



Patch Management For The Real World— A Manager's Guide

Whitepaper by:
Scott Carpenter
Product Manager

Table of Contents

| | |
|---|-----------|
| Executive Summary | 3 |
| <i>The Problem</i> | <i>3</i> |
| <i>The Solution.....</i> | <i>3</i> |
| The Challenges of Patch Management..... | 3 |
| Implementing a Solution | 5 |
| <i>Secure Executive Buy-In</i> | <i>5</i> |
| <i>Patch Management Is a Process—Not an Event</i> | <i>6</i> |
| Defining the Best Practices..... | 6 |
| <i>Step One: Discovery</i> | <i>7</i> |
| <i>Step Two: Analysis.....</i> | <i>8</i> |
| <i>Step Three: Research and Test</i> | <i>9</i> |
| Research..... | 9 |
| A Few Precautions | 10 |
| Test | 10 |
| Testing using Image-Based Systems..... | 11 |
| Patch Level Minimum Baselines | 11 |
| <i>Step Four – Remediate.....</i> | <i>12</i> |
| Incremental Rollout | 12 |
| Scheduling Reboots | 12 |
| Policy-based Remediation..... | 13 |
| A Few Precautions | 13 |
| <i>Step Five – The Safety Net.....</i> | <i>13</i> |
| <i>Step Six – Reporting.....</i> | <i>14</i> |
| <i>Return to Step One - Close the Loop</i> | <i>14</i> |
| <i>Choosing an effective patch management product</i> | <i>14</i> |
| <i>Conclusion.....</i> | <i>16</i> |
| Useful Links..... | 17 |

Executive Summary

The Problem

Speed, accuracy, and security in sending, receiving, and storing information are critical to business success. When information systems fail or become compromised due to a security breach, losses in time, money, and reputation can be disastrous.

Despite this simple fact, many organizations do not have an effective maintenance plan in place to protect the assets they value so dearly: information and the systems that house it.

The Solution

Solve the problem by tailoring a logical and viable maintenance plan for your specific IT infrastructure. A core component of the plan will be an effective and efficient patch management procedure. This document helps you create a patch management process by following a series of best practices developed and proven in the field. While each environment has a unique set of best practices, this guide defines a general framework around which you can develop the set of practices that will work best for you.

The Challenges of Patch Management

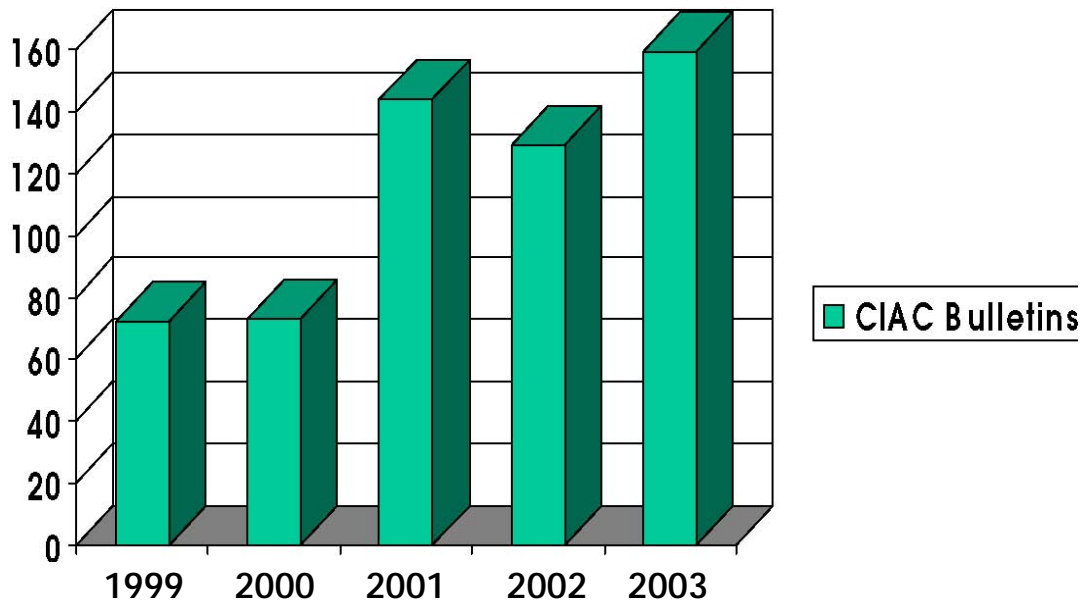
In 2001, when System Administrators were already busy with the day-to-day tasks of running a network, the last thing they needed was yet another job to do. Nevertheless, along came Code Red and Nimda—followed soon enough by Slammer, Sasser, and MyDoom. Patch management is now much more than a server-room buzzword. After Microsoft's sorely needed "Trustworthy Computing" initiative in the fall of 2001, the patch flood began, and has since escalated to a torrent of regularly released patches from Redmond.

