Your Patch Management

STRATEGY GUIDE

MODEL TECHNOLOGY SOLUTIONS

UEM

Unified Endpoint Management

YOUR PATCH MANAGEMENT STRATEGY GUIDE

A Walk Through for Simple and Smart Patch Management in Enterprise Organizations

CONTENTS

When we were kids, whenever we got stuck in a video game-- a boss a we couldn't beat, a level we couldn't pass, an item or secret room we couldn't find-- we could hop down to the local toy store or gaming den and buy the strategy guide.

That strategy guide would have a walk through for the entire game, along with tips, tricks, and things to try. No matter how good you were, you could learn something new from those guides.

Now we're older, and the stuff we deal with day-to-day is a lot more complex. Still, don't you sometimes find yourself wishing that there were a strategy guide to walk you through things?

That's how a lot of our clients feel when it comes to patch management. It's not that they didn't understand the game of patch management. Nor was the problem that they couldn't muddle through it themselves. But they desperately needed a strategy guide to walk them through the steps so they didn't miss anything or get stuck.

Patch management is kind of our thing here at Model, so we figured: Why don't we write that strategy guide?

WHAT YOU'LL LEARN HERE

- The importance of patch management for enterprise organizations
- The challenges that keep organizations from good patch management practices (and what to do about them)
- How to sell management on a better patching strategy
- Starting your strategy by thinking through goals and success metrics
- How the overall patching process looks
- Important questions to ask when patching both workstations/endpoints and servers/data-centers
- Plans for emergency patch deployment
- And more

FYI, much of the information in this guide has appeared, on one form or another, on our blog over the past year. So, if you're reading our blog religiously, we're sorry if some of this information seems redundant.

There is information that is all new in this guide, however. Readers interested in this new information should check out the sections "selling decision makers on patch management" and "automating patch management."

One last note: Though the principles here hold good no matter what your environment looks like, we here at Model are experts, first and foremost, in Windows environments and patch management tools. So, when we get down into the weeds, that's what we're going to be talking about.

For what it's worth, we genuinely feel that Microsoft has some of the best products on the market for endpoint management and updating. So, if you are struggling with your patch management strategy in your current environment, now might be the time to start transitioning to a secure Windows environment that can handle, even automate, your patching strategy. Just putting that out there.

THE IMPORTANCE OF PATCH MANAGEMENT FOR ENTERPRISE ORGANIZATIONS

STARTING WITH THE BASICS: WHAT IS PATCH MANAGEMENT?

Patch management is simply the process of acquiring, testing, and installing patches (and possibly other upgrades) for software applications.

It is the simplest, most effective thing you and your organization can do to minimize your risk profile, especially when it comes to cyber attacks.

That said, many organizations have little to no patch management strategy in place, leaving themselves vulnerable to attack. Others do patch management, but they do it manually. Depending on the number of endpoints, this could easily mean a whole team of people running around and trying to stay on top of patch releases and their effects on machines. Believe me: It's tiring just to watch.

Why Is Patch Management Important?

The importance of patch management cannot be overstated.

Software patches are a fact of life in any software environment: They are necessary not only to fix bugs post-release, but also to close security gaps and exploits that are discovered. In fact, most patches today are more about security than functionality.

Now consider, for a moment, just how many patches are needed in an enterprise environment. Operating systems and typical desktop apps will need patching, of course. But so will servers, routers, firewalls, and other pieces of network infrastructure, as well as websites and website apps. Laptops and mobile devices, too, need consistent patching. Very quickly, the number of patches required on a consistent basis becomes unmanageably huge.

Installing a patch requires a process, too, to guarantee success. It is not unheard-of for a faulty patch to prevent a critical application from functioning. When this happens to a single end-consumer, it's frustrating. Now imagine it happening to, say, your entire finance department. Those kinds of scenarios make testing and deploying patches a critically important part of patch management as well.

THE IMPORTANCE OF PATCH MANAGEMENT FOR ENTERPRISE ORGANIZATIONS

The Biggest Challenge(s) in Patch Management

Ask a room full of IT experts "Who here has mastered patch management?" and you are more likely to see people rolling their eyes than raising their hands. This is because patches can be a source of grief for system admins.

