Surviving Insecure IT: Effective Patch Management

Simon Liu, *US National Library of Medicine*
Rick Kuhn, *US National Institute of Standards and Technology*
Hart Rossman, *Science Applications International Corporation*

As we discussed in the last issue ("Introducing Insecure IT"), software developers are beginning to make some headway in reducing IT system vulnerabilities. But just as there will never be impenetrable armor, there will never be invulnerable software. In the battle between attackers and defenders, developers still make mistakes, and adversaries invent new ways to defeat the best safeguards. Consequently, enterprises need an effective patch management mechanism to survive the insecure IT environment. Effective patch management is a systematic and repeatable patch distribution process for closing IT system vulnerabilities in an enterprise. It involves pervasive system updates, including any or all the following: drivers, operating systems, scripts, applications, or data files. Patches usually originate from, and are supported by, IT product vendors. These vendors often use different terminologies for patches—for example, Microsoft has nine different types of patches (security update, critical update, feature pack, hotfix, service pack, software update, update, update rollup, and upgrade; see http://support.microsoft.com/kb/824684).

Patching is necessary for security, but it's difficult to manage systematically. Multiple, often conflicting, priorities must be balanced to minimize disruption to mission-critical systems. In general, effective patch management involves several steps.

Establish Timely and Practical Alerts

Software vendors routinely announce vulnerabilities as they're discovered, but many of these vulnerabilities don't apply to IT systems in an enterprise. A typical organization might have software from hundreds of vendors, so keeping track of announcements can be complicated and time-consuming, making it easy for overworked system administrators to miss a critical vulnerability notification. To reduce the effort required to keep up with announcements, administrators can turn to sites such as the US Computer Emergency Response Team (www.us-cert.gov/federal/) and the National Vulnerability Database (http://nvd.nist.gov). US-CERT analyzes security vulnerabilities, provides information and training, and sends consolidated announcements of new vulnerabilities. The National Vulnerability Database maintains standardized vulnerability data to enable automated vulnerability management and compliance checking.

Monitoring and paring vulnerabilities down to a list of alerts that relate only to an enterprise will make the vulnerability reports more focused, easier to follow, and less likely to be ignored, but this can only be accomplished if a complete and correct inventory of software applications is available. Periodic auditing of applications is thus an essential patch management component.

Receive Notification of Patches or Discover Them

An organization should maintain solid relationships with key IT

vendors that facilitate the timely release and distribution of information on product security issues and patches. These relationships can range from routine contacts with the account manager to simple subscriptions to the vendor's security notification list. Only subscribers to the notification list receive email notifications. Without a subscription, the organization would have to monitor each vendor's Web site for information on the availability and applicability of new patches. Alternatively, they can be obtained from mailing lists or service providers who consolidate patch information, or new patch releases could be observed as part of routine updates.

Download Patches and Documentation

A key patch management component is the intake of the identified patch and any associated documentation from the vendor, which will include the installation procedure. Verifying the patch's source and integrity is important to ensure that it's valid and hasn't been maliciously or accidentally altered. The vetting of information regarding both security issues and patch release is also critical. Enterprises must know which security issues and software updates are relevant to their environments.

Assess and Prioritize Vulnerabilities

Effective security patch management involves balancing multiple priorities to minimize and manage the potential disruption involved in implementing software changes on mission-critical systems. Any IT vulnerability presents some risk, but enterprises can't afford to treat every vulnerability equally. Vulnerabilities must be assessed, classified, and prioritized just like any other IT projects. In 2006, the Forum of Incident Response Teams (FIRST, www.first.org/cvss/) published a model known as the Common Vulnerability Scoring System (CVSS) for structuring vulnerability prioritization.

The CVSS is an open standard designed to provide users with an overall composite score representing a vulnerability's severity and risk. The CVSS itself is derived from metrics in three distinct categories:

- Base metrics contain qualities that are intrinsic to a given vulnerability and don't change over time or in different environments.
- Temporal metrics contain characteristics of a vulnerability that evolve over its lifetime.
- Environmental metrics contain characteristics of a vulnerability that are tied to an implementation in a specific environment.

The CVSS is a useful approach for enterprises to standardize the severity assessment and prioritization of IT vulnerabilities.

Perform Testing

Patches should be tested to ensure that they have no conflicts or incompatibilities before deployment. Two competing aspects often dictate patch testing: thoroughness and timeliness. Enterprise patch testing procedures must balance these competing goals so that testing is thorough enough to essentially rule out any potential issues but not take so long that it impacts the overall integrity of enterprise security by leaving systems unpatched.

The actual mechanics of testing a patch vary widely by organization. Patch testing could be as simple as installing a patch and making sure the system reboots, or as complex as executing a battery of detailed and elaborate test scripts that validate continued system and application functionality. In general, a suitable approach for patch testing is dictated by system criticality and availability requirements, available resources, and patch severity.