SearchSecurity.com and similar journals report that 4,677 new viruses appeared in the first six months of 2004 -- an increase of 21 percent over the same period last year. Similarly, security firm McAfee notes not only an increase in the number of security threats, but also a dramatic climb in those it finds worthy of watching. It counts a 20 percent increase in threats during the first half of 2004 compared to 2003, and has tagged more threats as "Medium" or higher during 2004's first quarter than it did in all of 2003.

It is important to note that Microsoft is not the only vendor to issue patches. Yes, Microsoft has the most widely deployed desktop operating system; however, most

enterprises have a multiplatform server environment. Thus, the need to apply patches consistently and quickly across UNIX, Linux, and other platforms has become apparent. There is also a growing requirement for patch management coverage of database management systems and applications. Today, patching is a process that affects all platforms and applications as increasingly sophisticated hackers exploit more and more security vulnerabilities.

Figure 1 illustrates the number of vulnerabilities reported by the CIAC (Computer Incident Advisories Capabilities) over the last few years and demonstrates the steady increase in the total number of vulnerabilities exposed annually.



Source: CIAC, a division of the U.S. Department of Energy. CIAC provides third-party advisories, bulletins and ratings upon discovery of system vulnerabilities. The graph shows the number of Bulletins and Advisories released by the CIAC between 1999 and 2003. Note that the years run from October of the previous year through September of the labeled year.

In response to the increase in vulnerability identifications, hackers have stepped-up their efforts to write exploitative code as quickly as possible. In the case of the famous SQL Slammer worm (W32.SQLExp.Worm), the internet community had six months between the vulnerability being identified—and a patch being released!—and when the worm was actually released. In the case of Nimda, the lead-time was nearly a year. Subsequently, however, the MS Blaster worm (W32.Blaster.Worm) allowed about a month between discovery and exploit.

The resulting sense of urgency means that vendors often release patches to fix a problem as quickly as possible. They have little time to test a patch fully for compatibility with all configurations. This introduces an element of risk to the patch deployment process.

In short—and this is very important—there is no true way to determine how a patch will effect your environment without actually installing the patch in your environment.

One apparent solution to this problem is for IT professionals to monitor vendors' websites looking for the latest security patches, download them and apply them to the pertinent machines before exploits appear. Most vendors offer notification services that can email users upon release of patches that may pertain to the user's environment.

This approach challenges administrators with:

- Manually downloading patches
- Determining which machines are vulnerable
- Testing the patch to verify compatibility
- Installing needed patches onto the appropriate systems.

Unfortunately, appropriate machines often number in the thousands. Therefore, a manual patching process is impractical.

With so much work involved in patch management, some companies accept the risk of not patching their systems and rely instead on strong perimeter security. Of those who do patch, some patch only their internet-facing systems, such as website and email servers.

Unfortunately, these solutions do not always help. For one thing, relying solely on perimeter security (firewalls, proxy servers, etc.) assumes that perimeter security is flawless, which is not always the case. Viruses are often written specifically to circumvent perimeter security (or sneak through) as in the case of worms and viruses that are spread as either email attachments or embedded within web pages.

Implementing a Solution

The solution to this growing problem is to develop a series of best practices. Although the exact procedures followed in each environment will differ slightly, it is possible to define best practices as a series of guidelines tailored to your environment. Once you have adapted a comprehensive patch policy, automate policy compliance with patch management software.

Secure Executive Buy-In

Sometimes the greatest hurdle to overcome is not a technical one. It is crucial, when undertaking any new project, to have the support of senior management. Making senior managers aware of security risks and the need for patches is important for successfully implementing a patch management program and ensuring that appropriate resources are available.

