Patching the Enterprise

Organizations of all sizes are spending considerable efforts on getting patch management right—their businesses depend on it.

Software patch management has grown to be a business-critical issue—from both a risk and a financial management perspective. According to a recent Aberdeen Group study, corporations spent more than $2 billion in 2002 on patch management for operating systems.[1] Gartner research further notes the cost of operating a well-managed PC was approximately $2,000 less annually than that of an unmanaged PC.[2] You might think that with critical mass and more sophisticated tools, the management cost per endpoint in large organizations would be lower, though in reality this may not be the case. The objective of this article is to provide some rationale—drawn from enterprise experience—to put these observations into context and present some approaches that could be useful to combat that trend.

There are a few main themes worth noting. The first and probably most important is that *patch management is a team effort*. In a large enterprise, many departments need to work together to correctly assess and remediate software vulnerabilities through patching. Good tooling can help, but enterprises can't succeed without establishing a well-defined process and good communications among the various teams involved.

GEORGE BRANDMAN, MORGAN STANLEY

Patching the Enterprise

A second important theme is that *no single tool will solve all problems*. For a company with tens of thousands of endpoints (e.g., desktops, laptops, servers), complementary systems are needed to effectively manage the patching process. Depending on the complexity of the internal network, vendor-developed tools may not fit neatly or completely address all needs, requiring some amount of custom development.

Patch management is reactive, good security practices are proactive. It may seem obvious, but it's easy to get caught up applying waves of patches without taking time out to engineer the tools and configurations that can reduce or eliminate certain vulnerabilities before they become exploited.

WHERE WE ARE TODAY

Most enterprise-quality patch management systems follow some of the same basic processes. In a nutshell, patches and the policies used for determining their applicability are first downloaded from a vendor's secure Web site to the internal corporate network. Next, computers are targeted to receive the package that contains the patch and corresponding policies at a scheduled time, or coincident with a system event such as user logon. The policies then determine whether the patch is applicable for the computer, and install it if appropriate. Finally, the computer will often need to be rebooted to complete the patch installation before it reports back to a central repository that the patch has been successfully installed. A simplified, typical architecture is shown in figure1.

Some of the main advantages of an enterprise-caliber system include sophisticated targeting, scheduling, reporting, and inventory capabilities. All of these are essential when working with a very large number of computers and thousands of installed applications.

Targeting typically involves selecting a specific set of computers to receive a package using some meaningful grouping within the organization. This could be a couple of floors occupied by a friendly department at first, scaling up to entire business units or campuses. Good patch management tools provide flexible mechanisms for targeting. This can range from LDAP queries into Active Directory for organizational units or mail groups, to looking at specific inventory records about a computer's hardware or software from connections to built-in or external databases.

Once a patch is delivered to a targeted computer, various automated mechanisms can be used to determine its applicability. The most common include looking for some predetermined footprint composed of files, DLL versions, and/or registry settings. As an example, Microsoft provides this capability with its Windows Update and MBSA (Microsoft Baseline Security Analyzer) utilities. There are situations, however, that are not covered by these tools. In addition, these utilities apply only to certain Microsoft software products and not to other vendors' products, such as Adobe Acrobat or Mozilla Firefox. In the "Getting it Right" section later in this article, we discuss some of these considerations in greater detail.

Scheduling identifies the necessary timing and system conditions for applying updates. In the most basic form, jobs would be given a calendar date and time to run. While this might meet the needs of a smaller organization, it wouldn't cut it for a global financial services firm with many business units, each with unique and time-