Why? Patch management shouldn't be a big deal. But it too often is. Here are the 4 biggest roadblocks we find-- see if any of them apply to you:

#1: "Fixing" What Isn't Broken

An admin's biggest fear when installing a patch is that the patch will be installed, the server or workstation will be restarted, and then…. Nothing. The machine either won't come back up, or will have issues when it does. This will mean further testing, and so further downtime—and more end-user frustration.

Nobody likes it when IT comes to "fix" or "improve" a machine, only to have it stop working. That fear is the #1 reason folks don't upgrade to begin with.

#2: Keeping Up Takes Time and Manpower

If you have several hundred (or several thousand) endpoints with dozens of apps, that's a lot of patches to keep up with, let alone test. We knew of one company that had a 20-man team working to keep patches downloaded, deployed, and tested. 20! That's a lot of IT manpower tied up in one task.

Not every organization has that kind of manpower or budget. So, rather than figure out how to do the same tasks with less manpower, they just let it fall to the wayside. That works until an app stops running-- or worse, a data breach occurs.

#3: There's a Lot More to it Than Meets the Eye

Managing patching and patch requirements is actually pretty difficult. It requires time and a solid understanding of things like network dependencies, services, and order of criticality. The picture is getting even more complicated with so many apps now running in the cloud. Let's face it: Although IT folks tend to be pretty smart people, organizations are just not putting patch management into their training budgets.

And if the expertise is not there, patching is going to take longer and be more susceptible to errors.

#4 Let's be Honest--It's Not Sexy

A colleague of ours compared patch management to wearing a hard hat at a construction site: It's not comfortable, fun, or sexy. In fact, it can seem inconvenient. And it doesn't even contribute to the building getting built. But you do it every time on the off-chance that something really bad happens.

Even if you do manage to get it done, the nature of the task can be problem. Keeping every system patched and secured against every vulnerability can become a tedious, never-ending task in itself. And we know what happens when a task gets tedious…the chances of making a mistake or missing something go up exponentially.

The Patching Dilemma

You can see why IT departments face a dilemma when it comes to patching. On the one hand, regular patching is important for fixing vulnerabilities, ensuring compliance, and keeping all machines current and set to a standard configuration.

On the other hand, few organizations have the manpower of the expertise to stay on top of it. The fear of downtime or "bricking" machines makes them even more afraid to even try.

Which is too bad, because most attacks exploit old, unpatched vulnerabilities that have been known for years. The attacks work, because organizations are not staying on top of those patches.

Selling Decision Makers on Better Patch Management

What if we could tell you about a better way to do patch management? That, in essence, is what the rest of this guide is about.

Still, we realize that many readers might need to convince others of the importance of patch management, and the wisdom of our approach. (Even if you, personally, have been tasked with patch management for your organization, you might need to convince management to open the purse strings in order to do it right, yeah?) So how do you do that?

Here are some ideas...

THROW SOME STATS AT THEM

There's some good empirical proof that inadequate patching is a huge business risk. For example:

57%

of organizations who report a breach say that they were breached due to a vulnerability **for which a patch was available, but not applied.** (This might be an underestimation; the true number might be as high as 80%.)[1]

Organizations that have been breached, rated themselves much (41%) lower, on a scale of 1 to 10, in their ability to patch in a timely manner[1].

ONLY 61%

of organizations complete their patching process; patches not completed after 12 weeks **tend to go unpatched[2].**

A SURVEY FOUND

50% of CIOs naming out-of-date security patches as one of their top information security issues[3].

Be Ready to Talk ROI

Find some numbers and outline three costs: The costs to the IT department to handle patching manually, the cost of having a third-party automated solution, and the cost of an actual data breach.

You can then compare the cost of both the manual and the automated solutions with the business risk posed by not patching:

COST OF SOLUTION < CHANCES OF DATA BREACH X COST OF DATA BREACH

Ideally, the cost of your patch management operations should be less than what it would cost if you had a data breach, times the chances of a breach occurring.