Executives like numbers and they like money. So here are some attention-grabbing money numbers: the Sasser worm cost businesses worldwide as much as \$500 million. Reportedly, the MyDoom virus is spiraling toward a \$4 billion price tag as it continues to spread despite its age.

For current statistics, browse www.ciac.org or www.sans.org to access all the alarming data needed to justify an investment in patch management.

Patch Management Is a Process—Not an Event

Many companies see patch management as event-driven, something done in response to an outbreak of some kind. For example, during the SQL Slammer outbreak in early 2003, companies scrambled to install patches across their SQL Server farms. Unfortunately, Slammer took all of nine minutes to spread worldwide—not much time to deploy a patch, let alone research and test one. This event-based patching philosophy is akin to fixing the barn door after the Trojan horse has come home. The time to patch a given vulnerability is *before* the exploit arrives. Letting the exploit get through, in many situations, may necessitate a full rebuild of the affected systems.

Therefore, look at Patch Management as a process, ideally a **closed-loop process**. A closed-loop-process includes an automatic control system in which feedback acts to maintain output at a desired level. A fully effective, automated patch management solution maintains desired patch levels with as little human intervention as possible.

Patch management, as an automated series of best practices, recurs regularly on your network to ensure protection from exposed vulnerabilities. Patch management requires the regular re-discovery of any systems susceptible to known vulnerabilities, timely download of patches and patch definition databases, reliable deployment of patches to the systems that need them, and accurate verification of installation.

Defining the Best Practices

What follows describes a six-step method to Patch Management. These six steps, discussed as a closed-loop solution, define an effective framework for patch deployment whether you are bringing an un-patched environment up to a baseline level or deploying a patch as part of an emergency response plan. The six steps are:

- **Discovery** – The discovery phase involves locating assets (workstations and servers) on your network and categorizing them.
- **Analysis** – The analysis process determines current patch levels and sets a minimum baseline policy.
- **Research and Testing** – This phase focuses on researching missing service packs and patches, conducting appropriate risk analyses, and testing patches on lab systems.
- **Remediation** – This involves bringing systems up to date by deploying patches in accordance with the established patch policy.
- **Safety Net** – The safety net, although not always a necessary step, provides the ability to roll back a patch should the need arise.

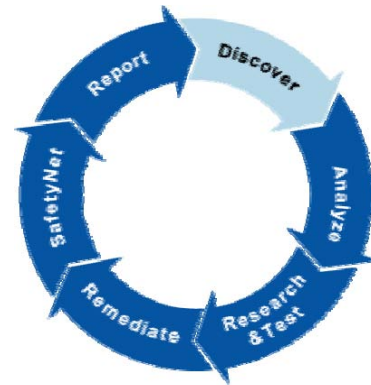


- **Reporting** – Reporting conducts a change review and verifies successful deployment of patches. Comprehensive reporting also includes enough review, analysis, and adjustment to close the loop and begin the cycle again automatically.

The following sections look at each step in detail.

Step One: Discovery

Effective Patch Management begins with defining a starting point. Develop a clear and accurate picture of what your network needs to get your patch situation under control. The first step is to identify and categorize your assets: take full inventory of all workstations and servers on your network. Typical IT environments often include dozens to hundreds of servers and hundreds to thousands of workstations. Locating and documenting each of those systems manually represent an enormous undertaking.



To ease this task, look for a patch management product that scans a network and locates hosts. There should be multiple discovery methods available, from Active Directory computer account location to the IP address and subnet scan, to ensure that the discovery phase is as complete as possible.

Once you identify assets, categorize them based on exposure and risk. By categorizing assets, you develop a picture of which machines require rapid patch management (e.g. within 24 hours) and which require standard management (e.g. within 3 days).

Categorizing your assets is usually a manual process. It is difficult to automate a process that essentially identifies “important machines” and “less-important machines.” When categorizing machines, always consider the information that machine protects. Other issues to consider are public visibility (as in the case of a website) and sensitivity (customer credit card numbers).

The most important question to ask is, “what will be the impact on the company if this machine is down or compromised?”

Risk Analysis should be an integral part of the Patch Management process. Please see the “Useful Links” section at the end of this paper for more information.

