Vulnerability Risk Mitigation - Patching the Microsoft Windows Environment

Tracy Lynn

GSEC Assignment v1.4
Option 1
Tracy Lynn

Vulnerability Risk Mitigation—Patching the Microsoft Windows Environment

Abstract
Vulnerabilities in Microsoft's products are being discovered with increasing frequency, and those who write code to exploit them are quick to take advantage of the opportunities presented by the defects. Well-known worms like Nimda and Code Red surface in response to previously known vulnerabilities for which known fixes exist. One of the administrator's ongoing challenges in fighting this problem is to maintain current patch level on all servers and workstations under his or her control. Even in very small networks comprised of only a few systems this task is not trivial; in larger networks the operation becomes daunting, given the sheer volume of machines to patch, downtime considerations, and available personnel to perform the updates. Automating this task is critical to keeping the environment protected.

This manuscript discusses procedures for regularly patching a Microsoft Windows environment, beginning with a discussion what vulnerabilities are, how they find their way into developers' code, and why they have become such an issue. The balance of the paper presents a number of options for patching the vulnerabilities, using either freely available tools or products that require purchasing licenses.

The Need to Patch
A vulnerability in an application exists when a defect in its software code is not discovered during testing and is released for production. No software vendor is immune from inserting vulnerabilities in its products. Microsoft in particular has experienced a significant amount of visibility due to defects discovered in its software. While no one knows how many vulnerabilities per unit code Microsoft generates versus other vendors, widespread use of Microsoft operating systems and applications makes it an appealing target for exploit. To make matters worse Microsoft's traditional strategy of building feature-rich products has created software with huge volumes of code. The amount of code to review, coupled with time-sensitive product releases that can hurry an application to the marketplace prematurely have made thorough product testing impossible. Instead, much testing effectively has been left to the consumer.

Estimates of the number of defects or "bugs" in any developer's code range from five to fifteen per one thousand lines of code[1] to one in every ten lines of code[2]. These defects generally do not produce obviously undesirable effects such as compromising

[1] Schneier, Bruce. Secrets and Lies—Digital Security in a Networked World. New York: John Wiley and Sons, Inc., 2000. 210
[2]Humphrey, Watts S. "Bugs or Defects?" SEI Interactive. March 1999. URL: http://interactive.sei.cmu.edu/news@sei/columns/watts_new/1999/March/watts-mar99.htm (18 May 2002).

functionality or performance, yet they all open up the system to the possibility of being exploited. Considering the 30 million lines of code in Windows 2000[3], the most conservative estimate of five defects per one thousand lines equates to no less than 150,000 defects.

According to Humphrey[4] software engineers typically feel the compiler will locate the trivial mistakes and that they need only focus on the design flaws. However, many mistakes are typing errors, rather than poorly designed logic, of which a percentage will get missed by the compiler. Those overlooked errors are considered "syntax-like" defects as they may produce a different result instead of producing an error. For example, typing "=" instead of "==" in C can cause an assignment instead of a comparison. In reviewing his own code, Humphrey found that 9.4% of his syntax errors were syntax-like. He also notes that the most trivial-seeming defects can have the most dramatic consequences. He cites a major manufacturer that found three defects in their production software: 1) an omitted line of code, 2) two characters interchanged in a name, and 3) an incorrect initialization. All together these errors caused millions dollars of damage to the company. Consider another example of a high profile failure that was the result of a software error. In 1996, the Ariane 5 rocket[5] exploded one minute into its flight due to its on-board computer's interpretation of incorrect guidance data, produced by a small computer program attempting to insert a 64-bit number into a 16-bit space.

More than just causing problems with production systems, vulnerabilities can provide the hacker with advantages not normally present had the software been more carefully designed and analyzed for errors. The Nimda worm well illustrates such an advantage. In September, 2001 Nimda rapidly infected thousands of businesses around the world, causing major disruptions. The worm mainly propagated by automatically executing an attachment delivered in an HTML e-mail. Microsoft posted a patch for the vulnerability six months before the virus surfaced. That many chose not to install the patch led to Nimda's rapid spread around the world.

The path to fixing a vulnerability can go one of two ways: through secrecy or full disclosure. Secrecy traditionally has been the method relied upon by software manufacturers. The Computer Emergency Response Team, or CERT[6], founded in 1988 by the Software Engineering Institute at Carnegie Mellon University and the Defense Advanced Research Projects Agency (DARPA), was set up as a research tool for vulnerability reference materials and incident response. This team gathered bug reports and notified the affected vendors, who presumably would develop a fix. Once the fix was released CERT would publish a report; however, there was no deadline for fixing a problem.

---

[3] http://www.businessweek.com/1999/99_08/b3617026.htm
[4] Humphrey, March 1999.
[5] Gleick, James. "A Bug and a Crash." URL: http://www.around.com/ariane.html (19 May 2002).
[6] http://www.cert.org/

According to Schneier[7] bug secrecy did not achieve the goal of fixing the vulnerabilities, as software vendors had little incentive to fix a problem that was not public knowledge and would not become public knowledge until the bug was fixed. Schneier[8] notes that there were incidents of vendors threatening researchers if they disclosed information about a bug and smear campaigns against researchers who announced the existence of vulnerabilities (even if they omitted details). As a result many vulnerabilities remained unfixed for years.

Full disclosure, on the other hand, is the release of information about a newly discovered vulnerability before curative measures are developed. Schneier[9] argues that full disclosure, which has gained popularity in recent years due to frustration with the secretive approach, has largely been successful. Today, many researchers publish vulnerabilities on mailing lists such as Bugtraq[10]. The Press publish them in computer magazines, which forces vendors to scramble to build the fixes so they can write their own press releases about how quickly they fixed them. But regardless of the method of disclosure widespread downtime continues to occur.

Microsoft has long maintained that public disclosure of newfound vulnerabilities without first notifying the vendor only gives the hacking community an opportunity to exploit a weakness. To help answer the problem of releasing vulnerability information prematurely Microsoft developed a variation of the secrecy model—they reward the discoverer of a bug with a citation that includes their name and their site's URL in the resultant knowledge base article that describes the problem and the fix. This citation only appears if the discoverer keeps the matter confidential with Microsoft.[11]

The security bulletin[12] that warned of the bug that allowed Nimda to breed, acknowledges the individual who discovered the vulnerability, Mr. Juan Carlos Cuartango. Microsoft's citation indicates that he confidentially reported the issue, helped them develop a fix, and kept the vulnerability secret. Even though the problem was not disclosed until the time of patch release, in the end the worm did a great deal of harm. Clearly it is not enough to have a fix available when publicly announcing a vulnerability. Companies must have a comprehensive, efficient solution to keep the fixes up to date.