---

1. https://www.servicenow.com/content/dam/servicenow-assets/public/en-us/doc-type/resource-center/analyst-report/ponemon-state-of-vulnerability-response.pdf
2. https://www.automox.com/blog/2017-cyber-incident-breach-trends-report-patching-critical
3. http://www.information-age.com/why-your-business-cant-afford-not-patch-123459579/

Cost of data breach include things like:

Direct costs that incur in resolving the breach. This can include forensics to determine the root cause of the data breach and affected users, disclosures and announcements, legal services, remediation services, and specialized training for your call center.

Opportunity costs, such as lost customers due to the breach, free or discounted services for victims, loss of reputation due to the breach, and added loyalty/acquisition programs needed to win customers back.

Indirect costs associated with all of the above, in terms of time and organizational resources.

You can find a lot of good data here in the **2018 Ponemon Institute Cost of a Data Breach Study.**

Have a Concrete Plan in Mind

If you throw all of these findings at someone, their response is likely to be something like this:

"Yikes! So what now?"

If at this point you start speaking in abstractions, or gesture at some fuzzy idea of what you could do, you're going to lose your audience. You need to have some concrete steps in mind for going forward. You are going to need a plan.

Fortunately, that is exactly what we are going to cover next...

STARTING YOUR PATCH MANAGEMENT PLAN

Let's start the walk through of your patch management strategy.

Planning is a huge part of a good patch management strategy. There are some initial decisions that have to be made which will affect large parts of the process later on. If you fail to formulate a plan, problems will arise—like unexpected downtime or incompatible patches stopping critical apps, or even bricking your devices.

Step 1: Define Goals and Success Metrics

There are a lot of reasons to keep up with the latest patches in a systematic way. We saw many of those in the last chapter. Still, it helps to know exactly which reasons are driving your patch management strategy, and how you will measure whether those goals have been successfully met.

For example, many industries—like retail and financial services—have heavy requirements when it comes to compliance. You may measure success, then, by the percentage of machines that are in compliance… once you have a clear definition of what compliance means. Does compliance pertain to critical updates only? All available updates? Important updates? Office updates? These options need to be thought through.

Another common metric is speed of patch deployment once a patch is available (for example, knowing which servers have patches that are non-compliant over 60 days since release). Understanding your requirements and business/compliance tolerance is important.

Step 2: Identify Targets

This next step seems simple but is crucial: Identify the list of things that need patching. At the very least, this will include which workstations, servers, and devices will need patching, both on-premise and out-of-office. (Hopefully, these have already been identified as part of your overall Unified Endpoint Management strategy.) It should also include a separate list of applications that will need regular patching.

Remember, though, that apps and devices are different, and have different testing functionality, than workstation packages. For example, a workstation that is patched and gives no response often means that it has passed all tests, whereas a "no response" response from an app could mean trouble.

Along these same lines, be sure to identify and document any exceptions. Some servers or applications might not be able to be upgraded or patched because they must maintain compatibility with a critical application that is in use. If that's the case, you need to make sure you have an alternative strategy for securing that system from the vulnerability left exposed by the inability to patch the software.

Step 3: Identify How the List of Available Patches is Determined

How do you determine which patches are available, and what subset of those are necessary or critical?

One way, of course, is to have the point of authority determine this for you. Automate that, and it will help ensure everything is up-to-date with the latest patches (depending on the policies you set, of course).