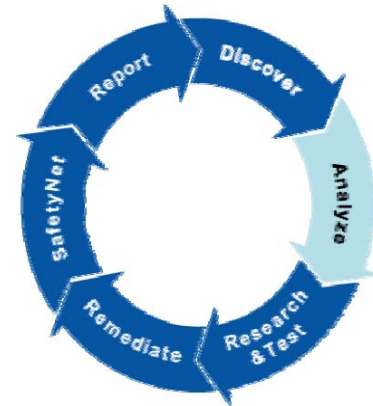
Most patch management tools support system grouping. Look for an application that allows administrators to manually create logical groups of computers based on risk, configuration, department, physical location, or other relevant criteria. Look for a tool that allows you to group systems by the roles they perform: SQL Servers, for example, or IIS boxes, Domain Controllers, or File Servers. Look also for user-definable ways of breaking these groups into logical sub-groups of high-risk and low-risk machines. Note, also, that systems must be able to belong to multiple

groups to be truly useful for deployment.

Use automated mechanisms to re-discover the network periodically to capture information about any systems either added to or removed from the network. Discover frequency varies directly with how often systems come, go, or are rebuilt on your network. Re-discovery processes should happen more frequently on networks that change often, and happen less frequently on more stable networks. What is most important is that there is a process in place to capture information regarding changes on the network.

Step Two: Analysis

The next step—analysis—assesses current patch levels. Done manually, this requires researching every system's configuration and current levels, which is not feasible with most staffing levels. Leading-edge patch management applications scan the systems they discover for installed and missing patches. The accuracy of this step is critical. Worst-case scenarios are “false positives”—reporting a patch as present when, in fact, it is not. This may tempt admins to skip applying the patch. The less-critical counter to this is a “false negative”—reporting a needed patch is not present when in fact it is. This will usually result in the re-application of the patch, with little harm done.



Patch analysis scrutinizes several information points, including the operating systems and applications installed on a given device. Based on this information, most tools consult a “master list” of patches available for a given OS and application and determine which of these patches are present and which are not. This “Master List” is analogous to antivirus software’s virus definition files. Best practices mandate that you regularly and faithfully download this list from vendor websites. Most patch management products download these files automatically.

Having collected this information, patch management products generate reports on installed and missing patches. Many applications show the data in several ways, depending on what specific piece of information an administrator needs to see. One particularly effective approach provides three distinct views of patch levels. A “Hosts” view allows you to see the installed and missing patches on a per-computer basis. A “Products” view shows the complete list of products supported by patch management software, which machines are running those apps, as well as the current patch levels. A “Patches” view focuses on a specific patch and reveals which machines have or do not have that patch.

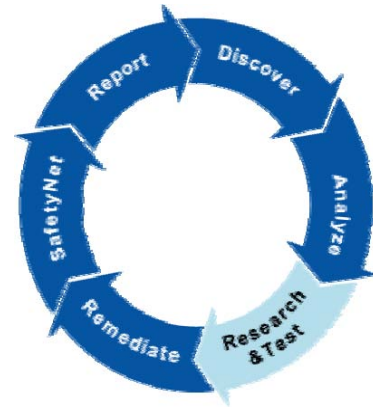
This last view can be helpful in identifying all instances of a specific vulnerability across a network. Performing network analysis within 24 to 48 hours of the release of a new patch helps determine a network’s exposure to the vulnerability. This information, together with severity and risk information, reveals how vital a specific patch is to a

given network's security.

Analysis is an essential step in remediating an unpatched or irregularly patched network. This phase verifies which machines have a particular service pack or patch and which machines do not. Use the patch management software to bring any exposed machines into compliance with established patch policy. Leading edge patch management solutions automate the definition and analysis processes across multiple systems.

Step Three: Research and Test

Steps One and Two develop a clear picture of a network's current patch levels. These levels will fall into one of three categories: 1) Fully-patched, in which all of your systems are completely up-to-date; 2) Totally unpatched or 3) Somewhere-in-the-middle. For those with networks in the first category: do not sit back and relax; there will always be new patches to deploy. For those in the second and third categories, read on.



Research

Best practices emphatically recommend that, before deploying any service packs or patches to your network, you research what you are about to deploy. Occasionally, a patch or even service pack will have an unexpected negative impact on a machine; therefore, it is necessary to understand what you are deploying to your network.