The security professional tasked with keeping a company's systems current witnesses a steady stream of Microsoft's security bulletins, many of which affect commonly used server and desktop products like Internet Information Server and Internet Explorer. Worse yet, the majority of these bulletins claim the installation of the patch is critical to

---

[7] Schneier, Bruce. "Crypto-Gram Newsletter." 15 November 2001. URL: http://www.counterpane.com/crypto-gram-0111.html (16 May 2002).
[8] Schneier, 15 November 2001.
[9] Schneier, 15 November 2001.
[10] http://online.securityfocus.com/archive/1
[11] http://microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/policy.asp
[12] http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-020.asp

preventing an intruder from commandeering the entire environment. A recent study by UK-based managed security services provider, Activis[13], argues that it doesn't take a large network to produce a significant patch workload. They surveyed the number of released updates for a range of typical products (virus scanning software, routers, and firewalls were omitted for the purpose of this discussion) during 2000 and found that they would have needed to apply 345 updates in a network with only twelve computers, as shown in Table 1.

Table 1

Product	Number of updates in 12 months	Instances of Product	Total
Microsoft NT 4.0	26	12	312
Microsoft IIS 4.0	4	3	12
Microsoft SQL server 7.0	9	1	9
Microsoft Exchange 5.5	12	1	12
Total Patches to Install			345

It is easy to imagine a larger network's requiring thousands of instances of patch installations during the year to stay current. And considering that more vulnerabilities were reported to CERT during the first three months of 2002 than in all of 2000[14], the job isn't getting easier. There are, fortunately, ways to automate this daunting task.

Patch Types

Microsoft releases patches as service packs, roll-ups/cumulative patches, and hotfixes. A hotfix usually addresses a singular issue and is the first release of the fix. A roll-up or cumulative patch is a package of hotfixes that addresses multiple issues. A service pack is the accumulation of all hotfixes since the last service pack or original release of the software. It is best to apply service packs first, followed by roll-ups, followed by hotfixes. This order ensures that all historical fixes are applied before the newest ones are addressed.

The following two sections detail products and methods that can be used to deliver patches for a Microsoft Windows environment. The purpose of these sections is not to endorse any particular tool or set of tools, but rather to give the reader some options when designing a patch distribution system.

Free Tools and Methods

Variables such as budget constraints and the number of affected systems will impact which tools are chosen to patch the environment. A manager of a small network may find it sufficient to deploy fixes manually or use tools that do not have expensive

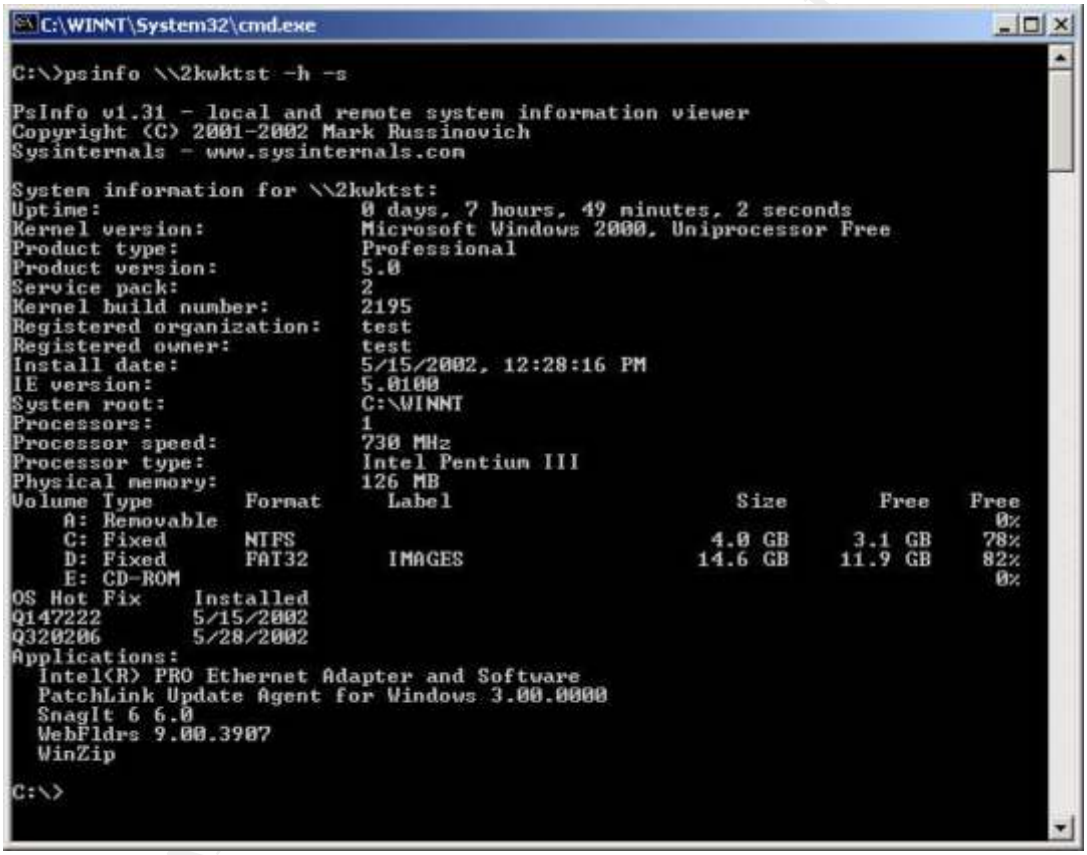
[13] http://www.activis.com/en/news/press_releases/2001/november/news29-11-01.html
[14] http://www.cert.org/stats/cert_stats.html

licensing requirements. While requiring more work to set up, scripted and shareware solutions may be the best choices for small budgets.

PSInfo

In order to roll out patches, it is helpful to determine which patches and service packs are installed on the system. SysInternals[15] has a number of excellent diagnostic applications, one of which, PSInfo[16], furnishes patch and other system information. It is part of their PSTools Suite[17]. The output of a remote workstation shown in Figure 1 demonstrates the system information that is retrieved.

Figure 1



The "-h" switch shows hotfixes by knowledge base article, and the "-s" switch displays installed software.

