

# The Challenges and Successes of Implementing an Enterprise Patch Management Solution

Tom Gerace  
Tulane University  
A. B. Freeman School of Business  
New Orleans, LA 70118  
(504) 865-5651  
tom@tulane.edu

Jean Mouton  
Tulane University  
A. B. Freeman School of Business  
New Orleans, LA 70118  
(504) 865-5040  
jmouton1@tulane.edu

## ABSTRACT

With the proliferation of security threats we are required today, more than ever before, to effectively manage computer systems. Computers that we are responsible for in labs, classrooms, and faculty and staff offices require constant attention as manufacturer-supplied program patches and updates become available. Missing a critical update or patch in your networked environment can spell disaster or cost countless hours of "cleaning up" after an errant worm wreaks havoc on unpatched machines.

Microsoft has done a good job of publishing operating system patches and critical updates; however, actually updating all of the machines in the academic environment can be a daunting task. While we can adequately use Windows Automatic Update in the homogeneous lab and classroom environment, the procedure can be circumvented by users in their offices. And while user education seems to be effective at first, system updates soon take a back seat to users' regular work and research tasks.

This paper discusses an enterprise update strategy that was adopted by Tulane University's Freeman School of Business. We will discuss the challenges associated with updating hundreds of workstations in labs, classrooms, and faculty and staff offices. We will look at several patch management software products that we reviewed and discuss their relative merits. We will then discuss the selected product and its implementation in the current production environment, including the user education involved and the political issues surrounding "touching" desktops. Finally, we will discuss the successes that we realized and the administrative challenges of ongoing patch management in a complex, enterprise environment.

## Categories & Subject Descriptors

D.4.6 [Operating Systems]: Security & Protection – *invasive software*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGUCCS '04, October 10–13, 2004, Baltimore, Maryland, USA  
Copyright 2004 ACM 1-58113-869-5/04/0010...\$5.00.

## General Terms

Management, Design, Reliability, Security, Human Factors

**Keywords:** Operating System, Update, Patch, Patch Management, Security, Automatic Update, HFNetChk, HFNetChkPro, SUS, Microsoft

## 1. INTRODUCTION

Information technology staffs today are scrambling to find ways to apply critical and security updates to all computers that they are responsible for. For these IT professionals, staying up-to-date with security patches is a necessity because of the recent well-publicized vulnerabilities in the Windows operating system. Given the difficulty in manually keeping every Windows machine in a department or school updated, a department- or school-wide automatic update solution seems ideal.

The Information Technology staff of Tulane University's A. B. Freeman School of Business researched enterprise operating system update offerings in an effort to combat the ever-increasing threat to desktop and server operating systems and supporting programs. Available options covered the spectrum from allowing the user to handle updates to complex server-based systems that "look inside" clients and determine the updates to be installed. Doing nothing was not an option. Ideally, we wanted a solution that would "push" operating system updates to client computers, would be flexible in scheduling, would allow discrimination in update selection, and would give us control over required system restarts.

This paper discusses the school-wide update strategy that was developed by the Information Technology staff of Tulane University's Freeman School of Business, and the challenges associated with updating hundreds of workstations in labs, classrooms, and faculty and staff offices. We will look at the patch management software products that were reviewed and discuss the selected product and its implementation in the current production environment, including the user education involved and the political issues surrounding "touching" desktops. Finally, we will discuss the successes that we realized and the administrative challenges of ongoing patch management in a complex, enterprise environment.

## 2. BACKGROUND

Tulane University is a private, non-sectarian research university with 11 schools and colleges. Located in New Orleans, Louisiana,

Tulane offers undergraduate, graduate and professional degrees in law, medicine, public health and tropical medicine, architecture, business, engineering, social work and the liberal arts and sciences. The Freeman School of Business has been highly ranked in every major published ranking both in the United States and Europe.

The Freeman School is housed in 2 buildings on the urban campus, collectively called the Goldring/Woldenberg Complex. A campus in Houston, Texas, is a recent addition to the Freeman School and is treated as a logical extension of the University. The Information Technology staff maintains a total of 540 computers across the 2 buildings and the Houston campus, which includes machines in labs, classrooms, and faculty and staff offices. In addition, all of the School's graduate programs have a notebook computer requirement, which brings in approximately 460 additional computers.