Look to your patch management solution vendor to provide independent engineering notes from patch testing. Expect vendors to test patches in-house and to note information regarding incompatibilities. Review resources such as the CIAC website that review and detail vulnerabilities. Vendors publish articles describing vulnerabilities and include release notes and/or a read-me files describing installation options and precautions. Vendor testing should never replace your own, however. Every environment is unique and third party or custom software makes for unpredictable interactions. Based on information you collect, you should determine the following for each patch you deploy:

- What is the nature of the vulnerability?
- What application or OS component does it affect?
- How easy is it to exploit the vulnerability?
- What is the severity of the vulnerability?
- How much damage will an exploit cause?
 - Vulnerabilities are typically rated as low, medium, high or critical, critical being the highest level of potential damage should the vulnerability go unpatched.
- What is your level of exposure to the vulnerability?
- How many (if any) machines on your network are affected?

Use the above information to guide patch deployment. Conduct a risk analysis. For example, a high occurrence of missing patches for severe vulnerabilities signals which systems need immediate attention. A severe vulnerability on a mission-critical application server needs remediation sooner than does a low-severity exposure on a workstation. Such determinations identify priorities in testing new patches for deployment.

A Few Precautions

Best practices include taking reasonable precautions in the case of system upgrades that involve installing a new operating system or a newer version of an existing OS. Before making any change review all appropriate release notes and any deployment guide. There may be recommendation to back up critical data or the entire system before deployment, so read carefully.

Test

Test patches before deploying them because patches sometimes break systems. This is an unalterable fact of patch management. Even a fully tested service pack may conflict with some as-yet-undiscovered quirk in a small number of environments and, when that conflict occurs, servers are at risk to come down. Therefore, the need to test in a specific environment, on specific machines, cannot be over-stressed.

Testing involves applying patches to a test environment prior to deploying them to a production system. Developers write patches quickly to address a critical issue. Therefore, there is not always time to thoroughly test a patch prior to release. This is not to imply that patches are untested; but the testing is not nearly as extensive as in the case of a service pack, which goes through beta testing and review prior to release. Of course, service packs should not be immune to your internal testing phase. Although vendors test service packs thoroughly, no vendor can test every update in every possible environment, so every patch or service pack needs testing before being deployed into your production environment.

So how do you test a patch or service pack? Deploy it to a test machine configured like the production system(s) that need the patch. Patch management vendors highly recommend developing a test environment and using that environment to test patch deployment before deploying to production. Large corporations often have a lab that contains enough systems to create an environment that mimics the actual corporate network, complete with Domain Controllers, Member Servers, and workstations. Smaller companies often settle for a test environment containing one or two machines configured exactly like their production machines, or virtual machines loaded and reloaded based on what needs testing. At the bare minimum, if a test environment is not available, deploy and test patches on low-priority production machines first.

Whatever the test environment, use the patch management software to create a logical group of machines and deploy patches that need testing to that group first. Then observe

and record the results.

- Is the system still functioning?
- Are the applications and services on it still functioning?
- Do the results of the install coincide with the expected results
 - Confirm updates to application extensions
 - Confirm updates to registry keys

If all runs smoothly, deem the patch safe. If a problem occurs, go back to the research phase. Check websites such as www.ntbugtraq.com to determine if anyone else is experiencing problems like yours and if there is a workaround. Determine the root-cause of the problem and decide if deploying the patch is still worthwhile

Testing using Image-Based Systems

Image-based system deployment involves using imaging (or cloning) software to create a “master” image of a computer hard drive. This image can then be compressed and stored on a server or CD-ROM. Create the image by copying data off a manually installed and configured system, block-by-block, and storing that data in a compressed file format. The image replicates a fully configured operating system, including applications, settings, and, in many cases, patches and service packs. The image can then be copied to a “bare-metal” system that, once rebooted, is an exact replica (or clone) of the original system.

This can be advantageous when testing patches and service packs. Most organizations that use the cloning process have several images stored representing individual workstation or server configurations on their networks. Therefore, patch testing involves copying one of these images onto a bare-metal system, deploying new patches to it, verifying stability, approving the patches for production, updating the master image, and rolling the patches out to production.

Patch Level Minimum Baselines

A fully rational patch policy asks that you specify the minimum patch level required on a network. This minimum patch baseline will be unique to each network and presumes a thorough understanding of the analysis, research, and test phases. Consider a series of unpatched machines and then ask the following questions:

- What patch level do I want to achieve?
- Do I want every possible patch deployed, regardless of severity?
- Do I want to have all Windows 2000 machines at Service Pack 4?

