A Practical Guide for Patch Testing

Nelson Ruest

A Report by Wise Solutions

Installing Web Applications: A Complicated Proposition

This white paper provides a practical overview of patch management and testing and its relation to enterprise software packaging. It aims to help administrators and packagers understand how to master the software and operating system patching process and integrate it in their packaging activities. It outlines a structured method for the implementation of patch management in any release strategy. In addition, it makes a strong case for patch testing, a process that is often overlooked due to the harried approach many use for patching. It points out that testing is often synonymous with stability and that there must be a balance between security, timely patch delivery and patch testing in every patching strategy.

Its intended audience is IT executives, system administrators and packaging technicians who belong to organizations that want to improve their control of software and operating system management in the enterprise.

About the Author

Nelson Ruest is an IT professional specializing in systems management and design. He is coauthor of multiple books and articles. His two latest books have been published by McGraw-Hill Osborne and are focused on Microsoft Windows Server. The first explains how to deploy Windows Server in the enterprise and is titled "Windows Server 2003: Best Practices for Enterprise Deployments", ISBN 0-07-222343-X and the second is a practical administration guide titled "Windows Server 2003 Pocket Administrator", ISBN 0-07-222977-2. He is a frequent speaker at conferences worldwide and also helps run a consulting firm from Victoria, British Columbia.

Table of Contents

1. The Case for Patch Management

In its latest semi-annual security report, Symantec Corporation indicates that Slammer is still the number one systems attack despite a patch being available for more than two years and the original attack being over a year and a half old.

– Symantec Internet Threat Security Report[2]

Patches, patches, patches. Everybody is talking about patches. They're the hottest thing in IT. Is there anything new to say about patching today? You might think not, but the answer is yes, especially since Microsoft just released Windows Installer version 3.0—a version that is completely concentrated on supporting patching. This new version of the Windows Installer is now available either through Service Pack 2 for Windows XP or through an independent download for other platforms such as Windows 2000[1].

If you're not packaging software today, then you won't be interested in this white paper. That's because you need to go back to the drawing board and review why you aren't controlling this most important aspect of your computing systems through the use of a comprehensive software packaging strategy. If you are packaging today, then read on and find out how you can integrate your patch management and your packaging strategies to reduce the amount of effort required to manage both.

In a way, patch management supersedes software management. That's because it doesn't only affect software products; it also affects operating systems (OS) and other core tools that are often integrated with the OS. For Windows, this includes items such as Internet Explorer, Microsoft® Data Access Components, and the Microsoft .NET Framework, to name a few. That's not to say that only Microsoft products are affected by patching. All products require patching to some degree. It's part of the lifecycle of software components. Microsoft seems more prominent in this arena because their repertoire of products is so varied. That's right; the Microsoft Web site currently lists more than 800 available hot fixes and patches.

Patching is a fact of life no matter which operating system or other tools you use. The question is: how do you balance the need to patch systems with the need for stability within the organization? The answer: implement a patching strategy integrated with your software management and operating system release strategy.

1.1 The Patching Dilemma

There are several patching strategies. One strategy is to let your security team dictate when and how patches should be deployed. In emergencies, this can lead to deployment without sufficient testing and cause more problems than repairs, as well as have a negative impact on users. A system that is patched but doesn't work may be secure, but it is not very useful to the business.

On the other hand, if patches are forced to fit into your normal software release strategy, they might be released too late to protect your organization from external threats. Somehow, you have to balance the need for security with the need for stability. This means you have to test patches before they are deployed, but these tests must be designed in such a manner that they can accelerate the release process for patches.

Thorough testing takes time. You know this, because each time you create an operating system build or a new software package for deployment you take the time to ensure that it is fully functional before it goes out the door. But, according to the Giga Group, most successful software attacks are based on known vulnerabilities. That's the patching dilemma: you need to make sure you thoroughly test patches before deployment, but you also need to make sure you deploy patches in a timely fashion if you want to be protected (see Figure 1.1).

Of course, your patch management strategy must go hand in hand with your overall security strategy. You can no longer rely solely on the firewall to keep your network protected. Attacks no longer occur only at the physical layer. More and more attacks affect the application layer. That's why any complete security strategy must now include firewalls, antivirus as well as system patching, among other elements. The best practice is to comprehensively harden systems and applications before deployment and to update them on a constant basis thereafter.

1 The Windows Installer version 3.0 redistributable file is available at http://www.microsoft.com/downloads/details.aspx?FamilyID=5fbc5470-b259-4733-a914-a956122e08e8&DisplayLang=en.

2 For more information on the Symantec Internet Threat Security Report, please see http://www.symantec.com/press/2003/n031001.html.

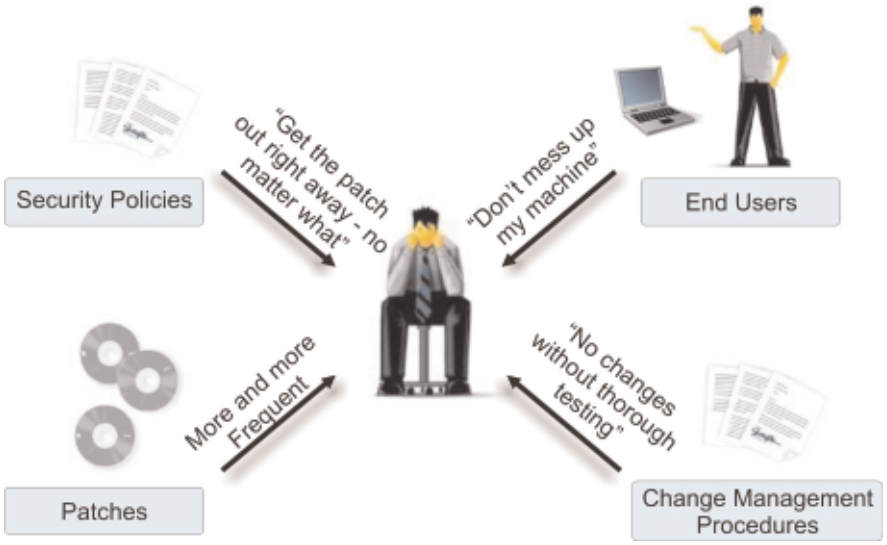

Figure 1.1 – The Patch Testing Dilemma

In the end, you have to move beyond simple patch management and on to systematic systems build management practices. This means you have to build both PCs and servers using a structured and layered approach, knowing what is in each functional layer at all times and knowing how each layer will be affected each time you add a new or updated component to it[3]. Then once your systems are deployed, you need to implement regularly-scheduled maintenance deliveries. These deliveries should include at least two practices. The first should deal with low priority updates and should be delivered at monthly or even quarterly intervals. The second should deal with high priority updates and should include a fast track delivery mechanism that still conforms to your quality control standards, but also supports your requirements for a secure infrastructure. Both should take into account the patch release schedules maintained by your software vendors or internal development teams.