Patch Management Architecture

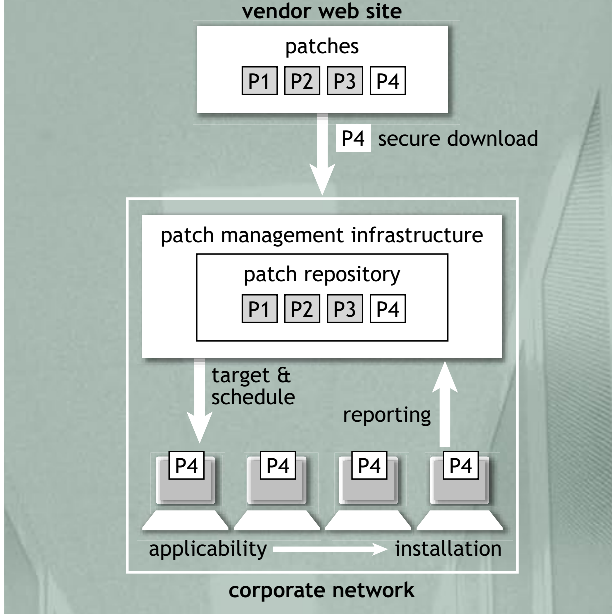

FIG 1

sensitive demands. For that type of organization, additional capabilities, including the following, are required:

- Support for optional and mandatory windows to apply patches.
- Detection of the current user and system state (e.g., is the user idle?).
- Ability to determine if and when to reboot.
- Flexibility for the user to defer or refuse the update under certain circumstances.

Good patch management systems should also support a concept of tiered scheduling, where unique deployment cycles can be created for different categories of patches. For example, patches placed in a critical bucket could be applied on a weekly (or more frequent) basis, while others would be applied only monthly.

Timely and accurate *reporting* from managed computers back to a central repository is a critical part of the patch management process. When working with a large number of endpoints, the data collection process can become onerous. A tiered, fan-in architecture can assist in protecting the central repository from being swamped by thousands of computers attempting to report status simultaneously. This is also an area where agent-less patch management systems (i.e., those with no local service running on the managed computer) may end up with a disproportionate amount of the work being done on the central servers collecting and processing results.

Inventory of endpoints, both managed and unmanaged, is the ongoing process of cataloging the computers on the corporate network. It includes tracking important hardware, application, and operating system characteristics and is used for both targeting patches and maintaining overall system health. The process can be resource intensive, so there needs to be a balance: collecting enough information to manage the environment appropriately, but not so much that the end-user experience is negatively impacted or important data gets lost in the tall grass. Some mechanism for collecting information about unmanaged endpoints should also be available to enable actions that mitigate their risk to the corporate network.

KEY CHALLENGES

Enterprise patch management costs and efforts may increase disproportionately with the scale and complexity of the target environment. Less-sophisticated tools and processes suitable for smaller organizations may not be appropriate for larger and more heterogeneous environments. Several factors discussed in this section may also have significant bearing on the overall effort required to maintain a healthy and secure environment.

REBOOT DISRUPTION

One of the biggest roadblocks to timely patching continues to be required reboots. This occurs in multiple vendor products, including Windows operating system patches, Internet Explorer, and other products and device drivers that operate in the kernel space. Some patch management systems provide the capability to install a patch, while deferring the reboot until a later time. While less intrusive to the end user, this practice can leave the system in an inconsistent state.

RATE OF ISSUE OF PATCHES

The need to reboot between updates can make patching schedules for a large organization resemble a block of Swiss cheese. Each business unit may have different considerations about when and where their users' machines may be rebooted. Some may restrict midweek reboots, others may have critical weekly or monthly processing that must be considered. Servers bring their own unique issues, as many line-of-business applications are 24/7, with no opportunity for a maintenance window.

PATCH MANAGEMENT SYSTEMS— ONE SIZE DOES NOT FIT ALL

Some of today's enterprise patch management systems may provide either incomplete or inconsistent policy definition capabilities for complex scenarios. Large organizations must be sensitive to date and time requirements, patch criticality, network connectivity, and machine state.

MOBILE USERS COMPLICATE THE PROBLEM

An increasingly mobile user base and complex corporate network composed of DMZs (demilitarized zones), remote laptops, and locked-down end-user and server environments all present unique challenges for keeping systems patched. Patch packages and delivery systems must be engineered for persistence and flexibility with regard to connectivity and various security configurations.