If you feel comfortable with Windows 2000 Service Pack 4, you may choose to define that as your minimum baseline. If you know several machines on your network are susceptible to a given vulnerability, the required patch for that vulnerability should also be part of your baseline.

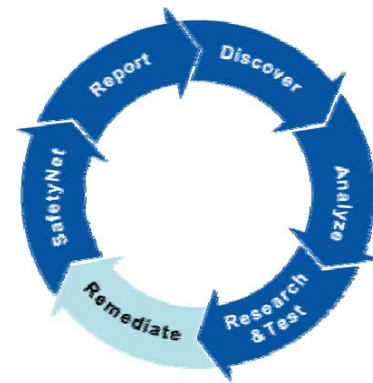
Some administrators are happy with service packs. Service packs are essentially rollup packages of bug fixes, security-fixes, and feature enhancements released every six to twelve months. They are usually beta-tested in production environments and fully tested

by the vendors. Because of the extensive testing, they usually represent the most stable and reliable operating system or application updates you can install. For this reason, many administrators will install only those patches included in a service pack (with the possible exception of high-severity patches for vulnerabilities with active exploits in the wild.) For these administrators, a Windows 2000 system with the current service pack installed may represent a well-patched system. For other environments, a well-patched system has not only the latest service packs applied, but all post-service pack patches are in place as well. Whatever the case may be the latest service packs will generally represent the best place to start.

Step Four – Remediate

Remediation means to correct, update, patch, or rollback to bring a system into compliance. This phase involves patch deployment, installation, and un-installation (if necessary) in a controlled manner.

The remediation phase is actual patch download and deployment. Remediation occurs during the process of bringing a network up to the minimum baseline, every time a new computer comes online, and every time a newly released patch applies to any systems on a network. This is where the automation of patch deployment is most effective.



Because downloading patches is critical to all phases of deployment, leading edge patch management applications can automatically contact vendor websites and download the most current patch-definition database and any new patches available.

Incremental Rollout

Best practices stress the need to deploy patches incrementally. Rather than blanketing a patch out to thousands of machines at once, follow the above testing recommendations, analyze the results, and then deploy to small groups of machines. This is a good way to identify incompatibilities without potentially wreaking havoc on the production network due to a bad patch. Once the patch is ready for production deployment, start with just a portion of the environment. This portion could be a single subnet, or perhaps a department. Following successful deployment to that subset, deploy to another subset, then another. Depending on the size of the environment, go as quickly or as slowly as is comfortable. This approach gives the advantage of rolling back problem patches from subsets rather than from an entire network.

Note: The testing phase of deployment technically begins the remediation process. Testing removes a given vulnerability from a test environment. Production remediation removes that same vulnerability from across the enterprise.

Scheduling Reboots

Another consideration for patch deployment is the reboot that are sometimes required following the installation of a patch or series of patches. In many environments, it is not feasible to have a production server down for any length of time during peak hours. Advanced patch management solutions allow you to schedule patch installs, especially those that require reboots, for off-peak hours or weekends, or at least deferring the reboot of the computer until a more convenient time.

Policy-based Remediation

Policy-based remediation is essential to an effective patch management strategy. Policy-based remediation promotes deployment by patch-level baselines and rule-based remediation. For example, a policy stipulates a rule such as, “All Windows 2000 Service Pack 3 machines, with MDAC 2.6 installed, must have xyz patch installed.” Once the rules and conditions are set, policy-based solutions enforce this policy across the enterprise or selected relevant groups. View policy-based remediation of multiple patches across multiple machines as an extension of the baseline concept.

It is essential that your patch management application allow you to create policies, analyze compliance with policies, and deploy based on non-compliance.

A Few Precautions

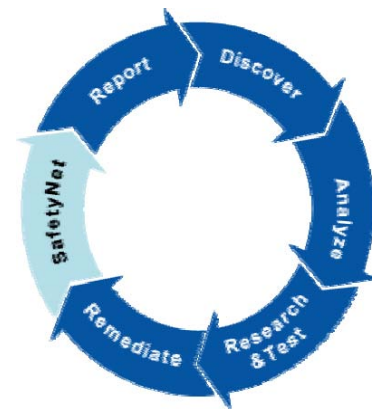
Best practices include taking reasonable precautions in the case of system upgrades that involve installing a new operating system or a newer version of an existing OS. Before making any change review all appropriate release notes and any deployment guide. There may be recommendation to back up critical data or the entire system before deployment, so read carefully.