1.2 Vendor Patch Release Schedules

The first thing to do then is to identify the patch release schedule your vendors use. For example, Microsoft has developed the following monthly release schedule:

- Second Tuesday of each month at 10:00 a.m. Pacific time, Microsoft posts updated Security Bulletins and posts update packages on the Microsoft Download Center (www.microsoft.com/downloads) as well as on the appropriate update site, such as Windows Update or Office Update.
- Security alerts and bulletins are also sent at that time to people who have subscribed to the Microsoft Security Notification Service. These notices are never sent until after both the bulletins and the patches have been made available on the respective Web sites.
- Microsoft also briefs several security industry reporters to help ensure that more customers are made aware of the vulnerability and the steps they can take to mitigate it.
- The day after the release at 10:00 a.m. Pacific time, Microsoft will host a Security Bulletin Webcast and conducts a question and answer on the topic session to provide information about the vulnerability.

"The majority of successful attacks against organizations are a result of the malicious masses using know vulnerabilities in which a patch has been already released. Companies that properly maintain security of their systems will eliminate 90 percent of all potential exploits."

– Giga Information Group

Security Alert!

Microsoft never sends out patches or updates as attachments to email messages.

Security Alert!

You should take the time to check that Microsoft has indeed posted the bulletin on the TechNet Web site (www.microsoft.com/technet/security/current.aspx) to validate that the electronic message you received does indeed identify a real threat.

3 For more information on layered systems build approaches, refer to Enterprise Software Packaging, Practices, Benefits and Strategic Advantages, a white paper from Wise Solutions, Inc. at www.wise.com/sysadmin_resources_page.asp.

If the update is deemed critical, Microsoft will release it outside of the monthly schedule in order to help customers protect themselves in a timely manner. Other manufacturers use different schedules, but with a similar strategy. At the very least, you should find out what the schedules are for the most popular products in your network.

On November 9, 2004, a new worm broke the speed record for the time between the announcement of a security-related vulnerability to the development of a full-blown virus. On November 5, Microsoft announced vulnerability in Internet Explorer and only four days later a worm exploiting the IFRAME vulnerability was discovered. This only strengthens the case for strong patch management in your network.

1.3 The Patch Management Process

The starting point for any patch management strategy is the environment you work with. You need to know and understand this environment fully. Both your PCs and servers must have documented baselines for each system category. The best way to do this is to create a single base system image for each OS, then to add either user or server roles as appropriate to include a series of software components to support each of them. Once these roles are identified, they must be fully documented and captured so that it is possible to revert to these baselines at any time. Any change to the baseline should be documented and captured as a delta which can be applied to modify the baseline. At specific intervals, the baseline must be updated to include the deltas and released as a new baseline.

In addition, staff must be trained to understand the impact of patch management and its integration to the release management strategies of your organization. Team roles[4] must be assigned and understood (see Figure 1.2). Team members must be given the right tools to monitor baselines and deltas as they evolve. Many tools are free—Microsoft Baseline System Analyzer, Microsoft Software Update Services (soon to be Windows Update Services), Windows Automatic Updates and the Microsoft Windows/Office Update Web sites all provide required functionalities. Others, such as integrated IT lifecycle management suites, must be purchased separately but provide a wider range of capabilities including packaging, patch management, and software inventory. Teams must be familiar with each of these tools and the role each plays in the release strategy.

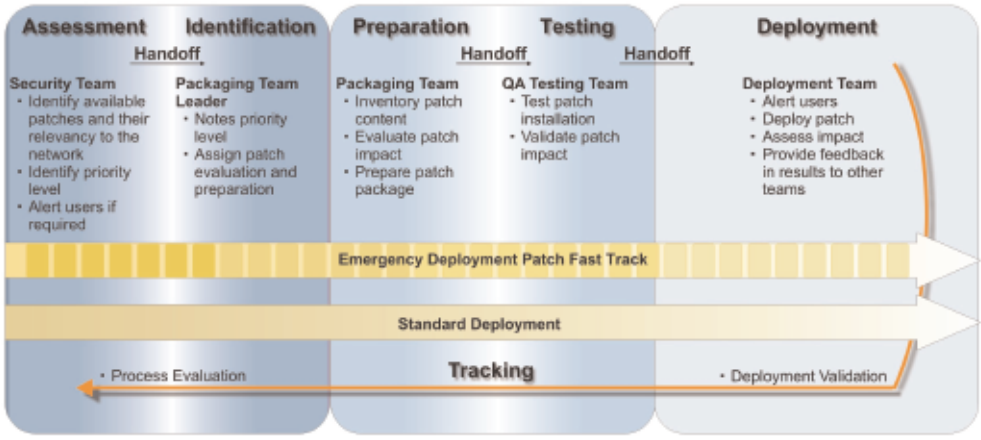

Figure 1.2 – The Patching Team and their Interaction

This team uses a structured process to support the interaction of each team role. This process has six phases: assessment, identification, preparation, testing, deployment and tracking. Each phase concentrates on one part of the whole. In addition, two schedules are available: the standard release schedule whose frequency is based on corporate requirements and the patch fast track which supports emergency releases.

4 For more information on team roles, refer to The Case for Quality Assurance in Enterprise Software Packaging, a white paper from Wise Solutions, Inc. at www.wise.com/sysadmin_resources_page.asp.

The first phase, assessment, is performed by the security team. It requires knowledge of the system baselines along with updated inventories of actual systems. It includes threat, vulnerability and infrastructure readiness assessments. Is the deployment infrastructure in place and ready to support emergency deployments? Are all processes as fine-tuned as they can be? One good way to find out is to use Microsoft's free Operation Framework Self-Assessment Tool (http://www.microsoft.com/technet/itsolutions/techguide/mof/moftool.mspx). It runs you through a series of questions on Service Management Functions or service areas to help you determine if your processes meet the requirements for excellence in the service area. Another good source for this type of assessment is the IT Infrastructure Library (ITIL). ITIL provides guidance at all levels of IT management. One good place to start is with the ITIL Toolkit (http://www.itil-toolkit.com).