GETTING IT RIGHT

In a large-scale financial services environment, taking too long to roll out a critical update can leave the enterprise vulnerable to an attack. That being said, deploying with insufficient testing could lead to, for example, the breakdown of line-of-business applications on the trading floor during market hours.

Getting it right is the balance of applying patches in an extremely timely way, while minimizing user impact from reboot disruptions and a continuous stream of updates. A well-coordinated team effort with representa-

Patching the Enterprise

tion from a number of different disciplines is required to execute effectively:

- Operations teams may be the ones to package, centrally test, and deploy the update.
- Engineers integrate it into the build and patch management systems.
- Line-of-business support teams ensure that business-critical applications are not impacted.
- Security teams assess the criticality, set the overall deployment schedule, and potentially prescribe defensive actions (e.g., shutting ports on the corporate firewalls).

All of this takes time. Some patches may need to be repackaged to install and configure correctly in the corporate network. One or more test deployments may be scheduled prior to a full global enterprise rollout to incrementally prove both the compatibility of a patch and the soundness of the package for installing it. After each iteration, the result is evaluated for any signs of larger problems that may loom with an expanded deployment. This can include unforeseen end-user impacts or hidden interdependencies with other products. Packages may then need to be revised to detect previous versions of incompatible software and either uninstall them or install different components applicable to the version detected.

MANAGING REBOOT CONSIDERATIONS

With some vendors releasing patches monthly, one way to minimize user impact is to create a package containing multiple required hot fixes. One or more of these patches might require a reboot. A tool for chaining hot fixes together can be used to minimize this disruption and the need for multiple reboots when installing several patches to a single, final reboot, though there may still be updates that require reboots between patches.

In a deferred reboot scenario, some files are overwritten to their new versions while others (such as system DLLs) are staged and waiting for the next reboot to be activated. Organizations need to weigh the risk of this "patch and reboot later" strategy vs. the impact of a system reboot to the end user. Using the former to more quickly deploy patched versions of binaries and libraries may present the perception that the endpoint risk has been mitigated, when in fact it may not have been if some components are waiting for a system reboot to be activated.

PULLING THE TRIGGER

At some point—after the centralized and end-user testing, pilot deployments, and validation of results have been completed—it's time to do a global rollout. Typically, unless there is an immediate threat, as was the case with the SQL Slammer virus, the rollout can be initiated over the course of a weekend to minimize user impact.

A policy defining the specifics for applying the patch (or package of patches) is usually implemented considering the following:

- Business hours at the endpoint.
- Whether users will be allowed to defer the installation or reboot and, if so, for how long.
- Appropriate user or machine states at the time the patch is applied.

Stitching together a schedule of acceptable patching windows across many time zones is often difficult. Patch criticality should also be considered when determining whether users will be allowed to defer the update.

A sample flow, including some of the decision points that might be included in a laptop patching policy, is shown in figure 2.

The process of collating results begins shortly after the initial mass enterprise rollout has completed. Tens of thousands of endpoints will report back to the central repository via a post-patch verification process. This typically includes similar checks for file versions or registry keys as were performed to determine applicability before the patch was installed.

Automated cleanup cycles can then be scheduled to address the remaining machines that were not successfully patched initially. Machines with health problems or those that were off-network when the deployment was originally scheduled can contribute to lowering initial success rates. Problems such as an inoperable patch management agent on the endpoint can be particularly troublesome, as they might require a desk visit to remediate the issues causing installations to fail.

Maintaining a well-defined, common base build and minimizing drift between end-user configurations can help reduce the overall number of machines with health problems and demonstrates good, proactive security practice. The removal of local administrator capabilities from as many end users as possible (often more difficult to achieve with developers' machines) and the implementation of policies to consistently decommission older and more problematic product versions can also help enterprises progress toward these goals.