As if maintaining application software and network connectivity on these machines weren't enough, the state of security requires frequent operating system patches and updates. Before moving to the monthly distribution of security patches, Microsoft's weekly distribution often caused consternation for the staff, as they had to apply patches frequently. On the other hand, Microsoft's decision to move from weekly to monthly software patches has increased the importance of staying updated in a timely fashion since several vulnerabilities are addressed by a single patch.

An automatic update strategy providing more functionality and flexibility than the Windows Automatic Update, built into the operating system, was required to keep our servers, our network, and the University network secure. At the project's outset, we were careful to note that many computers were entirely under our control (labs and classrooms with definite closing times) while some were not fully under our control (faculty and researchers who may work anytime). We were also sensitive to the possibility of negative feedback with respect to "touching" office computers.

### 3. CHALLENGES

Microsoft's Automatic Update built into the Windows operating system is effective in a controlled environment such as a lab, but can prove to be problematic in other venues. In classrooms, automatic update, if not configured properly, can launch during a class period and disrupt a presentation. In faculty and staff offices, the product can be interrupted or disabled by the user. Even with a carefully-targeted education campaign, end-users were not effective in keeping their machines updated with Windows Automatic Update.

Historically, no matter what update process was attempted, the Freeman School's Information Technology staff seemed to always be pulled back into the tried-and-true resource-intensive update process of running around and updating machines. What we needed was an automatic update strategy that would allow us to apply patches school-wide. Products and procedures were researched and a strategy was devised. The effects on end-users were taken into consideration, as was the politics of "touching" faculty and staff machines in the relatively open University computing environment.

The challenge was to find a product and develop a process that fulfilled our update requirements while maintaining a set of

parameters that we had developed regarding the process. We looked for a product with the following set of capabilities:

- Able to locate computers in our network
- Allow updates and patches to be downloaded and tested before being pushed out school-wide
- Allow updates to Microsoft Office products and key back-end products
- Allow the creation of groups in order to segregate computers in labs and classrooms from computers on faculty and staff desktops
- Allow different update configurations and update times for the defined groups
- Allow restart after update to be deferred, thus eliminating the interruption of work in progress, such as a long-running statistical analysis program
- Has good reporting capabilities for both discovery and post-update reporting

Testing available updates in a controlled environment was an essential requirement. We wished to first download updates and patches and install them on machines in a test environment. Using this procedure, we are able to determine the effects of any given update or patch on the operating system or other programs in order to better prepare for any post-update support.

Grouping computers is important in our environment in order to control the updates that are pushed to machines, the time of day that the update will occur, and the post-update behavior. For computers in labs and classrooms, we wished to push the updates out after closing time and then force a restart of the unattended machines. In faculty and staff offices, we wished to push the updates out after closing time but never perform a restart, because some faculty and researchers start long-running processes before leaving the office at night in order to have the results waiting for them upon their return in the morning. In order to ensure that these processes remained uninterrupted, we wished to specify that the update process would not perform a restart; we would rely on the user to restart the computer to complete the process.

Reporting functions in a product were important in the decision process. We found it desirable to have a product "discover" the condition of updates and patches on the computers in our network through a scan process while not actually performing those updates and patches. Of course, post-update reporting is important in order to assess the effectiveness of the update process.

Surveying the available tools, we discovered that not all solutions met our requirements. Windows Automatic Update worked with limited success in labs and classrooms. Not having the ability to test updates prior to installation was problematic. In addition, a missed update in a classroom, which can be caused by the machine being off during the specified update time, sometimes resulted in an update occurring during a class, interrupting a presentation in progress. Windows SUS was a potential solution, but only allowed specifying a no-restart after the update process on Windows Server 2003, Windows XP SP1, and Windows 2000 SP3 operating systems, causing problems in our mixed environment.

Windows Automatic Update allows only 3 modes of operation: notify before downloading updates and notify again before

installing them on the computer; download the updates automatically and notify when they are ready to be installed; automatically download updates and install them on a specified schedule. The first two notification options were typically ignored by most users. The third option was not feasible due to the possibility of interrupting programs or processes on faculty and researcher computers.