Reviewing system information for many computers using this tool can be time consuming, as the command line does not allow for multiple system input. The following

[15] www.sysinternals.com
[16] http://www.sysinternals.com/ntw2k/freeware/psinfo.shtml
[17] http://www.sysinternals.com/ntw2k/freeware/pstools.shtml

VBScript queries a list of machines using PSInfo and pipes the output to a text file. (It is provided "as-is" for instructional purposes only.)

```
'*********************************************************************************************
'*    Name:          PSInfo(c) Output for multiple servers                                     *
'*    Version:       1.0                                                                       *
'*    Author:        Tracy Lynn                                                                *
'*    Purpose:       This script uses PSInfo to retrieve information from multiple servers and *
'*                   exports the data to a text file                                           *
'*    Assumptions:   1)The file C:\PSInfo\SrvLst.txt has been created and contains the list of *
'*                   server names, 2)The contents of PSInfo.zip have been downloaded from      *
'*                   http://www.sysinternals.com/ntw2k/freeware/psinfo.shtml and expanded       *
'*                   into C:\PSinfo\, 3) Windows Scripting Host is installed on the machine that *
'*                   executes the script, 4) Remote Registry service is running on all         *
'*                   remote machines and account running this script has access to remote      *
'*                   HKLM\System on all queried machines, and 5) the system is NT 4 or greater.  *
'*    Legal:         Public Domain.  Modify and redistribute script at will.  No rights reserved.*
'*                   PSInfo is (c)2001-2002 Mark Russinovich (www.sysinternals.com)            *
'*********************************************************************************************

'Checks to make sure necessary files exist
Set oFileSystem = CreateObject("Scripting.FileSystemObject")
If Not oFileSystem.fileexists("C:\psinfo\srvlst.txt") Then
   msgbox "SrvLst.txt is missing!"
   msgbox "Please make sure C:\psinfo\srvlst.txt exists and run the script again."
   WScript.Quit
End If
If Not oFileSystem.fileexists("C:\psinfo\psinfo.exe") Then
   msgbox "Psinfo.exe is missing!"
   msgbox "Please make sure C:\psinfo\psinfo.exe exists and run the script again."
   WScript.Quit
End If
If Not oFileSystem.fileexists("C:\psinfo\pdh.dll") Then
   msgbox "Pdh.dll is missing!"
   msgbox "Please make sure C:\psinfo\pdh.dll exists and run the script again."
   WScript.Quit
End If

'Dialog asks for output filename
ask = "Please enter path and filename to create output text file.  Invalid path produces empty file."
title = "PSInfo(c) Output for Multiple Servers"
Do
file = InputBox(ask, title)
If file = vbEmpty Then
   WScript.Quit
End If
Loop While file = ""

'Reads lines of SrvLst.txt file, kicks off PSInfo, and pipes the output to the chosen file
Const ForReading = 1
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.OpenTextFile("C:\PSInfo\srvlst.txt", ForReading, True)

   do While Not f.AtEndOfStream
   servername = f.Readline

      Set wshshell = CreateObject("WScript.Shell")
      return = WshShell.Run("cmd.exe /c C:\PSInfo\psinfo.exe \\" & servername & " -h -s >>" & file ,0
,true)
      On Error Resume Next
   Loop

msgbox "Operation Complete!"
'*********************************************************************************************
******
```

Microsoft Network Security Hotfix Checker

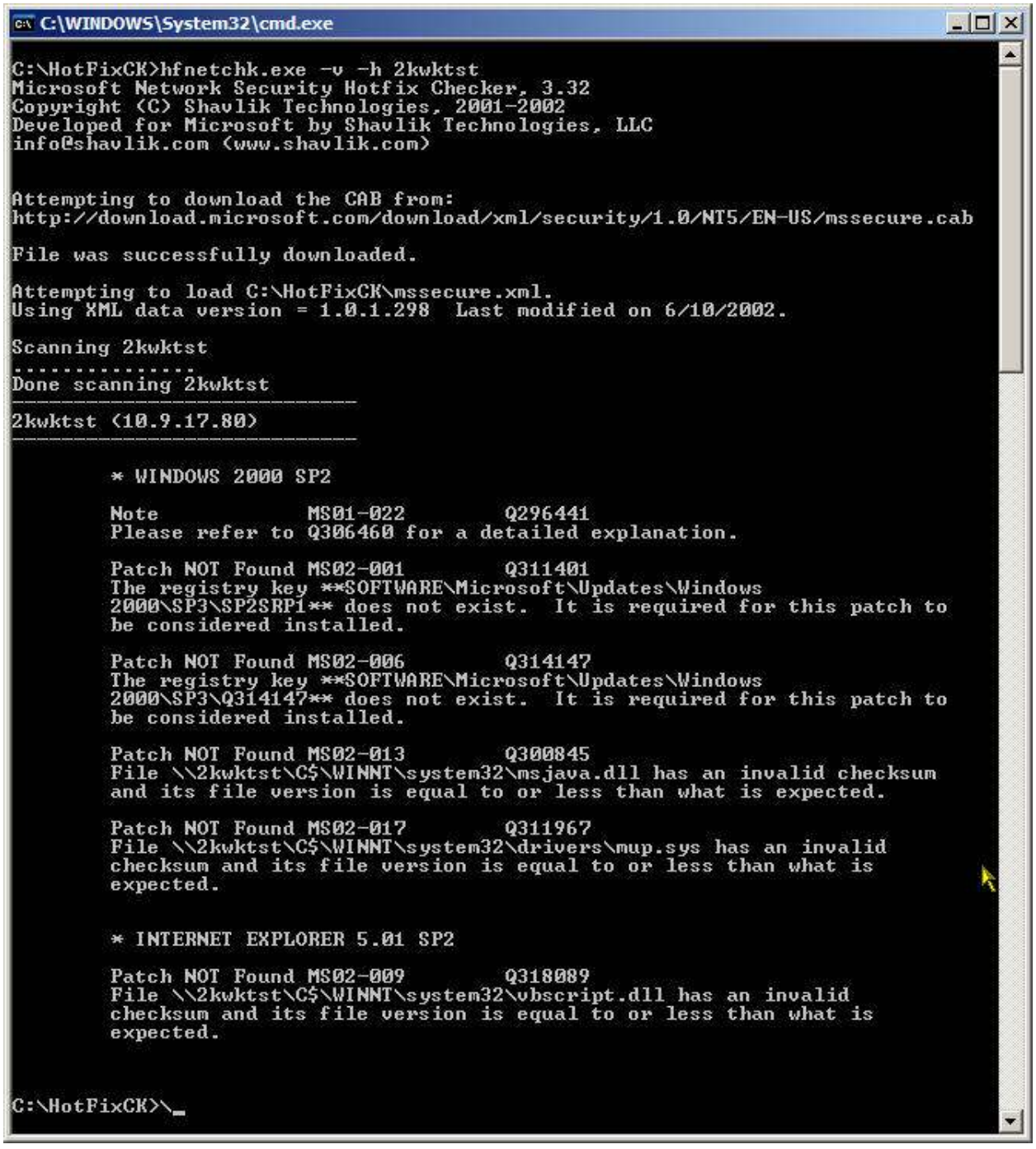
Another free tool that gathers patch information is the Hotfix Checker[18], developed by Shavlik Technologies[19] for Microsoft. This command-line tool can query multiple machines to determine patch status. It will only run on a Windows NT, 2000, or XP computer. The computer also must have Internet Explorer 5.0 or greater loaded, otherwise a standalone XML parser[20] must be installed. Hotfix Checker can query Windows NT 4.0, 2000, XP, Internet Information Server 4.0 and 5.0, SQL Server 7.0 and 2000, and Internet Explorer 5.0 and above. Currently it does not support Office applications or Exchange Server.