The second phase focuses on the identification of available updates and their relevance to your network. Notices are received by email or through your patch management system and are first validated, then assessed for deployment in the network. If the update is marked for deployment, it must be rated—is it critical, important, or of moderate or low importance? The result of this phase is a released change request to the packaging team leader who assesses the requirement in turn to assign it to the proper packaging team and process. This assessment includes the identification of targeted systems, other ongoing change activities on the network, an evaluation of the cost of update release (time and effort), the distribution method, the resources required and the release dependencies.

The third step should be viewed as the core of the patching process because it focuses on update installation automation. The change request includes the update rating. Given each of the ratings, the evaluation and preparation process will proceed at different speeds:

- Emergency updates are released within 24 hours.
- High importance updates are released within one month.
- Medium and low importance updates are released within four months to a year.

The packaging team needs to use the appropriate tools to evaluate the impact of the patch on target systems. This is a key aspect of the update preparation because software is increasingly complex and interdependent, so a patch intended to fix a problem in the operating system or in a particular application may have unintended consequences to other applications. Since it is the packaging team's responsibility to maintain system stability at all times, patch impact evaluations are an essential part of the preparation process. Once the evaluation is complete and potential impacts have been identified, the packaging team proceeds to the automation of the update installation and performs basic testing on the update.

The update is then handed off to the quality assurance team whose role is to validate that the update works properly and will not adversely affect target systems. This step is overlooked more often than not, yet it is a critical part of the patching process. Update tests target the package delivery strategy, identify if reboots are required, validate that uninstallation works and ensure the patch cohabitates with other applications. Tests can also target a small number of production machines to ensure processes are complete and functional. Once the update is fully tested and you have confirmed that it does not compromise business critical systems and applications, it is added to the baseline and delta inventories and a release timetable is prepared. Both update and timetable are passed on to the deployment team.

Phase five focuses on the release of the update to production systems. The deployment team must alert users of the upcoming patch as well as warn the help desk team of the upcoming deployment. It must prepare for installation rollback if required, and then proceed to the deployment. Once deployment is initiated, it must be monitored to ensure that all target systems receive the update.

The last phase focuses on tracking both the update deployment and ensuring that system builds include the update from now on, as well as ensuring the delivery process has met all requirements. The first is performed by the deployment team and the second is completed by the security team.

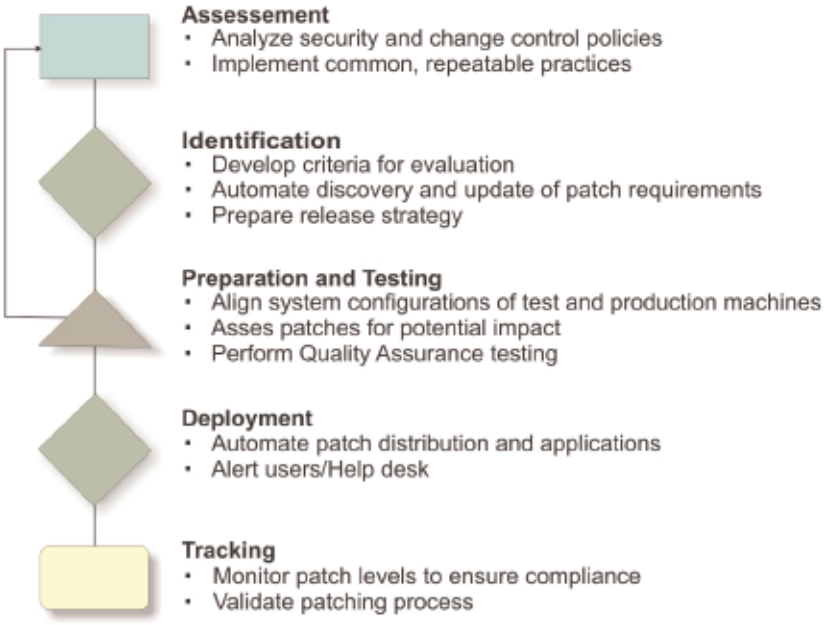

Figure 1.3 – The Update Preparation Process

The requirement to maintain updated systems only reinforces the need to reduce the diversity of desktops and applications in your organization as well as the importance of having proper policies in place for application preparation and deployment. Both are supported by a proper classification of applications by target user group or server role and a reduction of the number of base images supported by your organization. The less you use, the easier it is to maintain. This is one reason why application rationalization—or the removal of applications providing duplicate functionalities—is so critical to the support of a proper patch management strategy.

In the end, you know you need to patch, but you also have to control the impact of the patches you deploy on your systems. The only way to do this is to put in place both the proper tools and processes to fully test all updates before their release.

2. The Anatomy of a Patch

Before you can integrate patching to your release strategy, you need a thorough understanding of patches in general. Though Microsoft is working hard at reducing this number, it currently has 10 different types of patches that can be deployed, depending on the affected product. Until Microsoft changes this strategy, it is important for you to understand each patch type.

In addition, you'll need two different strategies for patch management. The first addresses new products and products that have not yet been installed. These are easy to patch since they haven't been deployed yet. The second strategy deals with products that are already deployed in your network. These are more difficult to patch since a delivery system for the patches is required. They are also more difficult to deal with because they may affect or even disrupt production.

2.1 Working with Different Patch Types

The starting point for any patch management strategy is the environment you work with. You need to know and understand this environment fully. Both your PCs and servers m

Microsoft has recently confirmed its terminology for Software Updates, which became effective July 26, 2004[5]. It now has ten different types of patches.