In addition to providing operating system updates, we also wished to find a product that would handle Microsoft Office updates as well as updates to other key user and back-end applications. Windows Update does not offer this feature, nor did the other Microsoft update solutions.

#### 4. IMPLEMENTATION

The solution that we selected is HFNetChkPro 4 from Shavlik Corporation. According to Shavlik, HFNetChkPro is built upon the same engine that powers the Microsoft Baseline Security Analyzer and the SMS Feature Pack, and is driven by the same database schema used by the Microsoft Security Bulletin website. The similarities end there, however, as HFNetChkPro provides the flexibility that we required and includes several important operational highlights that make the product a good choice for our labs as well as our staff and faculty computer population.

Some of the HFNetChkPro features that made it the better solution for our environment are highlighted in the following list. Since Windows Update is the predominant update solution available, we offer comparisons with HFNetChkPro where appropriate:

- HFNetChkPro requires no agent, or installed program, on target machines, while Windows Update does. Agentless patch management allowed us to streamline the rollout of our automatic update strategy and allows us to see rogue machines on the network through the scanning process.
- HFNetChkPro covers all security updates, regardless of criticality rating. By comparison, Windows Update focuses only on Critical security updates and does not typically cover Low, Moderate, or Important security updates.
- HFNetChkPro handles updates for Microsoft SQL Server (and MSDE), Microsoft Exchange, ISA, NT 4, and the Microsoft Office products. Windows Update handles only operating system updates.
- Reporting features in HFNetChkPro give us the flexibility to scan all machines in the network and see installed and missing patches and updates, or scan for a single patch in the domain and see the machines that have the patch installed and those that do not.
- HFNetChkPro allows us to group machines (i.e., faculty, staff, PhD, labs, classrooms, servers) providing the flexibility to granularly control the patches or service packs that are pushed as well as the restart behavior. We specify restarts in labs and classrooms, but not on faculty, staff, or researcher desktops.

The IT staff set up a patch management server to hold the updates and then push them out to the specified computers. Initially, computers in labs and classrooms – those machines that we have

complete control over – were targeted for push updates. We realized immediate success. From the start of the process, none of the targeted machines were affected by the well-publicized vulnerabilities for which patches were made available.

After the successes in labs and classrooms, we wished to move to full-scale implementation, including all faculty and staff desktops. Prior to beginning the full-scale effort, we worked to ensure that there was a complete understanding of the need for the process and sought school-wide end-user cooperation by enlisting help from the top of the organization. The potential political challenge of “touching” end-user desktops was of particular concern. Surprisingly, however, no negative feedback was garnered after the initial school-wide announcement of the impending process.

To get started, we first launched a user education effort. We sent announcements to all faculty and staff once again detailing the importance of operating system updates. The announcements outlined the plans to begin the updates, detailed how the users would be affected, and asked the user community to take a few simple steps to help in the update process. End-users were requested to modify their workday routine by ensuring that they left their computers on all the time, thus allowing the update server to see them. We asked users to restart their machines daily, usually at the end of the day, in order to complete the update process (if required), which also left their machines in a secure state with the network logon screen displayed. The reaction among the approximately 130 staff and faculty was positive. Most were happy to be relieved of the onerous update process.

Full implementation across the school was highly successful. No user machines were negatively affected by the updates. We could scan at any time to determine the state of updates and patches school-wide. The few machines that required attention were identified and handled on an as-needed basis.

#### 5. SUCCESSES

The enterprise automated patch management process created by the Freeman School’s information technology group has been a great success. Since beginning the process in February, 2004, we have had no vulnerability breach, and machine scans conducted by central computing have resulted in no reports of unpatched machines within the school. Our first major success was realized with the availability of the MS04-007 patch, when all machines within the school were updated within 3 days of the patch becoming available. In mid-April, 2004, the MS04-011 patch, which included protection from the Sasser worm, was pushed overnight to 332 computers needing the update. Subsequent patches and updates have been applied school-wide in similar timeframes.

Even with this tremendous success, we may never reach a state of 100% patched machines. Although we conducted a massive education and socialization process, users may not be in the office on any given day, adjunct faculty may have turned off their machines, and some classroom computers may have been turned off accidentally. These daily hurdles, however, affect us only slightly as we have attained as high as a 97% success rate at any given time.