For the operating systems and applications it does support, the information it gathers is detailed. Hotfix Checker returns a list of patches that the computer *needs*, instead of simply lists what already is installed. It achieves this task by downloading a .CAB file to the machine that executes Hotfix Checker. The .CAB file is then expanded into an XML file containing a list of hotfixes that are available for each supported operating system and applications. The list also contains attributes related to the patches, such as added or edited Registry keys, version or checksum values of affected files, and patch-specific informational comments. The target machine is scanned to determine the operating system, service pack, and applications installed. Then the XML file is parsed, and the information within it is compared to the information gathered from the target machine. When a discrepancy is identified, Hotfix Checker assumes non-compliance and lists the patch that is missing, along with the specific discrepancy.

Figure 2 shows the output of Hotfix Checker after being run against a single workstation. Observe that the first patch entry indicates "Note" instead of "Patch Not Found." According Knowledge Base Article, Q206460[21], the XML file does not contain information to determine if fix MS01-022 has been applied. Subsequent entries indicate why patches were not located.

[18] http://www.microsoft.com/downloads/release.asp?releaseid=31154
[19] http://shavlik.com/security
[20] http://msdn.microsoft.com/downloads/default.asp?url=/downloads/sample.asp?url=/msdn-files/027/001/766/msdncompositedoc.xml
[21] http://support.microsoft.com/default.aspx?scid=kb;EN-US;q306460

Figure 2



It is possible to scan multiple computers by entering their hostnames or IP addresses individually. If a larger number of machines are to be scanned, HotFix checker can enumerate a list of computer NetBIOS names from a text file. You can also enter a range of IP addresses or even an entire domain to scan.

Scripts/Batch Files

Though not ideal for the purpose, batch files can be used to distribute patches in the enterprise. The following logon script delivers fix q321232 and also keeps the domain time synchronized. It assumes an NT or 2000 environment with no legacy clients and uses the freeware command line mailer, Command Mail[22]. Please note line nineteen wraps from previous line. (It is provided "as-is" for instructional purposes only.)

```
rem ********************************************************************************
rem     Name:           logon.bat
rem     Author:         Tracy Lynn
rem     Purpose:        Windows login script that installs hotfixes, synchronizes time
rem     Assumptions:    This file, Commail.exe, reg.exe, & hotfix exist in netlogon share, NT/2000
environment
rem     Legal:          Public Domain.  Modify and redistribute at will.  No rights reserved.

@echo off

rem ADD HOTFIXES BELOW

rem-------------------------hotfix q321232-----------------------------------------------------------------
SET sys= %WINDIR%\system32
IF exist %SYS%\hold1.txt goto MAIL
IF exist %SYS%\q321232.exe goto CONTINUE
Copy %0\..\q321232.exe %SYS%\q321232.exe
START /wait %SYS%\q321232.exe /q
Echo Placeholder for q321232 install > %SYS%\hold1.txt
Goto CONTINUE

:MAIL
IF EXIST C:\IPOUT.TXT DEL C:\IPOUT.TXT
ipconfig /all  >> c:\ipout.txt
echo UserName = %username% >> c:\ipout.txt

%0\..\reg.exe QUERY HKLM\System\CurrentControlSet\Control\ComputerName\ComputerName >> c:\ipout.txt
%0\..\reg.exe "QUERY HKLM\Software\Microsoft\Windows\CurrentVersion\Internet Settings" >> c:\ipout.txt

IF NOT EXIST %SYS%\commail.exe copy %0\..\commail.exe %SYS%
%SYS%\commail.exe -host=mail.company.com -from=%username% -to=installs@company.com  -
subject="q321232" -msg=c:\ipout.txt
IF EXIST %sys%\hold1.txt DEL %SYS%\hold1.txt
Rem----------------------end hotfix q321232-------------------------------------------------------------

:CONTINUE
net time \\YourTimeServer /set /yes
rem ***********************************end script*****************************************
```

[22] http://www.xwebware.com/products/commail/

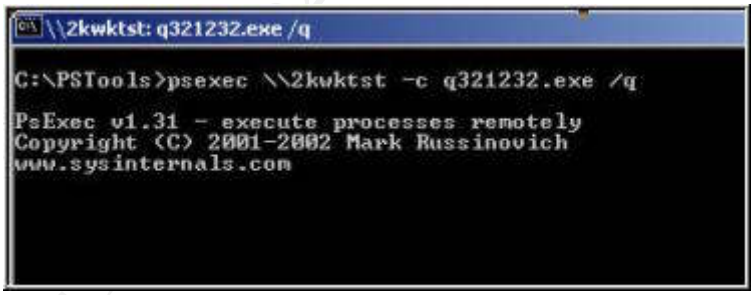
After the fix has been executed on the local machine an informational e-mail is sent to a public folder and contains user and system information as well as a registry key that contains the name of the hotfix if the install was successful. It will be necessary to choose a different registry key or other proof of success depending on the fix chosen.

Running a patch from the logon script, however, has special considerations. First, someone must execute the script at the machine by logging on with an account that has been configured to run that particular logon script. This may be desirable for client workstations, but it becomes a problem for servers, as an administrator must logon to every server requiring the fix. Second, the installation may be visible to the user community if intervention such as acknowledging a reboot dialog box is required. Third, conflicts can occur between the patch and other applications that start up during the logon process. Fourth, new logic must be written into the script and tested whenever a new patch is installed.

PSExec

PSExec[23] is another tool in the PSTools suite. This product allows the administrator to execute an application on a remote machine. The advantage of using this tool is that it is not necessary for a user to manually execute the patch application on the target computer. Figure 3 shows the cumulative I.E. patch Q321232 being deployed to a remote workstation.