- **Security update** — A security update is a broadly released fix for a product-specific, security-related vulnerability. Security vulnerabilities are rated based on their severity. The severity rating is indicated in the Microsoft security bulletin as critical, important, moderate, or low. Microsoft security updates are available for customers to download and are accompanied by two documents: a security bulletin and a Microsoft Knowledge Base article.
- **Critical update** — A critical update is a broadly released fix for a specific problem that addresses a critical, non-security-related bug. Critical updates are available for customers to download and are accompanied by a Microsoft Knowledge Base article.
- **Update** — An update is a broadly released fix for a specific problem. An update addresses a non-critical, non-security-related bug. Microsoft updates are available for customers to download and are accompanied by a Microsoft Knowledge Base article.
- **Update rollup** — An update rollup is a tested, cumulative set of hotfixes, security updates, critical updates and updates that are packaged together for easy deployment. A rollup generally targets a specific area, such as security, or a component of a product, such as Internet Information Services (IIS).
- **Software Update** — A software update is any update, update rollup, service pack, feature pack, critical update, security update, or hotfix that is used to improve or to fix a software product that is released by Microsoft Corporation.
- **Hotfix** — A hotfix is a single, cumulative package that includes one or more files that are used to address a problem in a product. Hotfixes address a specific customer situation and may not be distributed outside the customer organization. Hotfixes are distributed by Microsoft Product Support Services. Customers may not redistribute hotfixes without written, legal consent from Microsoft. The terms Quick Fix Engineering (QFE) update, patch, and update have been used in the past as synonyms for hotfix.
- **Service pack** — A service pack is a tested, cumulative set of all hotfixes, security updates, critical updates and updates. Service packs may also contain additional fixes for problems that are found internally since the release of the product and a limited number of customer-requested design changes or features. Service packs are broadly distributed and are tested by Microsoft more than any other software updates. Microsoft service packs are available for download and are accompanied by Microsoft Knowledge Base articles.

5 For more information, see the fllowing Microsoft Knowledge Base article: http://support.microsoft.com/default.aspx?kbid=824684.

- **Integrated service pack** — The combination of a product and a service pack in one package. This feature is based on the service pack slipstream concept. By "slipstreaming" a service pack or applying it to installation code, you can create an updated version of the installation code for a specific Microsoft product. Slipstream service packs are supported by Microsoft operating systems and many of its other products.
- **Feature pack** — A feature pack is new product functionality that is first distributed outside the context of a product release and that is typically included in the next full product release. For example, Windows Server 2003 includes quite a series of feature packs: Active Directory in Application Mode, Windows SharePoint Services, Automated Deployment Services and Group Policy Management Console, to name a few.
- **Upgrade** — An upgrade is a software package that replaces an installed version of a product with a newer version of the same product. The upgrade process typically leaves existing customer data and preferences intact while replacing the existing software with the newer version.

As you can see, there are a considerable number of different updates available from Microsoft alone. You should also make sure you understand the patching terminology for all your other vendors.

2.2 Patches and Software Deployment

Software must be packaged before it is deployed. Packaging software means automating its installation and customizing its configuration so that users are not tasked with these activities, which are out of their scope of work anyway. In addition, software should be patched as much as possible before it is deployed. Take for example, Office 2003. If you haven't deployed it yet, then you should ensure you integrate service pack 1 into your deployment package. This will avoid having to deliver two different packages—one for Office, followed by one for the service pack. This is done by slipstreaming the service pack into the original installation CD. Basically, you perform an administrative installation of the original Office CD, then unpack the administrative installation version of the service pack and run a command to apply the service pack to the administrative installation. You can then use this updated installation source for your deployments.

Integration with Software Deployment

If you choose to use a commercial patch and software delivery system, you should ensure that the software packaging tool you choose supports this distribution mechanism. Ideally, it will let you create packages and automate the hand off of these packages to the distribution system.

This example illustrates a significant issue with patching. The process you use to patch software depends mostly on when in the software product's lifecycle you decide to deploy it. If you deploy a product when it is first released, your patch strategy will be to update the product once it is deployed. If, on the other hand, you decide to deploy a product when it is considered more mature and has been out for some time, then you can update the product before deployment. Of course, once it has been deployed, you then fall back onto the same scenario outlined above; you must patch it while in use.

Both scenarios mean you need to have patch distribution software in place. Whether it is Microsoft's free Software Update Services or another commercial product, your tool must be able to identify available updates, let you examine which products are affected, let you test the patch and let you deploy it to target systems.

For the time being, a commercial product is most likely the best approach since Software Update Services does not address all of Microsoft's products. This is set to change in the near future as Microsoft rolls out Windows Update Services and a revamped Windows Update Web site. Both will include a framework which will support the addition of new Microsoft products as they become available without the need to change the underlying framework. In its first release, Windows Update Services will support the patching of various versions of Windows, Office, Exchange, SQL Server and the Microsoft SQL Server Desktop Edition (MSDE). In the long term, it is slated to support all Microsoft products.

Microsoft's effort to revamp their patch distribution tool demonstrates the importance of a comprehensive system that supports patch delivery in any organization. There are several reasons why you shouldn't wait for this new version and move to procure a third-party patch delivery system if you haven't done so already.

First, a good commercial patch delivery system could ideally be the same as your software delivery tool, performing two functions from one single interface. Second, a commercial tool will support more than just Microsoft product patching and patch all the products in your network. Third, having a patch delivery system in place will let you manage patches in a more timely fashion. Finally, a commercial product would more fully support patch testing prior to deployment, helping maintain long term system stability.

3. The Patch Testing Process

Now that you have the prerequisites to patch management in place, you can proceed to the elaboration of your own patch strategy. The first place to start for this is to identify how patch preparation, testing and deployment fit into the software packaging lifecycle (see Figure 3.1). As you can see, software packaging is much like assembly-line production. Each team works on specific aspects of the entire process. Everything starts with the request which can come from a variety of sources. Once the request is approved, you move on to package preparation which includes discovery, then the actual packaging of the product to deploy. Discovery is the ideal stage to identify if there are any security patches to apply to this product before deployment.