The importance of these successes is the timeliness of applying the patches and updates. Using automated patch management,

patches and updates are applied literally overnight, while the sneaker-patch approach takes days to complete.

## 6. ADMINISTRATIVE CHALLENGES

While we have realized success in performing school-wide patch management, we have also realized some challenges in administering such a system. Some of these challenges are technical in nature, while others simply involve the human factor in the process.

One of our process requirements was also one of our major initial challenges – the requirement that users leave their machines on at all times. Machines that are turned off obviously will not be found by the scanning process. While some of our users were already working in this mode, most had to be retrained, and change can be hard. Today, only a small number of computers are not found during any given network scan.

Another challenge is the timing of the update process. Daily updates would be ideal if not for user-related challenges. Users may not be in the office for several days, so an operating system restart will not be performed. This can negatively impact the update process since some updates and service packs require a restart for the process to be completed. If a user is away and a restart is not performed, applying additional patches and updates may cause additional operating system problems. Timing the update process so that the potential for non-restart problems is minimized is a challenge much like an approximation game.

Given that at this time Microsoft distributes updates once a month, we have chosen a bi-weekly update process. In our approximation game, we can only hope that users are out of the office less than 2 weeks at a time or that the specific updates required when a user is out of the office do not require a restart. These approximations create a set of assumptions under which we currently operate the update process. The possibility of missed restarts may become less of a problem in future operating systems as they incorporate dynamic update, eliminating the need to restart.

A complex network can present another challenge to the patch administrator. In order to find all machines in the network, a Master Browser has to exist in each subnet. In our network consisting of 6 subnets, at least one machine in each subnet was identified to have the Master Browser process started at all times.

A challenge that will become more pervasive over time is the presence of a firewall on a user computer. Any push solution such as HFNetChkPro has to access the target computer using a network protocol and a specific network port. The presence of a firewall on a user computer with the required port closed will prevent the update server from communicating with that computer. Even with the excellent reporting tools in HFNetChkPro, we cannot determine if a computer was skipped in the update process because it was turned off or because a firewall blocked the communications. Users will have to be trained how to configure the firewall software to let the required communication

protocol proceed, or administrators may be able to use the power of group policy to set the required rule on computers throughout the enterprise.

It is interesting to note that, as we write this in May, 2004, we are anticipating more firewall-related challenges that will actually be self-inflicted by the update process. Microsoft has announced that some upcoming updates will turn on the firewall in Windows XP operating systems. We are already planning how to handle the post-update state of affairs with a combination of announcements, user education, group policy management, and house-calls.

Ongoing challenges include new computers that are added to the network, computers that move and are renamed within the network, computers that are removed from the network, and notebook computers that come and go. For HFNetChk's scanning and update process, all machines must be found in the Windows Active Directory, so maintaining a clean Active Directory for machine accounts is a necessity. Active Directory maintenance is an administrative task whose processes must be understood, if not undertaken, by the entire support staff responsible for computer placement and maintenance.

Handling exceptions, or requests to be excluded from the update process, is a potential political challenge that could hurt the integrity of the update process itself. To date, we have had no exception requests from faculty; we do, however, exclude certain servers and specialized machines that are maintained by the Information Technology staff. We believe that the "extra mile" effort that goes into pre-testing patches as well as the scheduling of updates in the early morning hours has instilled confidence in the process into our users.

## 7. CONCLUSION

In today's unsafe computing environment, we realized the need to implement a process to handle new vulnerabilities when they are revealed so that we can act quickly to scan for vulnerable machines, test patches, and deploy patches or apply workarounds as needed. The challenges associated with updating a department full of workstations in labs, classrooms, and faculty and staff offices demanded that such a process be timely, flexible, automated, and that it work across the enterprise. Finding the right solution required an understanding of the users' needs and the development of a process that fit those needs while allowing us to achieve our goal of securing workstations and the network.

The automated update strategy developed by the Freeman School's Information Technology staff has been tremendously successful. This success comes in part from the availability of the tools used to perform the updates, but it also comes from user acceptance of the procedure. Users, relieved of the duties of updating their workstations, readily accepted the plans and changed their work habits to help it happen. Success today is measured not only in the achievement of a patch rate of almost 100%, but also in how the process keeps workstations, servers, and the network secure with minimal intrusion into the users' daily work regimen.