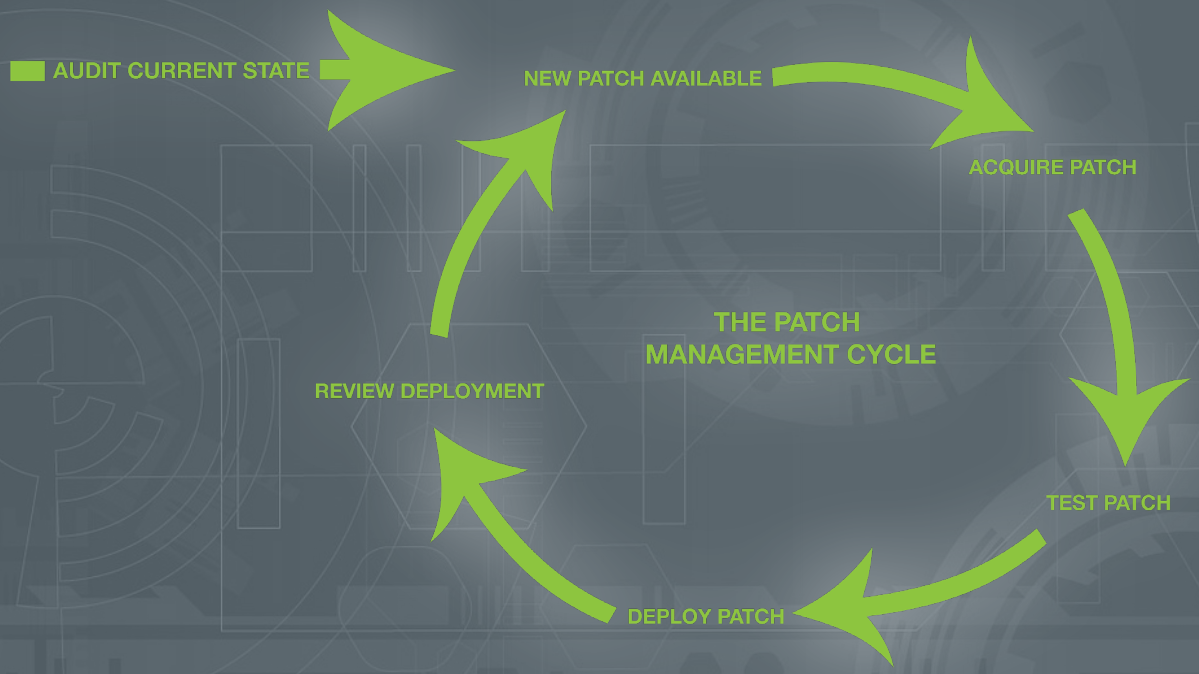
Another way is to have your own internal security team do this. There are many scenarios where you might not want to trust the point of authority at all times. For example, if you are a law firm that depends day-to-day on certain Microsoft Office add-ins, you might not want to run the risk that a patch breaks that those add-ins. Sometimes, there is a trade-off between showing compliance and overall compatibility.

Step 4: Start Thinking Through Deployment Rules

This is the point at which you should review deployment rules as well. Some updates, such as security updates, might need to be deployed quickly and broadly. Rules will need to be defined (or tweaked) to reflect this.

Let's say, for example, that you are automating the patching process using SCCM. This will allow you to define rules for automatically finding and deploying patches; these are called Automatic Deployment Rules (ADRs). Each time an ADR runs, it can add software updates to a new or existing software update group. For example, you can have all machines save for mission-critical ones receive all critical security updates a week after "Patch Tuesday" and deploy automatically. You can read more about ADRs from Microsoft's own article on the topic. (There's a nice example here, too.)

THE PATCHING PROCESS



Best practice is to follow a standard 22-day procedure for piloting and testing. This includes:

1. Test stage ("smoke test") A patch is acquired and installed on a low-priority workstation or server. This verifies that the patch works and does not break key applications, and also gives you a sense of what problems to anticipate.

2. Pilot stage It's not pictured in the image above, but we highly recommend a second testing phase that we call the Pilot stage where the patch is deployed to a pilot group. If patching workstations, this pilot group should be actual users from different departments and locations; that not only gives you the most information but also ensures that, should there be an issue, you don't shut down an entire department or location. Users are encouraged to work normally with the new patch for one to three weeks, to ensure that everything runs smoothly.

3. Deployment stage The patch is then rolled out to all users in waves. You will need to keep an ear out for potential problems, and be able to roll back the patch if a particular group is affected negatively.

4. Analysis, reporting, and (possible) rollback Really, you should be collecting feedback and mitigating after each stage is done, particularly the test and pilot stages. Return to this feedback, and the process overall, after the patch is deployed. You may well learn things that end up causing you to revise your patching strategy.

THE PATCHING PROCESS

This process looks a little different for workstations than it does for servers and data centers. Each process needs different questions addressed-- for example, user behavior matters when patching workstations, but not so much when working with a data center. Then again, uptime and server security are much bigger concerns with servers.

For that reason, we're going to treat each kind of device differently.