Step Five – The Safety Net

Step five should not, hopefully, require constant attention, but may become necessary in the event that an applied patch causes problems. The safety net comes into play when a patch requires rollback.

Rollback is the ability to uninstall a patch and restore the system to a prior state. When a patch causes problems, having the ability to uninstall the patch is highly desirable. Best practices favor a patch management solution that allows for rolling back patches individually or removing them from policies, depending upon the nature of the patch.

Rollback is also effective whenever a patch gets deployed without having gone through proper authorization. Many companies employ a change management policy. This essentially describes the processes and procedures that govern any change to a network or machine. Many such policies consider patches or service packs as configuration changes



that require prior approval or authorization.

Since roll-back support is not universal to all patches from all vendors, it would be wise to include a procedure or process for documenting configurations and tracking changes in your best practices. Before and after snapshots of system settings and registry keys allow manual restoration of a system if necessary.

Step Six – Reporting

Reporting is the final step in the Patch Management process. Administrators must be able to confirm to upper management the successful deployment of patches and verify that there is no negative impact. Reporting should expose situations, such as failure to deploy, that require an immediate return to the analysis phase. Reporting not only gives managers an opportunity to review Patch Management process and look for areas of improvement, but also provides valuable statistical information regarding patching activity. In environments subject to internal or external audits to meet industry or federal regulations, documenting changes is crucial to proving compliance.



Recent developments among patch management vendors have energized and expanded this mission-critical reporting process. It is now possible to generate graphical reports that plot user-selected variables in easily understood bar and pie chart formats. This valuable capability allows managers to capture compliance trends over time, for example—“how are we doing with regard to installing critical patches?”—or system status at a specific point—“where were we back on day ‘x’”—and preserve the data in clear, unambiguous, non-technical formats.

Similarly, best reporting practices allow administrators to pinpoint the precise information they need. When presenting a summary of missing patches, for example, a fully effective reporting function allows an administrator to click on a field for a specific patch and access additional engineering notes relevant to that patch.

Return to Step One - Close the Loop

These six steps comprise a closed-loop patch management process. Best practices emphatically stress the need to repeat each of these steps as often as possible. Solutions that automate the process make the loop reasonably self-maintaining. For some networks, this will be daily, for some, weekly, and others, monthly. The preceding six steps are central to a regular maintenance plan for any enterprise. The steps are as basic and as fundamental as defragmenting drives and updating antivirus software.

Choosing an effective patch management product

There are several variables involved in selecting an effective and efficient patch

management product for your environment. First, agree that you *will* have to choose one. By now, it is clear that complying with a rational, comprehensive patch policy requires an automated solution. Key selection criteria include:

Platform Support

Does the solution support all of the operating systems present in your environment? One of the most popular products for deploying patches to Microsoft networks is Microsoft's own Software Update Service (SUS). However, SUS does not apply UNIX environments or patch popular applications such as Microsoft Office, Exchange, or SQL.

Application Support

Does the solution support all of the applications present in your environment supported, at least the most vulnerable or most critical to the business?

Usability

How steep is the solution's learning curve? When choosing any product, strike a balance between ease-of-use and functionality. It will do no good to buy the latest and greatest product if the interface is confusing and time-consuming.

Features

Choose a solution that allows for as much scheduling and automation as possible. Some of the things your product should include:

- Scheduling and alerting
- Discovery of servers and workstations through multiple methods
- Flexible resource grouping
- Quick, accurate, and thorough analysis
- Policy-based analysis and remediation
- Automated roll-back support
- Reporting for both administrators and managers
- Ability to compare test systems with production systems

Agent-based vs. agentless

One hot debate focuses on agent-based vs. agentless deployment. Agent-based solutions generally provide more functionality, consume less network bandwidth, and support mobile users via local client scanning technology. They also consume time, money, and resources in deploying agents, and they risk introducing destabilizing effects on clients and workstations.