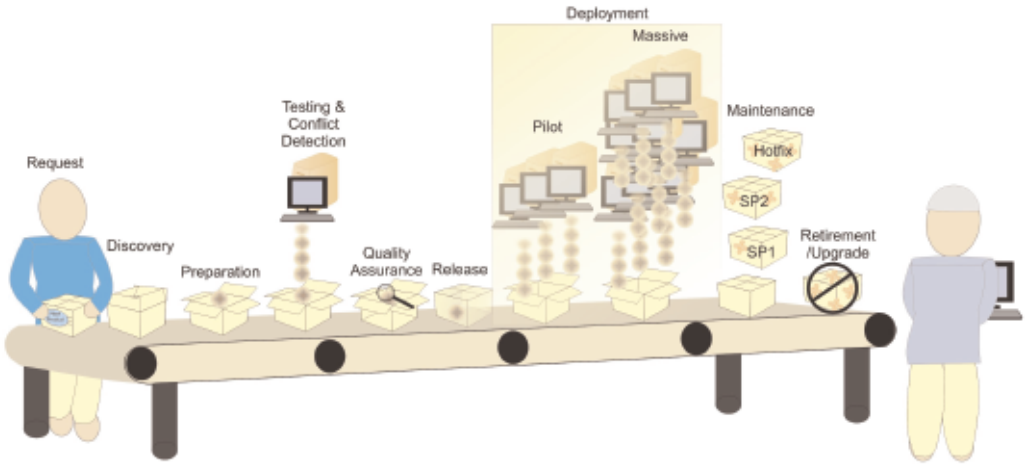

Figure 3.1 – The Software Packaging Lifecycle

The next step is testing and conflict detection. Both of these steps are key to the patch discovery process because they allow you to identify potential impact without having to deploy on actual machines. This is followed by quality assurance (QA). This step validates that everything is as expected. Whenever possible, QA team members should be different than the packaging team members to ensure that another pair of eyes takes a look at the product.

Once QA is complete, the product is released. If it is a new product, you will probably want to test its deployment with a pilot project. Here you should also include the patching process as part of your pilot, deliberately omitting a non-critical patch so that you may deploy it during use. This will ensure that you will be able to maintain the product once it is deployed to the massive user base. Once the product is deployed, it falls into maintenance mode. This mostly consists of delivering adjustments and patches as they become available. Large patches such as service packs require official deployments. Smaller patches can simply go through the patching system. When the product is ready for retirement, it is much like a patched box as is illustrated at the end of the production line in Figure 3.1. Here, the number of patches applied to a product will help determine whether or not your organization should move to either a newer version of the product or another product altogether. Severely patched products are much more costly to maintain than newer, repaired versions.

3.1 To Test or Not to Test

It goes without saying that the key to any patching strategy relies on two core elements:

- The first involves having a patching technology in place. You must have a patching system in place, even if it is only the free system offered by Microsoft. This includes at least two tools: Software Update Services and the Microsoft Baseline Security Analyzer.

Deployment Testing

Newer versions of integrated packaging environments now include the ability to perform "pre-flight deployments"– deployments which send empty packages to target systems. These pre-flight deployments help ensure that target systems meet all the requirements to host the new product.

Timely Patch Testing

The amount of time that administrators have to protect systems against potential vulnerabilities continues to shrink. For example, the vulnerability related to the buffer overrun in JPEG processing was exploited within 12 hours of the time that Microsoft released the hot fix to remedy the problem.

- The second deals with having a patch management strategy in place that assesses, identifies, evaluates and prepares, deploys and tracks patches in your network.

These two elements are critical to your ability to deal with patches, but even having both in place, you'll soon realize that at some point in time you need to make some hard decisions about how patches, especially critical security updates, are managed. The major issue with these types of patches is the testing strategy that is used prior to deployment. There are several key factors that can greatly help reduce testing time in this case because the longer you take to test, the more at risk you may be. It is true that if you use traditional testing methods—targeting tests on installed physical machines, you cannot test patch and application compatibility in a timely manner. These types of tests take time because they rely on an operator basically checking the operation of every component on the target machine. This is why your testing strategy must be revamped.

You know you have to test, but you want to be able to do it in a timely manner while still maintaining a level of quality that will not adversely affect your production systems once the patch is deployed.

3.1.1 The Testing Environment

To ensure that you have the proper capability to test, you need to have the proper tools in place. These include:

- A patch identification tool. This tool is designed to let you view available patches for the products running in your network. It lists patch content, version, affected products and potential vulnerabilities. This should be an integral part of your patch delivery system. You do not want to duplicate this functionality in another tool.
- An integrated packaging environment. The IPE is key to proper patching testing and preparation. It should include the following features:
  - A conflict detection database that can store information about both base systems, especially in the form of standard operating environments, as well as information about each of the applications you deploy in your network.
  - This database should support the importing of patch data, automatically if at all possible. This way, all you have to do is download the patch to the appropriate directory and it will be automatically imported into your database. This database should also support the identification of patch dependencies and sequencing (and possibly supersedence). It should store metadata for patches and it should support the grouping of patches into families.
  - Preflight patch testing that supports the creation of an empty patch package to test target machines and identify the impact of the deployment. This test simulates the patch deployment to see what impact it would have on the target PC or server, and then report on whether to deploy the patch or deployment group as is, to customize it before deployment, or simply to not deploy it because the risk is too high.
  - Local system testing because you may not yet have a complete conflict database that includes all of the applications in your system. Local testing lets you target real machines to identify conflicts.
  - Support for Patch Grouping. The advantage of such a tool is that it can handle setting the correct command lines to run installations silently and to suppress reboots, as well as to integrate the QChain.exe command automatically if necessary.
  - Support for virtual testing, identifying the proper operation of the patch package in deployment, but without affecting the actual target machine.
  - Most importantly, it should have explicit support for Windows Installer version 3.0.
- Virtual machine technology. This technology goes way beyond the use of disk images to restore physical machines to a pristine state. With virtual machines, you can simply restore a machine to a pristine state by either using undoable disks or by having a backup copy of the files that make it up. This saves untold amounts of time.

In addition to the tools, you need to have documented processes for patch fast tracking as well as standard patch evaluations.

3.1.2 Testing Processes – Patch Impact Assessment

There are several types of tests that you must run on patches. The very first test is the identification of the relevance of the patch in your environment. Some patches or updates may simply not apply to you. This assessment is performed at the identification stage.

The next step has to do with patch impact. This is why a comprehensive conflict management database is so crucial to the patch evaluation process. It only takes a few minutes to evaluate which products are affected by a patch or a series of patches when they are loaded into a conflict detection database.

It only takes a few minutes to evaluate which products are affected by a patch when it is loaded into a conflict detection database.

That's right. Having a properly configured conflict detection database that not only includes your applications but also your standard operating environments or system stacks saves you enormous amounts of time because it quickly identifies which products may be affected by the patch. This means that you won't have to evaluate the impact on every single product in your network. You only have to test out the products which have been identified by your conflict database. In large and even small networks this can mean the difference between tests on a handful of applications instead of dozens or even hundreds. **This is the single most important factor in fast-tracking patch deployment.** Once this evaluation is done, you can move on to the standard battery of tests on the patch.