Figure 3



The "/q" argument runs the installation in quite mode, thus eliminating the need for user intervention; however, the system requires that the local user reboot the machine if the patch requires a reboot. Patches that do not require a reboot run without intervention using the "/q" argument. Logic in the above script can be modified to run PSExec for patch deployment on multiple machines.

[23] http://www.sysinternals.com/ntw2k/freeware/psexec.shtml

Windows Update
Microsoft's Windows Update[24] is a web-based interface that queries the local machine for installed hotfixes, rollups, and service packs. Though it works well for individual computers it has some limitations that make it impractical as an enterprise patching solution. First, it has no mechanism for deploying patches to remote machines, as a user must interactively execute the update process on every machine. Second, it only updates operating systems Windows 98, ME, NT 4.0, and 2000, as well as IIS 4.0, IIS 5.0, and Internet Explorer; it does not provide patches for server products like Exchange Server or SQL Server.

Windows XP Professional and .NET Server have a built-in automatic update feature that can download required updates and prompt the user to install them. Though this process is now automated, it still only provides the same spectrum of updates as those found on the Microsoft Windows Update site. And considering the time it will take most businesses to migrate to .NET, other methods of patch delivery must be implemented in the short term.

Commercially Available Tools
The ease of deploying patches without having to write scripts or interact with browser sessions comes at a price. The following options can distribute patches throughout the network, on both servers and workstations, unattended and by a schedule, if desired. These products charge per managed node or per authenticated user and require varying levels of effort to deploy.