PATCH MANAGEMENT STRATEGY FOR WORKSTATIONS

So, you've already identified your goals and metrics, organized your list of patch sources, and thought through deployment rules. If some of devices you identified are workstations, there are some further questions you will need to work through. Many of these have to do with different device connectivity scenarios, as well as the patching point of authority for distribution and reporting.

Step 1: Determine Connectivity (In-Office vs. Roaming Devices)

Some devices in your organization will be in-office, while others are roaming. How will you go about patching devices in each of these groups?

For in-office devices, there are different options. For example, one could use Microsoft's Intune. This gives you a lot of flexibility for managing both in-office and out-of-office devices, but there is less control. WSUS is simpler but, again, does not give you much control or ability to automate. SCCM, by contrast, gives you tons more control…but it requires much more management, and thus may eat away at IT time and resources (unless you hire an outside firm, like us).

For roaming devices, you will need to think not only about services, but also connectivity. For example, you can use SCCM for managing these devices, but you will need to decide whether this will be done via some sort of internet-based client management app or using Microsoft Azure's cloud proxy. Other examples are Windows Update for Business and, again, Intune. (Connectivity, patching, and up-time can be a major issue depending on your particular environment. For example, see our use case with Air Methods for an example where the organization critically relied on a large fleet of mobile devices that had to be updated at specific times.)

Step 2: Decide What the End-User Experience Should Be Like

Unlike patching servers or routers, patching workstations must take into account the end-user experience. There are two general ways of creating patch management policies, depending on the involvement of the end-user. These are the high-visibility and low-visibility approaches.

High-visibility This is a more modern approach to patching where the user is made aware of the need for a patch and is given more control over when the patch is deployed. This is a better approach in environments where a critical device needs to function at certain times (such as in a hospital), or when devices frequently travel in and out of network.

If you choose a high-visibility approach, you will need to determine:

- The cadence and length of the patch release cycle
- What end-user messaging is going to be used to inform and remind the user of the patch
- The frequency of the reminder messaging
- Test reminders and messaging
- Messaging indicating success (or problems)

A great tool for a high-visibility approach is our own Update Notification User Interface (UNUI). UNUI is an application that detects available updates or pending reboots stemming from updates or applications deployed via SCCM, and provides the end-user with the ability to install or reboot at their leisure—all while providing a maximum time limit to ensure security requirements are met. It's a great way to put patching in the users hands while keeping up with security and compliance requirements. (You can read more about this and our other products here.)

Low-visibility The low-visibility approach leaves patching to run quietly in the background. The advantage of this is that it does not disrupt workflow (assuming that everything goes according to plan) and takes less work to develop messages, etc. The disadvantage is that it removes the feeling of control from the user. This especially can make users upset when something goes wrong.

Thus, a big aim of the low-visibility approach is to ensure that, if something does go wrong, the disruption is minimal. This means determining the proper maintenance window and creating a deployment strategy that starts with the most adaptable users and continuing to the least adaptable.

PATCH MANAGEMENT STRATEGY FOR SERVERS

Patching servers in a data center is a little different from patching your typical workstation or endpoint, and so a slightly different approach is needed. Not only are uptime and server security much bigger concerns, but there are different approaches that one can take.

Step 1: Identify Device Connectivity Scenarios and Point of Authority

"Servers" is almost a misleading term, because there are so many different scenarios these days, and many do not look like the classical server that an IT department would keep in a closet somewhere on prem (though that does count, too). Many IT professionals talk about data centers, but that seems restrictive, too.

Therefore, it helps to think through the different server connectivity scenarios:

In-office/on-prem devices In other words, hardware that exists on site. There are many ways to update and patch these servers: Group Policy Objects (GPOs) through Windows Server Update Services (WSUS), and SCCM to name a few.

To use an internal WSUS server, you will need to configure clients with automated update settings and also configure the server with which they will communicate. Additionally, you can configure the clients to be a member of a specific WSUS computer group, if you're deploying patches in WSUS based on computer group targets.

SCCM handles more than just server updates; it also handles operating system deployment, application infrastructure, software inventory, and more. For a larger organization, it's handy to have all of this capability from a single dashboard. As far as patching servers goes, it's helpful because it gives you more control over the schedule for patching and rebooting, which is really important for mission-critical servers.