3.2 Standard Patch Testing

Once you've evaluated the impact the patch may have on specific applications, you can create a target system that includes all of the affected applications. Ideally, your IPE can group these applications together into a deployment group that can be targeted to either physical or virtual machines. Virtual machines would be best because they can be created much faster than physical machines. Packaging the applications into a deployment group automatically links their installations together into a single sequence. This saves enormous amounts of time in test system preparation. If multiple patches are being tested, you can also deploy the patches as a group, ensuring only one installation sequence is required for all patches. Ideally, the deployment groups your IPE supports would include all of the rules and validations that patch families support to ensure proper installation sequencing.

Monthly Releases

Microsoft releases operating system and product patches on the second Tuesday of each month. Ideally, your IPE will allow you to integrate all patches into a family or group that can be tested as a single whole. If any one patch of the group presents a higher risk, you can remove it from the group and continue with your deployment preparation for the rest.

Here are some of the tests you should run on the patch or patch deployment group. They are divided into two testing groups: lab testing and preflight deployment testing.

- Lab testing relates to tests of the packaged update on target systems located in the lab. They include:
  - Installation tests – These tests are used to validate that the patch installs without error and that any launch conditions contained in Windows Installer patches are working properly.
  - Verification tests – These tests are used to verify that shortcuts, help files and file associations set or modified by the patch are working properly. The verification tests are also used to verify that Class IDs and Prog IDs set by the patch can be properly instantiated.
  - Execution tests - These tests are used to verify whether the files and registry keys created or modified by the patch can be read and updated when the application is executed by typical users who do not have administrator-level privileges.
  - Standard tests - These tests are used to verify that the installation of a patch does not negatively impact the ability to execute another application found on the desktop or the ability to connect to a URL, network share, or database.
- Pre-flight deployment tests are based on the deployment of an "empty" package that simulates the operation of the patch or update package on all target machines. They include:
  - Disk Space Check - Does the target machine have enough free disk space to install the patch?
  - File and Registry Security (Read and Write) - Do users of the target machine have sufficient privileges to read and update the files and registry keys that are created or modified by the patch?
  - Files in Use - Are any of these files that are replaced by the patch in use on the target machine?

– Files Installed - How many of the files in the patch will actually be installed on the target machine? How many of the files will not get installed due to file versioning rules, the conditional nature of some of the installation's components, etc.?

– Valid Patch Target - Is the machine a valid target for the patch?

– Patch Installed - Has the patch been previously installed on the target machine? If so, it may not be necessary to deploy the patch to that machine.

These tests focus on both patch content and patch installation. They also help you identify how the patch will affect other components in your system architectures.

3.3 Integrating Testing to the Release Management Process

Finally, this testing schedule must be integrated into your release management strategy. Ideally, your systems are constructed through a model—a model that includes a core standard operating environment (SOE), additional application groups arranged by user profile for PCs and arranged by server role for servers as well as specific standalone packages for special users or smaller target groups. One such model, called the Point of Access to Secure Services or PASS, is outlined in Enterprise Software Packaging, Practices, Benefits and Strategic Advantages, a previously released white paper from Wise Solutions, Inc.[6]

In support of this model, you need to have a regular release schedule for the update and maintenance of each component that is in the SOE or "Kernel" of the system. Because of the number of items you include in the SOE—core operating system, Internet Explorer, runtimes and utilities, Microsoft Office, and so on, you may want to extend this release schedule to give yourself the time to properly prepare your deployment. The ideal release schedule is once every four months or three times a year (see Figure 3.2). The test of time has proven that longer release schedules—six months to a year—often cause the up-date package group to be larger in size than the core SOE it will repair. This makes for very impractical deployments. In addition, shorter release cycles put too much pressure on the system administration and maintenance team. Note that the same type of release schedule is required for application groups as well as for the ad hoc applications which do not fit into any group.

Another advantage of this release strategy is that all change requests are gathered and collated in one central location. This allows administrators to ensure that all changes fit the standard change management strategy your organization supports.

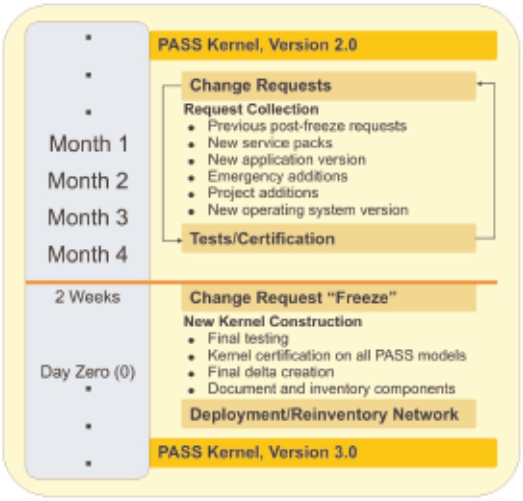

Figure 3.2 – The PASS Kernel Update Release Schedule

6 This white paper can be found at www.wise.com/sysadmin_resources_page.asp

This release schedule also allows you to fit in emergency releases (see Figure 3.3). If urgent change requests arrive, they cannot wait to be fitted into the normal update release schedule because this schedule takes too much time. Urgent requests can come from many sources. In patching, they emerge from unforeseen critical updates. In standard operations, they can emerge from deployments that go awry or from urgent configuration change requests. A proper testing strategy for all releases and all deployments goes a long way to reducing the event of the latter, but the first instance—emergency patching—is always unforeseen. That's why your patch management strategy needs to take this into account and be ready for the emergency deployments.

When dealing with emergency releases, you should use the same testing strategies outlined earlier, but to fast-track the process, you should focus only on the following:

1. Identify available patches.
2. Validate the emergency situation and ensure it is not a hoax.
3. Assess your own level of vulnerability and the relevance of the patch or patches to your environment.
4. If the update applies to your environment, load it into your conflict management database.
5. Identify potentially impacted products from your entire product set and review the level of the impact. Use the conflict management tool to resolve immediate issues.
6. Generate the patch package.
7. Run it through the integrated testing tool included in your IPE. Repair/fix any non-working elements and remove any extraneous files/components.
8. Generate a product deployment group that includes impacted products.
9. Deploy the product group to virtual machine images of your SOEs.
10. Deploy the patch package to the same SOE images.
11. Test the proper operation of the package and the products it affects.
12. If time permits, prepare a pre-flight deployment package and target key machines. Repair any issues.
13. Evaluate if you can proceed with the deployment.
14. Deploy the patch package and monitor its success rate. Repair any issues that arise.