Update Expert
St. Bernard Software makes a patch distribution product called Update Expert[25]. This application can be installed on any Windows NT or 2000 machine and requires no client agent on managed computers, a plus if one is concerned about overhead. It deploys updates to Windows NT 4.0, 2000, XP, IIS, SQL Server, Exchange Server, Internet Explorer, Media Player and Media Services, Net Meeting, Office 2000 and XP, and Outlook.

Once the machines to manage are chosen they are inventoried to determine which patches and service packs are installed. The product maintains a database of fix information that can be set to update from St. Bernard's site on a regular basis; however, the files themselves are not downloaded to the machine running the application until they are requested. One can either download the patches for archiving or download them at the time of deployment.

[24] http://windowsupdate.microsoft.com
[25] http://www.stbernard.com/products/updateexpert/products_updateexpert.asp

Update Expert's scheduling feature gives control over distribution by allowing the administrator to schedule any number of updates to any number of servers and workstations at any time.

Figure 4 shows "research" view. Managed machines appear in a tree view on the left. On the right all available fixes are listed in categorized folders. Installed fixes are identified by a green dot. The fix database contains links to Microsoft knowledge base articles that can be viewed in the pane below by highlighting a fix.

Figure 4

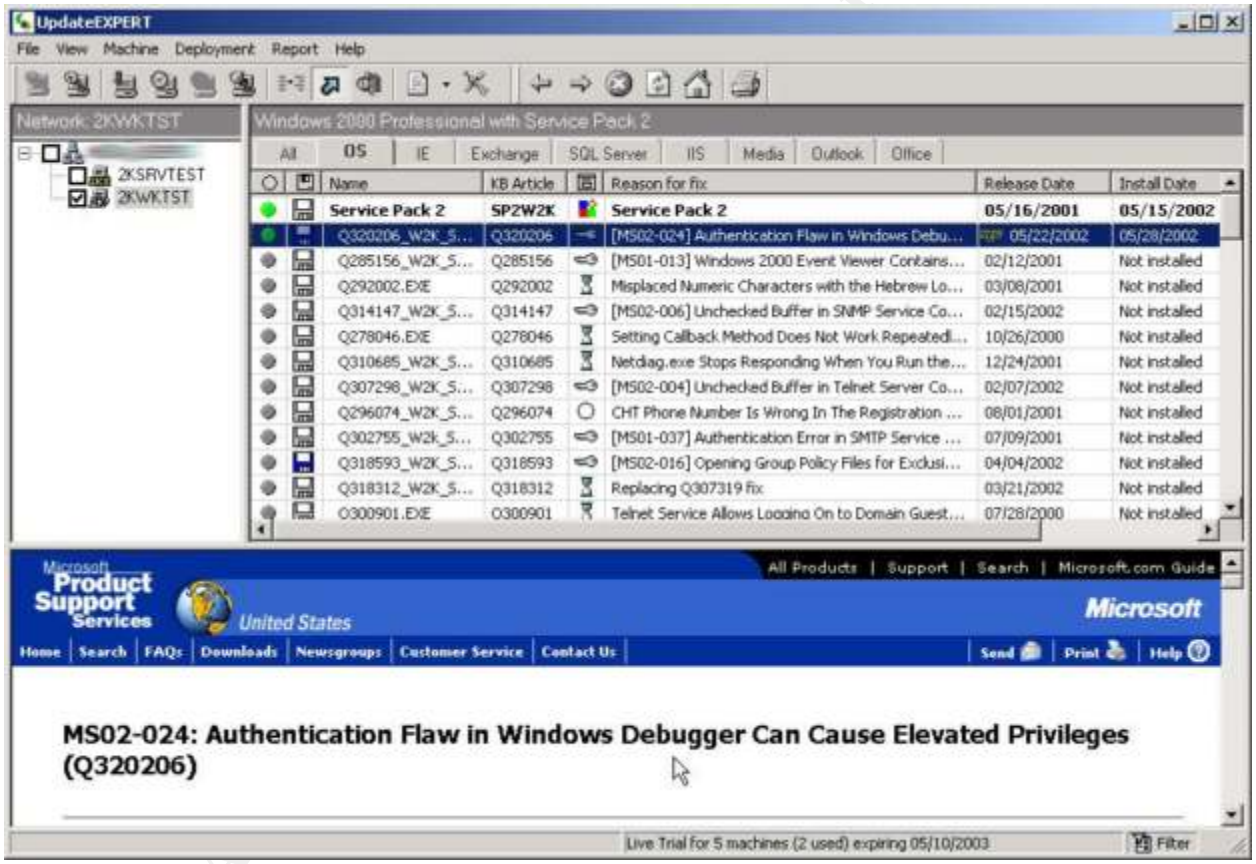
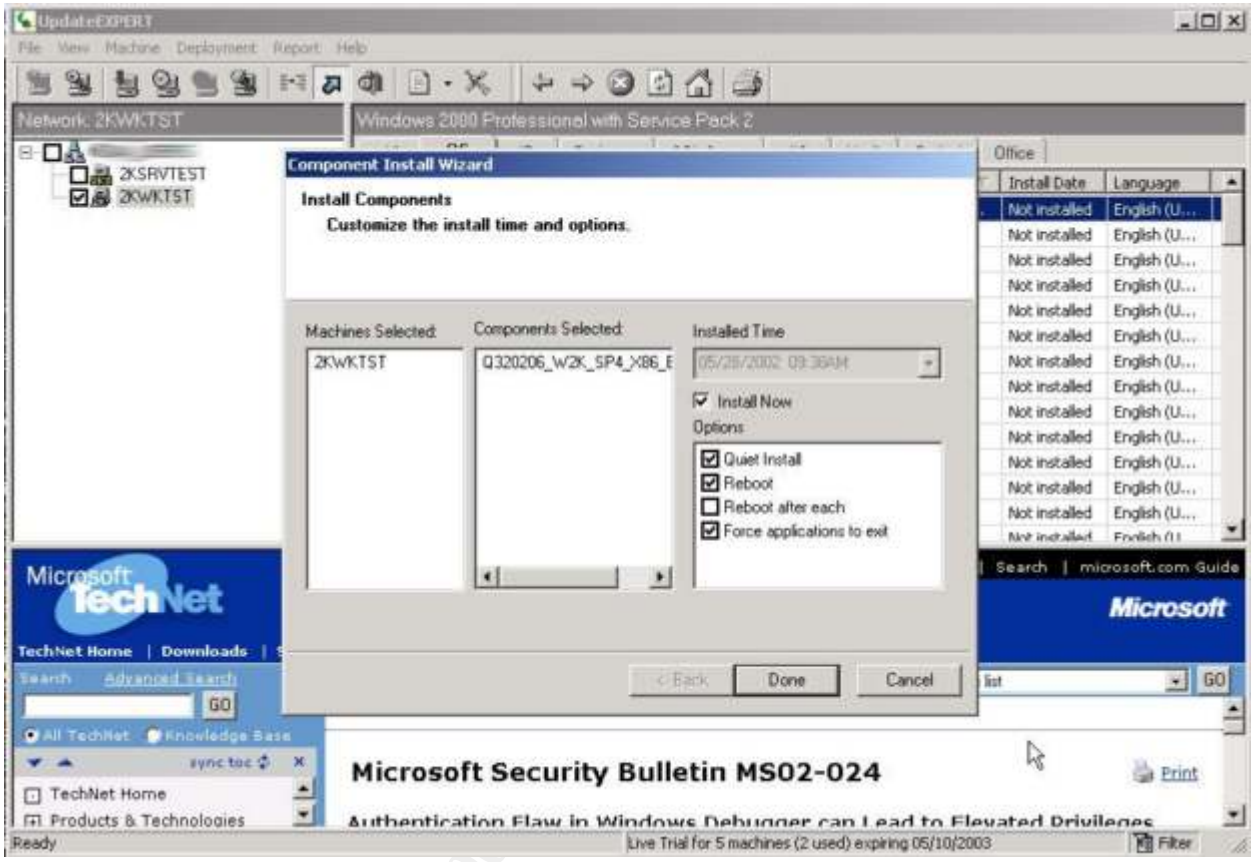


