnerable?" *Information Security*, Feb. 2002, pp. 42–52; www.infosecurymag.com/2002/feb/features_vulnerable.shtml.
14. S. Sidel and A. Briney, "Patching across the Enterprise," *Information Security*, Feb. 2002, pp. 48–49; www.infosecurymag.com/2002/feb/features_sidebar1.shtml.
15. R. Martin, "Managing Vulnerabilities in Networked Systems," *Computer*, vol. 34, no. 11, Nov. 2001, pp. 32–38.
16. R.S. Kaplan and D.P. Norton, *The Balanced Scorecard: Translating Strategy into Action*, Harvard Business School Press, Cambridge, Mass., 1996.
17. D.A. Wheeler, *Secure Programming for Linux and Unix HOWTO*, 2002, www.dwheeler.com/secure-programs.
18. J. Viega and G. McGraw, *Building Secure Software*, Addison-Wesley, Boston, 2002.

For more information on this or any other computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.

About the Authors



Bill Brykczynski leads the Software Productivity Consortium's security program. His professional and research interests include applied security research and software inspection processes. He received his BS in systems science from the University of West Florida, his MS in information management from George Washington University, and his PhD in information technology from George Mason University. Contact him at the Software Productivity Consortium, 2214 Rock Hill Rd., Herndon, VA 20170-4227; bryk@software.org.

Robert A. Small is a principal member of the technical staff at the Software Productivity Consortium. His areas of interest include applied security research and building communities of practice. He received his BS in economics from Ursinus College and his MS in computer science from the University of New Haven. Contact him at the Software Productivity Consortium, 2214 Rock Hill Rd., Herndon, VA 20170-4227; small@software.org.