In the end, SCCM provides greater flexibility and control for patching servers (and more), giving you more choices, more extensive updating, and better reporting. Still, something is better than nothing, and sometimes organizations have to make hard choices about the degree of complexity they want in their patching procedures. (That said, we can handle a lot of that complexity for you—just ask!)

Servers in the DMZ As a freshener, DMZs are a sort of "buffer zone" between the public internet and your internal network(s). A good example would be web servers that have some public-facing components. Patching servers in a DMZ often requires careful management of firewall rules and/or placing a WSUS server in the DMZ.

Standalone cloud devices Cloud servers are sometimes standalone, and so would have little-to-no network connectivity to production management systems. That being the case, using SaaS tools such as Microsoft's Intune, or OMS, would be the way to go for updates. Given that they have support for multiplatform and are network agnostic, these tools would be preferred point of authorities over SCCM or group policies, as these on-prem tools simply won't work for this scenario.

Step 2: Identify Production Deployment Strategy

Patching servers effectively will require a standard deployment process according to known business rules. This deployment process follows the same stages as discussed in the chapter "The Patch Management Process", including test, pilot, deployment, and assessment stages.

Also be sure that your production deployment strategy works in the necessary feedback loops. Verifying success early on—or catching problems—depends on it.

Step 3: Determine Server Patching Groups

Patching groups can be defined in many ways; some of the more common ones include:

- Server owner
- Service (application hosting, communications, web hosting, file serving, and so on)
- Criticality
- Redundancy with other devices

Servers will need to be added to groups once you have your deployment strategy settled. Be sure that each server is in a group!

When dealing with multiple servers, first identify any dependencies that might require a certain server reboot order; else, you might find that a given server can't access needed data (for example) even though the patch works just fine. For example, it is best practice to bring down a multi-tier system by starting with the presentation tier (web server), then the application tier, and then finally the database tier. These systems should then be brought back up in reverse order.

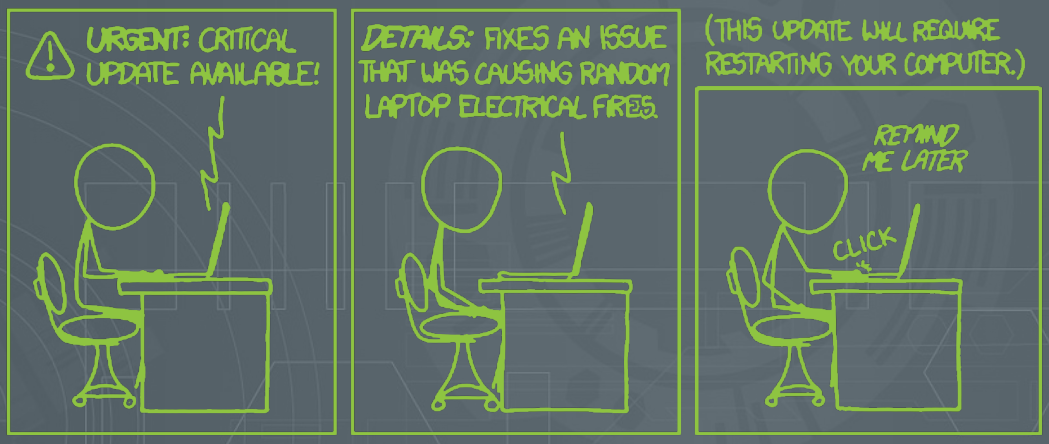
Step 4: Defining Patching Schedules/Maintenance Windows

Maintenance windows are configured on a per-collection basis and consist of a start time, end time, and recurrence pattern. Servers in larger organizations tend to have their own maintenance windows already defined; if that is not the case, you will have to do so. The frequency of these windows depends on the cadence that the organization needs to set, which in turn depends on the SLAs for the business.

Why SLAs? These often have parameters for an agreed amount of uptime. Critical functions in some industries, for example, need to be up as much as possible. Credit card processing is a good example: The servers processing transactions need to be up almost all of the time, and they cannot afford frequent outages for patching. Therefore, they have a different cadence than most. The same would go for critical facilities, like a hospital or a power plant.

So check your organization's SLAs, note any language that promises a certain amount of server uptime, and set your patching cadences accordingly.