Figure 5 shows preparation for a deployment. Right-clicking the desired patch and choosing "install" displays the Component Install Wizard. From here the patch is either installed immediately or set up on a schedule. A custom installation option also is provided, allowing the administrator to deploy patches not supported by St. Bernard. Any application can be installed on managed machines in this way, as long as it is contained within one executable.

Figure 5



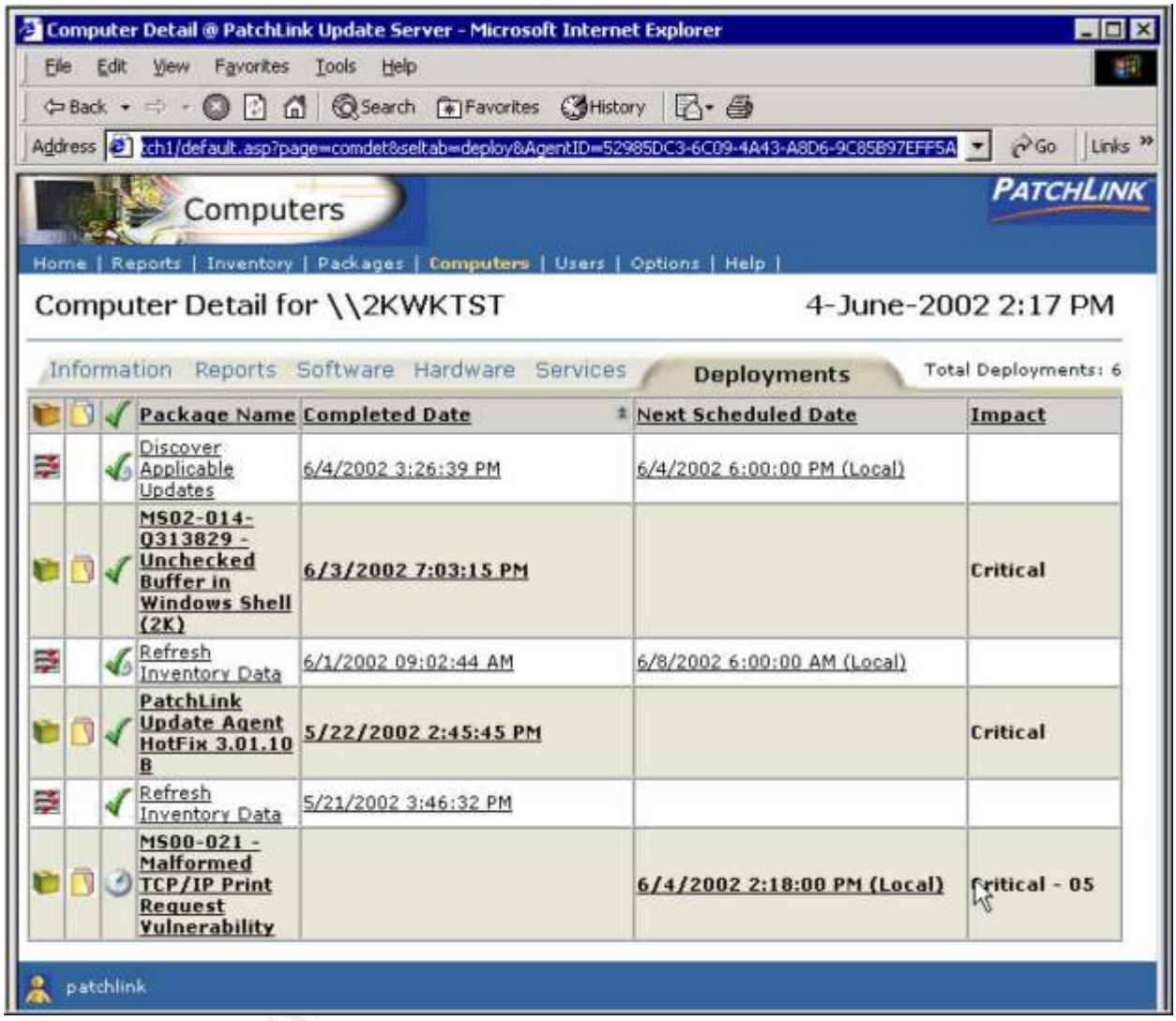
Patchlink Update Server

Patchlink Update Server[26] differs from Update Expert in a few significant ways. First, it must be set up on an NT/2000 server that has no other roles. This hardware requirement can significantly increase the cost of implementation. Second, it requires the installation of a client on every managed machine, which consumes local system resources. Third, it is operated from a web interface, which makes product access more convenient than Update Expert. Besides these differences, both Update Expert and Patchlink Update Server do essentially the same things, such as organizing patches applicable to a particular machine, deploying patches to one or many machines immediately or on a schedule, and reporting on installed patches as well as on those machines that are not patched.

Figure 6 shows deployment detail for one workstation. The "Deployments" tab shows successfully deployed packages as well as system updates and queries.

[26] http://www.patchlink.com/

Figure 6



Systems Management Server Software Distribution

As a software distribution product Systems Management Server[27] (SMS), is effective and versatile. It also contains a number of other management tools such as hardware and software inventory, determining product licensing compliance, and managing remote users through terminal sessions. Using SMS as a patch distribution product, however, has a couple of disadvantages over both Patchlink and Update Expert. First, training is necessary in order to understand how to design, deploy, and manage SMS in the environment. Second, the administrator must manually research and download the patches needed and cross-reference the application or operating system version with the

[27] http://www.microsoft.com/smserver/

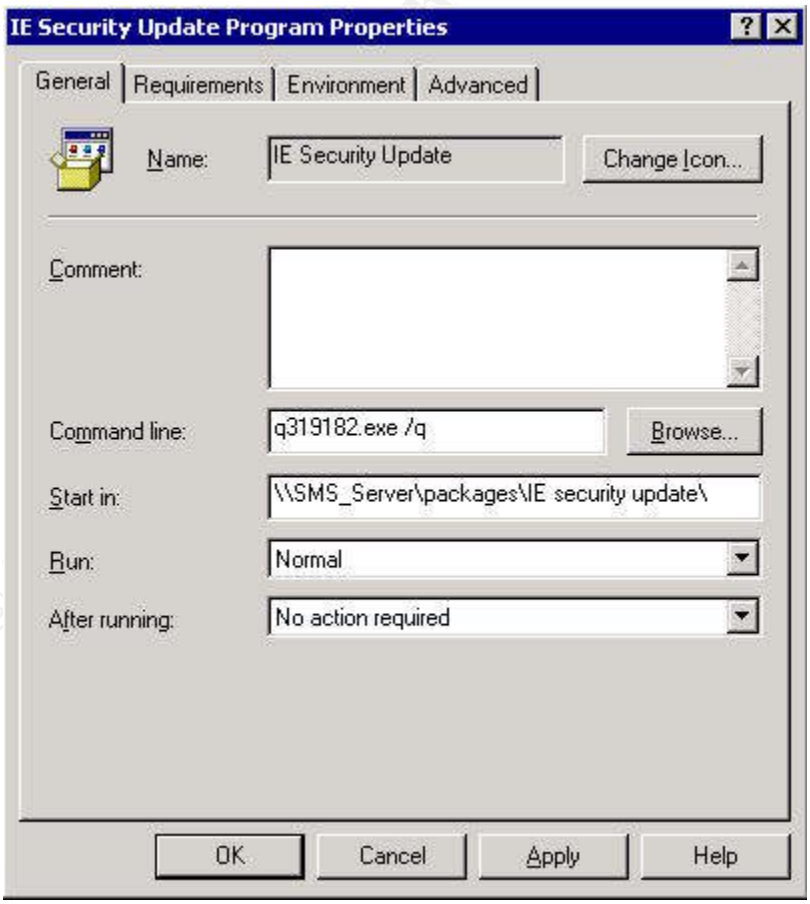
appropriate patch before distributing. Like Patchlink, SMS requires its own server, and client agents must be installed on all managed machines.

To distribute a patch with SMS a package is created. The package contains the source file(s) needed for the installation, one of which must be a program that executes the package.

Then an advertisement is built to deliver the package. The advertisement contains a number of parameters that control the delivery of the package. First, the advertisement sends out the package based on a collection, which is a group of machines organized by an attribute, such as username, hostname, subnet, type of operating system, etc. The advertisement is based on a schedule, which means that one can deploy to members of the collection immediately or choose a time in the future. Once the advertisement has been scheduled, it is considered to be "assigned." The user can be given a window of time to manually execute the package or it can be executed automatically.

Figures 7, 8, and 9 illustrate the setup of an Internet Explorer update to be distributed to a collection of all NT 4.0 workstations. In figure 7 the program is put into the package.

Figure 7



The advertisement is then constructed, shown by figure 8. Select the package and program within (multiple programs can be set up in a package) and choose the collection of machines to which the update will be distributed.