Patches should be tested on nonproduction systems because remediation can easily produce unintended consequences. Although the perfect test environment will mirror production as closely as possible, it's important to at least account for the majority of critical applications and supported operating platforms in the patch testing infrastructure.

However, no matter how well the testing environment is configured, minor differences in production systems could present challenges or problems when actually applying the patch. Therefore, rather than unleashing the patch on the entire enterprise, it's wise to conduct pilot testing. Organizations often use a subset of production systems as an ad hoc test environment; department-level servers and IT employee systems are typically used in these cases. Regardless of the available test equipment and systems, exposing the update to as many variations of production-like systems as possible will help ensure a smooth and predictable rollout.

Deploy Patches

Patch deployment is where the real work of applying patches and updates to production systems occurs. The most important technical factor affecting deployment is the choice of methods and tools used. The patching process can be fully automated, semiautomated, or manual, but the degree of automation will depend primarily on the target environment. Automated and semiautomated tools are sometimes free or vendor-specific. For standard Windows desktop operating systems, for instance,

Microsoft's free Windows Server Update Services tool can manage and automate the patching process (http://technet.microsoft.com/en-us/wsus/default.aspx). Vendor-specific tools can manage and automate third-party software patches (such as the Firefox browser and many other common desktop applications).

Patching for other desktop operating systems occurs mostly on an ad hoc basis. Macintosh computers have an automated system update check turned on by default that prompts users to update. Linux desktops often have a manual trigger but can be automated through scheduled jobs. Patching for network devices, servers, Web applications, databases, and other packaged applications is often performed with little automated support and follows a strict change control and testing process due to the potential impact for all users.

It's logical to strive for a consolidated tool strategy wherever possible, but it's important to recognize that only a few vendors offer best-of-breed patching. Although support for multiplatform patching is an emerging requirement for cross-platform patches, it's still challenging to implement. Many vendors offer support for Windows and Linux, as well as some Unix platforms, but enterprises must check references for required platforms, multiplatform compliance reports, and support for scalable environments for PCs and server infrastructures.

Automated updating is an important component of patch management, but automation brings its own set of issues for administrators. Updates during business hours can obviously introduce problems by creating performance loads on PCs when they might be needed most. However, scheduling all updates for 2:00 a.m. isn't a solution either because thousands of machines simultaneously downloading large patches could overload the organization's network connections. Distributing update times across nonbusiness hours seems like a simple solution, but not all applications have the same volume or size of updates: some might have large, frequently released patches, whereas others might require occasional updates. Allocating update times to minimize system load and reduce the risk of disrupting operations requires a careful review of patch frequency, plus knowledge about patch size averages and distributions for enterprise applications. This schedule should also factor in the need to reboot after patch deployment.

Fortunately, planning and scheduling are familiar problems for successful enterprises, but management must ensure that planning skills are applied just as carefully to software patches as they are to core business operations. Various firms now offer automated update scheduling software to assist in this process.

Audit and Assessment

Systematic audit and assessment is critical to gauge the success and extent of patch management efforts. After patch deployment, organizations should verify that they have fixed or mitigated vulnerabilities as intended. They can accomplish this by reviewing patch logs to verify whether the recommended patches were installed properly, conducting follow-up scans, and in some cases conducting penetration tests to make sure their systems aren't vulnerable to the exploit code the patch is designed to thwart.

Despite some progress, the volume of vulnerabilities in most enterprises remains high, yet the amount of time that enterprises have in which to protect their systems against potential vulnerability continues to shrink. Effective patch management is more essential than ever to shore up security vulnerabilities, protect system functionalities, and maintain the stability of the enterprise production environment.

Acknowledgments

We thank Karen Scarfone and Peter Mell at NIST for many helpful comments on an early draft of this article. We identify certain products in this document, but such identification doesn't imply recommendation by the US National Institute of Standards and Technology or other agencies of the US government, nor does it imply that the products identified are necessarily the best available for the purpose.

Simon Liu *is the director of information systems at the US National Library of Medicine. His research interests include IT architecture, cybersecurity, software engineering, and database and data mining. Liu has two doctoral degrees in computer science and higher education administration from George Washington University. Contact him at simon_liu@nlm.nih.gov.*

Rick Kuhn *is a computer scientist at the US National Institute of Standards and Technology. His research interests include information security, software assurance, and empirical studies of software failure. Kuhn has an MS in computer science from the University of Maryland, College Park, and an MBA from William & Mary. Contact him at kuhn@nist.gov*

Hart Rossman *is a vice president and CTO of Science Applications International Corporation. He also serves as a faculty member with the Institute for Applied Network Security. Rossman has a CISSP, a BA in communication from the University of Maryland, College Park, and an MBA from the University of Maryland, Robert H. Smith School of Business. Contact him at hart.m.rossman@saic.com.*