ADDRESSING THE NEEDS OF REMOTE AND MOBILE USERS

Patch criticality may be considered relative to the end-user environment, where both remotely connected and mobile user needs must be addressed. Certain patches may be deemed mandatory for LAN-connected users, but optional for remote or mobile users. An endpoint may be based on the standard corporate base build and configuration or an out-of-the-box vendor operating system installation. In either case, VPN-connected endpoints may have certain protocols restricted, such as SMB (server message block). Distribution tools should also support the protocols common to basic Internet connectivity, including HTTP/S and FTP.

PUSH VERSUS PULL

Many current software distribution products are capable of both push-based software distribution and pull-based polling for software patch updates. Both of these methodologies have a place in an enterprise-grade software distribution system.

Push-based deployments are positioned to ensure that a specific patch gets deployed to a large number of machines at the same time. This is well suited for emergency patches and meeting certain deployment window restrictions. The downside to push methodology is that target machines must be reachable on the network or have elaborate retry logic to compensate.

Pull-based deployments do not typically require an endpoint to be reachable via the corporate network to receive a patch. Simple polling algorithms, LAN-port link status, and system reboot can all cause a pull event to take place. This enables machines that become available on the network to immediately poll the distribution servers to see what they may need to become current. An additional benefit of pull-based technology is that it can be used as a mechanism for endpoints to initially register themselves as clients of the patch management system. A downside to pull methodology is that all machines must reach their polling interval before it can be assured that a patch has been deployed.

The most robust software distribution systems have both concepts coupled together with fine-grained endpoint targeting control. Deployment packages are made available to all systems matching some well-defined inventory criteria or group association. A "push" is then initiated to all matching clients with each client asked to "pull now." Targeted machines not on the network will pull whatever is needed at their next polling interval, using the appropriate connectivity option at the time.

Patch Update Policy Flow

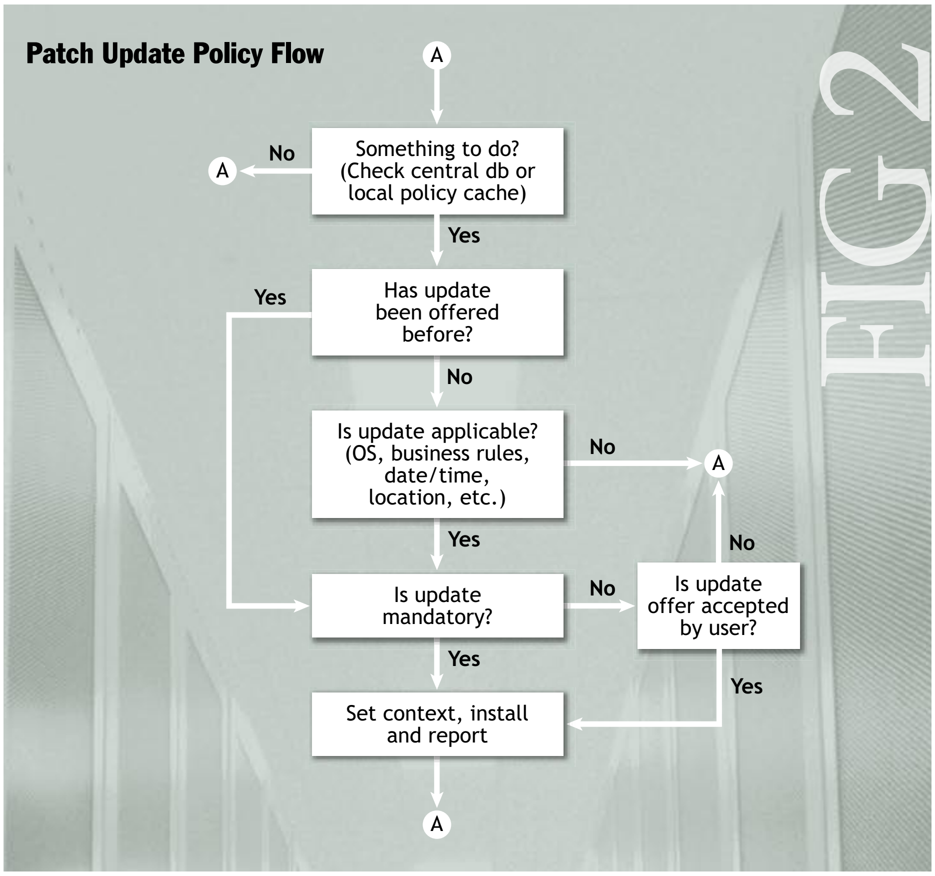
A
Something to do? (Check central db or local policy cache)
No
A
Yes
Has update been offered before?
Yes
No
Is update applicable? (OS, business rules, date/time, location, etc.)
No
A
Yes
No
Is update mandatory?
No
Is update offer accepted by user?
Yes
Yes
Set context, install and report
A