EMERGENCY PATCH DEPLOYMENT



The previous chapters assume that you are starting a patch management strategy more or less from scratch, and that you are taking the time to get things right. Of course, this isn't always the case.

Specifically, there are times when you might need emergency patch deployment. In these cases, you do not have the luxury of a full 22-day test-and-pilot cycle. And you might have to manually package and deploy the patch, too.

If you find yourself in that situation:

Prioritize systems that need patching by how critical they are, and the potential severity of the consequences of not patching. Workstations with mission-critical apps that present a clear vulnerability would be high priority, obviously, whereas a workstation at lower risk can wait—even if a known risk has been plastered all over the news.

Run your test on systems most like the high priority ones you identified. Testing on a clean workstation that looks nothing like your high-priority endpoints doesn't tell you much.

Test systems post-reboot. Don't just reboot a patched system and pat yourself on the back when the user logs in. Test out applications and make sure everything works. Be ready to rollback the patch if the system becomes unusable.

Have a plan in place for emergency rollback in case something does go wrong.

What about servers and data centers? Like workstations, servers will sometimes need emergency patching as well, and the process looks largely the same: Start by prioritizing systems by how critical the services being provided, the disruption that will be caused by patching/downtime, and the potential severity of the consequences of not patching. Test on a low-priority system before deploying to high-priority ones. Follow up, and be ready to roll back if there are any issues.

AUTOMATING PATCH MANAGEMENT

Why is There Debate?

If you want to see a fight break out at an IT conference, ask a room full of people whether or not to automate their patch management. While it might seem that automation can save time and man-power (which, recall, keep a lot of enterprise organization from patching in the first place), there are those that argue that automated patching can introduce just as many problems as they solve.

We'll be the first to admit that some organizations have been burned: They jumped in with an automated solution and, because it was not set up correctly in accordance with a patch management strategy, there were problems. Sometimes severe problems.

Here's the thing: **Patch management itself does not vary much from industry to industry, or environment to environment.** So, once you know the best practices and have refined the process, it can be automated and deployed easily.

Ah, but there's the trick: Using a refined process.

Take sever order, discussed in the chapter on patching servers. If one deployed an automated solution to patch a multi-tiered server structure without taking into account boot order, it will lead to problems. And it might take an IT department hours or days to figure out what went wrong.

Another example: Deploying a patch to an entire workgroup without thinking through exceptions. We once worked with a law firm that used a legacy plug-in for document management—a critical application in their business. A new patch was deployed that was incompatible with the plugin, and things ground to a halt. Again, automation wasn't so much the problem, as was the failure to think through the various use cases.

However, if the needed business rules are well defined and ready to put into place, an automated patch solution can be deployed quickly, in as little as an hour. From then on, patch deployments takes no more than a week. That's a time-savings that most organizations can get behind.

Even better, automating the process means that IT staff no longer has to manually babysit the patch management process, freeing them to do other, more strategic and profitable things. Business continuity can be maximized even while vulnerabilities are addressed.

How to Automate

Most organizations use a 3rd party tool to help manage and automate patching. Many of these integrate well with both WSUS and SCCM.

To be honest, buying such tools outright is often unnecessary. Microsoft's own tools are up to the job, but they need to be expertly configured. Issues with automation don't go away by throwing more automation software at the problems. They go away by following best practices. (Which, of course, is why we wrote this guide.)

If you are ready to explore your options when it comes to patch management automation, just reach out to Model.

WHO WE ARE | MODEL TECHNOLOGY SOLUTIONS

AT MODEL, WE ARE INSPIRED BY THE ADVANCEMENTS WE'VE SEEN IN BUSINESSES THROUGH MICROSOFT AUTOMATION TECHNOLOGY.

Automation fascinates us, and helping businesses achieve their objectives through automation is our professional reason for being. Our team has invested years of our lives studying and perfecting the art of automation, and we are excited to share our expertise to help you see the breakthroughs for your business.

Just as equally, Model is passionate about cultivating a strong IT community. We know that our brand is a reflection of our people, and therefore we are committed to cultivating an empowered, champion technologist culture. And, beyond the Model team, we are invested in contributing to the growth of Microsoft technologies and the world of innovators who support it.

Model-technology.com