Figure 8

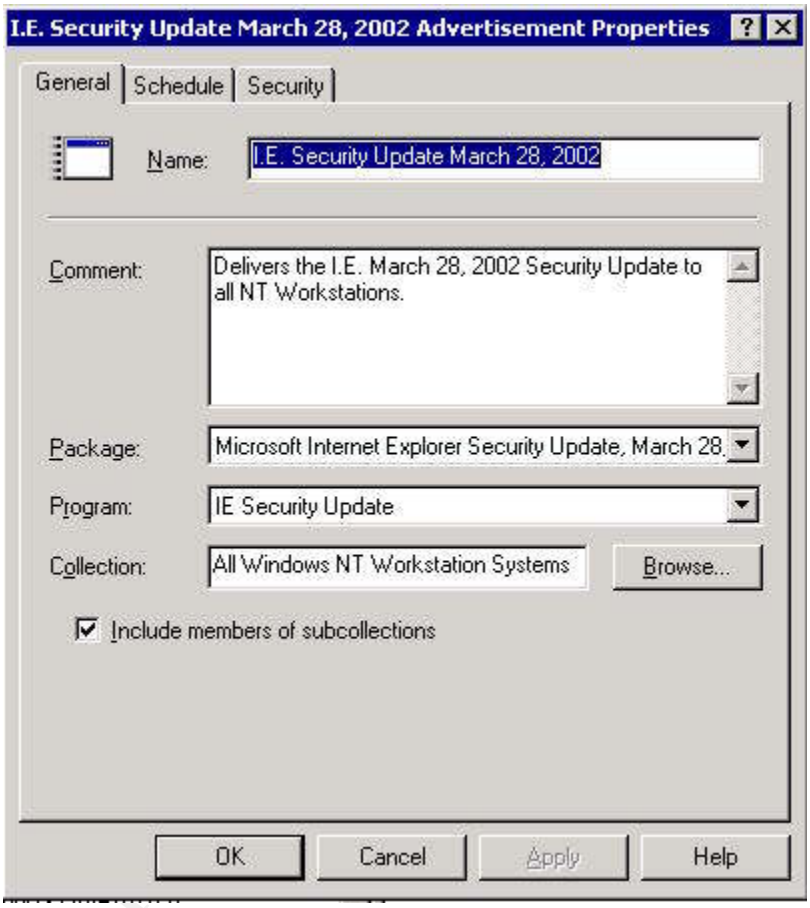
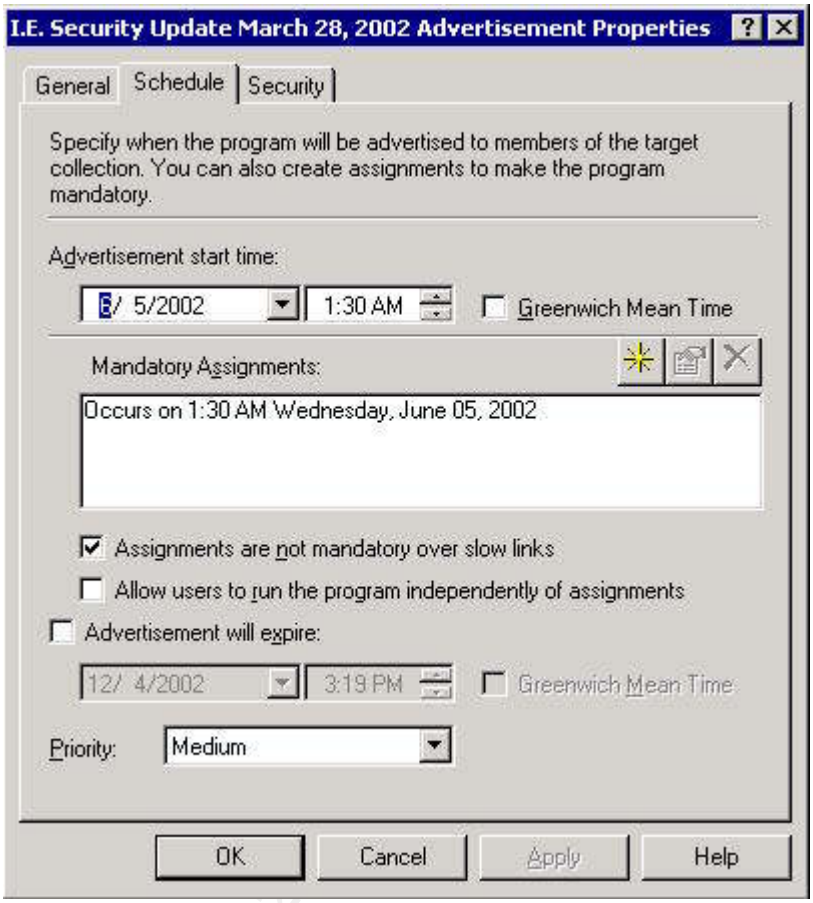


Figure 9 shows the schedule configuration. In this example the package is mandatory and was deployed on 6/5/02 at 01:30. It installed automatically on all machines in the collection that were running at the time, with or without a currently logged in user. Note that if a mandatory assignment was not indicated the package would be available for the user to install at his or her discretion up until the advertisement expiration date.

Figure 9



SMS also features a product called SMS Installer which can be used as a wrapper for patches and other software deployments. In order to install it you must have a licensed copy of SMS, otherwise the files will not extract. SMS Installer creates a compiled self-extracting executable that contains the patch, or the patch can be executed externally from the installer application. While designed primarily as an application deployment tool, it can be advantageous for patch distribution. For example, the administrator may wish to include custom dialog boxes or graphics to communicate actions to users or may need to run a different version of a patch depending on what version of an application the computer is using. These tasks can be built into the executable.

The Future of Patching

Enterprise patching solutions quickly are emerging in response to the growing number of fixes to deploy. Among other vendors and developers, Microsoft has increased its efforts to develop products to help administrators stay on top of patching. The .NET and XP operating systems come with an automatic update feature that can download and install patches without intervention. Though still not as robust as some third-party products, Automatic Update will handle much of the patching once the environment is pure

.NET/XP. Future operating systems will continue to improve on the patch delivery process as need for eliminating vulnerabilities increases. Unless third-party tools offer advanced functionalities that the native tools do not have, they likely will become obsolete. But regardless of the tool used, maintaining a patched environment will continue to be an important security priority for the administrator.

References

CERT/CC Statistics 1988-2002. URL: http://www.cert.org/stats/cert_stats.html (13 May 2002).

Company Reports. "The Long and Winding Windows NT Road." BusinessWeek Online. 22 February 1999. URL: http://www.businessweek.com/1999/99_08/b3617026.htm (15 May 1999).

Gleick, James. "A Bug and a Crash." URL: http://www.around.com/ariane.html (19 May 2002).

Humphrey, Watts S. "Bugs or Defects?" SEI Interactive. March, 1999. URL: http://interactive.sei.cmu.edu/news@sei/columns/watts_new/1999/March/watts-mar99.htm (18 May 2002).

Microsoft Knowledge Base Article. "Hfnetchk.exe Returns NOTE Messages for Installed Patches (Q306460)." 24 October 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q306460 (14 May 2002).

Press Release. "Activis warns that too many demands on IT Managers is jeopardizing security." 29 November 2001. URL: http://www.activis.com/en/news/press_releases/2001/november/news29-11-01.html (13 May 2002).

Schneier, Bruce. "Crypto-Gram Newsletter." 15 November 2001. URL: http://www.counterpane.com/crypto-gram-0111.html (16 May 2002).

Schneier, Bruce. Secrets and Lies—Digital Security in a Networked World. New York: John Wiley and Sons, Inc., 2000. 210.