FIG 2

PRIORITIZING TOOL CAPABILITIES ACROSS ORGANIZATIONS

Maintaining a reasonable state of health and security for a diverse computing environment can be a challenging task for a patch management system. It may be difficult for a single tool to meet the challenges of every organization cost effectively. Influencing factors may include the types and numbers of targeted endpoints and capabilities of existing internally developed infrastructure. Some tools may have excellent reporting and targeting capabilities. Others may excel in secure transmission of updates across the Internet or integration with corporate software repositories.

Each enterprise may prioritize certain capabilities of a product depending on its size and connectivity needs. Large organizations may have internally developed infrastructure for content replication and software deployment, placing more emphasis on well-documented vendor APIs and integration capabilities. Smaller organizations may derive more value from a completely turnkey solution. Agent-less products focusing on ease of installation and deployment with minimal server requirements may be a good fit for these enterprises. Global organizations often create geographic hierarchies for centrally managing infrastructure with one or more "root" hierarchies and regional or campuswide sub-hierarchies. Current product architectures that support fan in/out capabilities between the content and policy source and the target endpoints may be well suited for this type of management. Support for inheritance and override of policy, including targeting at all levels of the hierarchy, is also extremely valuable. Ultimately, several best-in-category products may need to be considered by an enterprise to best meet the diverse set of capabilities required.

STAYING PATCHED

Sometimes a new version of a vendor product may downgrade a common component to a level below the latest patched version. An enterprise may be able to address this through a centralized software packaging team or standardized tools that compare the package with a current version of the base build configuration for the targeted end-user environment.

Ongoing inventory and reporting can also be effective to ensure that endpoints stay patched. Mechanisms exist at the vendor product, operating system, and hardware levels to perform comprehensive inventorying. WMI (Windows Management Instrumentation) exposes inventory and management information stored by OEMs using the DMTF CIM (Distributed Management Task Force Common Information Model) specification. An enterprise may use these APIs, as well as direct querying of hardware where appropriate, to gather required system inventory information. Checking machine registry and file system information to verify patch signatures and updating this information in a central repository should be an ongoing process, in addition to one that takes place immediately following the initial patch deployment.

With some of the more recent fast-spreading viruses, it can be a challenge to prevent a newly built machine from becoming infected *before* the automated build process can apply current patches. One way this can potentially be mitigated is to *slipstream,* or inject directly into the build process, the updated components so they are installed as a part of the automated base build, rather than being applied on top. Unfortunately, this process can significantly add to the cost of maintaining the base build and is not necessarily appropriate for all types of patches.

Virtual machine technology typically provides checkpoint capabilities that can be very useful for testers and developers. The user can take a snapshot of the system state and then quickly roll the virtual machine back to this state on command. A problematic side effect is that if not properly managed, users can actually unpatch a machine on the corporate network through this action. Engineering efforts may be required to ensure that this capability can be enabled safely.

WHAT'S NEXT?