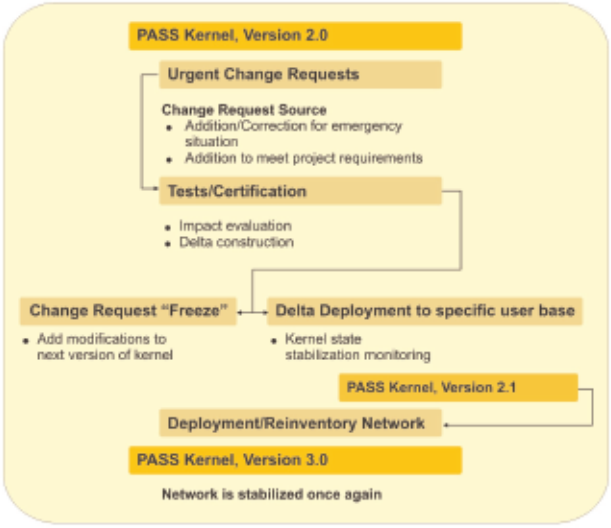

Figure 3.3 – The PASS Kernel Emergency Release Schedule

If you have the proper tools in place and your team is properly trained, this entire process should not take longer than a single day.

4. Windows Installer 3.0 Features that Support Patching

Microsoft recently released Windows Installer version 3.0. Windows installer drives the installation of most any software product on Windows today. With version 3.0, Microsoft has enhanced the ability to manage patches and updates through Windows Installer. For example, when patching products such as Microsoft Office, you no longer require access to the original installation files. This is only one new feature included in this product. There are many more, all focused on patching.

Today, most every program released for Windows is installed through the Windows Installer service. This service creates an installation database on the target computer. This installation database is then used to support program viability features such as program consistency and self-repair. Applying patches for software products that are integrated to the Windows Installer service means updating this installation database and modifying key components, often dynamic link libraries (DLLs) of the program. Original installation files are MSI files. These files can contain all of the components of a program or, if the program is too complex, can contain only installation instructions and point to additional installation component files. Original MSI files are often transformed through MST files to customize their installation within corporate networks and adapt them to corporate standards. When patches are released for products that are integrated to the Windows Installer service, they use the MSP extension. Patches modify the original installation database as well as key program components (see Figure 4.1)

Deploying Patches with Active Directory

One of the major advantages of the MSI/MST/MSP approach is that it is integrated to Active Directory's software delivery capabilities. If you do not have a delivery engine in place, this integration becomes a godsend since all you have to do is assign the delivery to specific targets within your directory structure. Non-MSI-based patches can also be delivered with this mechanism through "ZAP" files. Zap files are small text files with a .zap extension that include basic information about non-MSI applications.

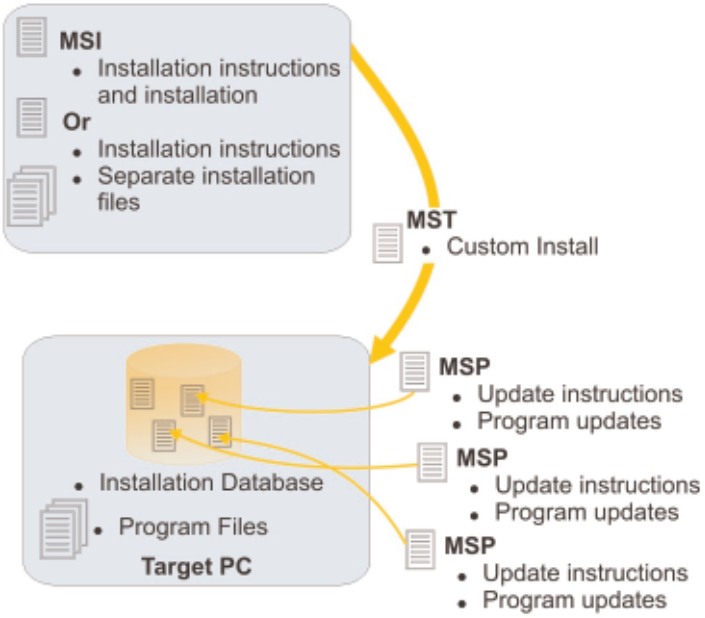

Figure 4.1 – The MSP Patching Process

Unfortunately, not all products are integrated to the Windows Installer service. For example, Microsoft SQL Server 2000 does not install through Windows Installer and therefore cannot use the MSP patching process. The same applies to operating system patches. Since the operating system is the component that hosts the Windows Installer service, it cannot use this service to install its own updates. Finally, device drivers are often not integrated to the Windows Installer service and therefore updates cannot be in the MSP format.

With the release of version 3.0 of the Windows Installer service, Microsoft has concentrated on making patching smarter for installed applications. Before version 3.0, there was no structured way to approach patches. This means that independent software vendors (ISV) pretty well used the strategy they deemed best to apply patches to their original MSIs.

This lack of standardization made for the proliferation of patching strategies at the Windows Installer level. It also created confusion because there was no set manner to ensure that patches were applied in the proper sequence. With version 3.0, Microsoft has standardized all approaches to patching by including specific features that support the ability to update an installed program and make sure the updated program also updates the installed database that maintains its consistency.

One new feature of the MSI database is the new MSIPatchSequence table. This table ensures that patches are applied in the proper sequence, especially if they have prerequisites. It also ensures that if a user applies a series of patches in the wrong chronological order, say Service Pack 1 after Service Pack 2, Windows Installer will not overwrite the updated components, but will still record that both patches have been applied. Patch chronological order control is now an integral part of Windows Installer. This is a significant change. Previous versions of the service installed patches as they encountered them on the target machine. With chronological ordering of patches, Windows Installer can now make use of the logical order patches should be installed in. If it encounters a series of patches on the target machine, it will put them in the proper logical order before applying them.