Agentless solutions install easily and take less time to deploy. Because they do not install software onto servers or workstations, they minimize potential risks of downtime due to conflicts. Conversely, an agentless solution does not automate patching of hardened or "locked down" system, or laptop machines that irregularly connect to a network.

Current best practices migrate toward solutions that provide an optional agent architecture. With this approach, administrators can deploy agents only to those devices

requiring them, thereby delivering an automated solution to the entire enterprise. The optional agent approach allows administrators to reserve bandwidth and to minimize resources dedicated to patching the majority of systems that do not need an agent.

Cost

Patch management solution prices range from “free” to “how much have you got?” Generally, higher costs bring added features and functionality, but not always. Best practices mandate considering the total cost of ownership: figure in cost of implementation, training, and any associated learning curve, customer service, and maintenance.

Security Policies

Security best practices go beyond those appropriate to patch management software. Security policies are documents that describe expectations regarding all aspects of security in networked environments. These policies cover everything from Internet usage to password policies and, ideally, describe how users should handle e-mail attachments, unsolicited e-mail, unknown websites, and other common conduits for viruses and worms. A good security policy contains provisions for patch deployment, describing how and when to apply new patches throughout the enterprise, and an acceptable “discovery-to-patch” timeframe. A security policy will also discuss how to enforce and audit the policy, as well as how to handle violations.

Change Management Plans

Often undocumented or uncoordinated changes seriously affect a system or network; therefore, it is important to create controls that prevent such changes from happening arbitrarily. A change management plan includes an approval process for a change to take place, as well as the procedures for carrying out the change. Following the written plan keeps unexpected changes to a minimum and provided documentation and roll back guidelines if planned changes wreak havoc.

Emergency Response Plans

A viable emergency response plan describes what to do in the event of an emergency—be it a single computer security compromise to a full-scale natural disaster. These plans typically include emergency telephone contact numbers, evacuation plans, and business-continuity plans (BCP) in the event of a total-asset loss.

Conclusion

Effective Patch Management is mission-critical for today’s information technology environments. This is so because of: 1) The ongoing discovery of vulnerabilities in existing operating systems and applications, 2) The continuing threat of hackers developing applications that exploit those vulnerabilities, and 3) Vendors’ needs to patch vulnerabilities via the release of patches. These points illustrate the need for continually applying patches to computing environments. Today’s economic constraints mandate adhering to automated best practices to get the job done.

Therefore, it is important to look at patch management as a closed-loop process that identifies and removes network vulnerabilities. Patch Management requires the regular discovery of systems, scanning those systems for vulnerabilities, downloading patches and patch definition databases, and deploying patches to systems that need them. To recap the six-steps:

- **Discovery** – The discovery phase involves locating assets (workstations and servers) on your network and categorizing them.
- **Analysis** – The analysis process determines current patch levels and sets a minimum baseline policy.
- **Research and Testing** – This phase focuses on researching missing service packs and patches, conducting appropriate risk analyses, and testing patches on lab systems.
- **Remediation** – This involves bringing systems up to date by deploying patches in accordance with the established patch policy.
- **Safety Net** – The safety net, although not always a necessary step, provides the ability to roll back a patch should the need arise.
- **Reporting** – Reporting conducts a change review and verifies successful deployment of patches. Comprehensive reporting also includes enough review, analysis, and adjustment to close the loop and begin the cycle again automatically.

Following these six steps and repeating them regularly makes the process of bringing your network into patch compliance quick, effective, and accurate.

Useful Links

For more information on Risk Analysis, visit [The Society for Risk Analysis](#)

For more information on Change Management, visit the [Change Management Resource Library](#)

For information regarding general security best practices, visit [The Rainbow Series Library](#) and Microsoft's [Security Center](#).

We hope you have found this white paper useful.
Please email dpratt@ecora.com with any comments or suggestions.

© 2004 Ecora Software Corporation. All rights reserved.

Novell is a registered trademark of Novell, Inc. Cisco is a registered trademark of Cisco Systems. Solaris is a trademark of Sun Microsystems, Inc. Microsoft, MS-SQL, and Windows NT are registered trademarks of the Microsoft Corporation. Oracle is a registered trademark of Oracle Corporation. Lotus and Domino are trademarks of Lotus Corporation. Ecora is a registered trademark of Ecora Software Corporation