Large enterprises and the current breed of patch management systems have become extremely sophisticated in their approach to laying down patches on all types of machines including laptops, desktops, and servers. As discussed throughout the article, the costs of keeping machines patched and healthy can also be minimized through the use of automated build and configuration management systems.

So, with this technology already in place, what more can be done to achieve significant improvements with respect to the cost, reliability, and speed of managing an enterprise environment? One possible answer may lie in a fundamental change in the way we think about managing endpoints. What if enterprises were effectively able to run all of their business applications and large parts of the machine's operating system from a network file system? In this model, executable files and configuration information would be sourced from read-only areas on the network, then run locally in memory resident on the endpoint.

Deploying updates to a few hundred file servers is significantly less time-consuming and error-prone than physically installing those same updates on tens of thousands of machines. This would require new ways of abstracting stateful installs, in addition to mechanisms for managing different operating system and application configurations in the network namespace. This may not mitigate the need for reboots to activate a configuration change in some cases, but the benefits are considerable.

Rather than patching every system by physically installing a piece of software onto it, the patch would need to be installed and tested only in the network namespace, then the target machines would be pointed to the new configuration. Testing updates on a handful of end-user configurations in this environment would provide significantly greater confidence in the outcome, since configuration drift on the targeted machines would be at an absolute minimum. Managing the remaining local configuration drift on endpoints could be accomplished through policy-based capabilities that proactively restore critical settings and files as they are changed or removed. Global rollout and rollback, if necessary, could be accomplished in a fraction of the time that it takes today.

RUNTIME ENVIRONMENTS AND LOCAL CACHING

Java and Microsoft .NET development technologies are engineered to address run-from-network considerations. Older Microsoft technologies such as COM are possible, but more difficult, to abstract to this model. From a system level, Posix-style operating systems provide certain capabilities to make both the application and operating system run-from-network scenarios more viable.

While desktops or servers can generally count on a reliable broadband connection for retrieving their application software and operating system configuration, laptops are often off-network and need to be functional to their end users even while disconnected. For these scenarios, some form of intelligent caching would need to be available that provides local capabilities while disconnected, but automatically updates the cache when the corporate network becomes available.

Laptops need the capability to be patched across various types and qualities of network connectivity including LAN, VPN, or simply an Internet connection. All of this needs to integrate with the local cache management on the machine and appear seamless to the end user.

OTHER MANAGEMENT POSSIBILITIES

The corporate network itself could play a role in patch management. In addition to file servers, network nodes such as routers or gateways could actively work with endpoints such as desktops or servers to detect and remediate security or other types of issues. Additionally, some vendors are beginning to offer management capabilities that provide more granular, centralized security configuration controls in certain areas.

For many corporate end users, high-performance desktops aren't explicitly required. For these users, providing thin-client access through terminal emulation to a VM (virtual machine) or terminal server could dramatically reduce the number of endpoints to be managed. The back-end VM or terminal servers could still take advantage of all the previously described capabilities offered through a run-from-network model. An additional benefit of this model is the higher-bandwidth connections typically available from these servers to the network file servers as compared with that of an end user's desktop.

PULLING IT ALL TOGETHER

Technologies available today and those coming down the road will continue to simplify patch and systems management. The ongoing integration of automated security practices will ultimately blend seamlessly into these tools. Vendors in networking and platform technologies are already beginning to work together toward this goal. New approaches may be in order to rethink the way large corporate computing networks are configured and change managed for the greatest efficiency of scale. Regardless of the approach used, a practical balance of risk management will remain a key ingredient requiring the thoughtful coordination of disciplines across the organization. Q

REFERENCES

1. Schwartz, K. D. Patch management grows up. June 3, 2004. The Channel Insider; http://channelzone.ziffdavis.com.
2. Silver, M. and Pescatore, J. October 14, 2004. Security holes increase Windows client TCO. Gartner Research Note; http://www.gartner.com.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

GEORGE BRANDMAN is an executive director in Morgan Stanley's Institutional Securities Information Technology Department.