Because of this feature, Windows Installer version 3.0 brings some minor changes to the terminology outlined earlier in section 2.1. These changes affect three specific terms:

- QFE Patch — A QFE patch—also called a "Small Update" patch or hot fix—is a patch which does not change the ProductCode or ProductVersion value for the product within its installation database. These patches are created to make small scale changes to the product and usually update a limited number of the product's files.
- Service Pack Patch — Service Pack patches (also called "Minor Upgrade" patches) are patches that provide significant updates to the product. These patches always increment the ProductVersion but never change the ProductCode. Service Pack patches usually update many files in the product and usually incorporate the changes made by all prior hot fixes for the product.
- Depending on the manufacturer's strategy, service pack patches may be cumulative or non-cumulative. Cumulative patches incorporate the changes made by all prior Service Pack patches and target several versions of the product, including the first version of the product. Non-cumulative patches do not include changes made by earlier Service Packs and only target the most recent version of the product. Some patches may combine the two behaviors, incorporating changes from Service Packs released after a certain date or version, but not older patches.
- Major Upgrade Patch — A Major Upgrade patch is a patch that changes the ProductCode of the patch and may change the ProductVersion. Major Upgrade patches may not be uninstalled by Windows Installer 3.0, and may not be sequenced using MSI 3.0 sequencing behavior. In general, releasing a new MSI file that upgrades a product using the RemoveExistingProducts action is preferred over using a Major Upgrade patch.

Windows Installer sequencing functionality focuses on QFE patches because when multiple QFE patches apply to the same version of the product, the ProductVersion value stays the same for all of the patches. For example, if you have three patches to apply and all of them apply to version 1.2 of the product, there is no way to identify in which order these patches should apply. The ProductVersion value of a package does not provide enough information about the patches for the Windows Installer service to determine the best ordering sequence. To do so, Windows Installer needs additional information—information about patches that update similar functionality, restrictions in the application order of the patch, whether or not the one patch includes changes made by other patches and how to handle patch removal. With Windows Installer 3.0, this information can now be captured and stored in the sequencing metadata.

As a consequence, Windows Installer version 3.0 fully supports patch uninstalls and removals. This means that you now have a mechanism for rollback in the event of a problem arising from patch deployment. This alone is one good reason to deploy this version of the installer service.

Interfacing to the MSIPatchSequence Table

You should make sure that the software packaging tool you use has been updated to work with Windows Installer version 3.0. One easy way to do so is to see if it offers a user-friendly interface to the MSIPatchSequence table because this table can normally only be viewed through the patch creation tool once the patchwiz.dll (from the Windows SP2 software development kit) has been installed.

Managing Patch Families

The software packaging tool you use should support the concept of patch families and allow you to create your own families or otherwise group patches together for deployment to target machines. Patch sequencing of this type is essential to every patching strategy because it avoids having to do multiple deployments when multiple patches are available at the same time.

Patches can also be grouped into "families" which serve to provide grouping logic for the order of installation of the patches. Using patch families, ISVs can provide instructions which identify the role each patch has in servicing the target product. Families group together patches that update a specific set of functionality. Patch families generally only apply to a single product, but can span multiple products in the case of functionality used by multiple applications. This means that most products only need a single patch family, but may require multiple patch families if they use shared components. In this case, patch families can be compared to mini-service packs that target only a given set of functionalities in a product. A Service Pack itself will now be made of several patch families grouped together in a given sequencing scenario.

Finally, one of the greatest features of Windows Installer version 3.0 is its ability to apply product patches by relying on the installed product database. Prior to version 3.0, Windows Installer required access to the original installation source to be able to apply a patch. If for some reason, the exact original source was not available, patch installation would fail. In version 3.0, Microsoft removed this limitation and rightly so. Because of the former limitation, many organizations went without patching because they could not locate the original installation source code. This new functionality will go a long way towards making sure all products are patched at all times. This enhancement has also been extended to applications that are integrated to the .NET Framework and now eliminates the need for Windows Installer to access the original installation source in order to patch an assembly installed in the Global Assembly Cache (GAC).

The bottom line: you should move to Windows Installer version 3.0 as soon as possible in order to help simplify your patching strategy.

5. Using Testing to Reduce Patch Management Overhead

There is no doubt that you need to deal with patches today. It is a fact of life. The best way to do this is to have on hand the right tools and techniques to meet the patching challenge and integrate it into your normal release strategies.

There is also no doubt that patches cannot be released without as thorough a testing as you have time to apply to them. Testing patches lets you achieve several goals and benefits:

- Reliable computing performance and availability
- Fewer desk-side interruptions to users by IT support staff
- Proactive vulnerability management
- Up-to-date distributed applications
- Productive and responsive system administration
- Empowered end users who have properly functioning tools
- Substantial cost savings because your team reacts proactively rather than reactively
- Improved change management because your release systems include how to handle emergency situations
- Better use of new and existing IT assets

All of these advantages help make your IT environment operate as it should: constantly. This is quite an asset given that IT lives in an ever-changing world.

Building a trustworthy environment requires a comprehensive and integrated approach that helps you maintain control over your organization's IT assets and resources. This approach can only be accomplished by constructing and implementing a comprehensive structure that tightly integrates:

- People
- Policies
- Architecture
- Asset management
- Software distribution and management
- Recovery
- Patch management
- Integrated management tools
- Release Management

In the end, a proper patch management strategy will balance speed with thorough testing to ensure your IT environment remains secure, but stable at all times.

References

Enterprise Software Packaging, Practices, Benefits and Strategic Advantages, a white paper from Wise Solutions, Inc.

The Case for Quality Assurance in Enterprise Software Packaging, a white paper from Wise Solutions, Inc.

Preparing for .NET Enterprise Technologies. Nelson Ruest & Danielle Ruest. Addison-Wesley. ISBN: 0-201-73487-7.

Windows Server 2003: Best Practices for Enterprise Deployments. Nelson Ruest & Danielle Ruest. McGraw-Hill Osborne. ISBN: 0-07-222343-X.

Repackaging Basics. Wise Solutions Inc.

Measuring the Returns of Software Packaging. Tim Wilson, Enterprise Management Associates, April 2003.

Desktop Management, Certifying Software Prior to Deployment. David Friedlander, Giga Information Group, September, 2003.

Microsoft Knowledge Base article number 824684:
http://support.microsoft.com/default.aspx?kbid=824684

Microsoft TechNet Patch Management Process:
http://www.microsoft.com/technet/security/guidance/secmod193.mspx

Active Patch Management, a white paper from Altiris, Inc.:
http://www.altiris.com/eval/wp/