

ITPro<sup>TM</sup>  
SERIES

WindowsITPro

 **eBooks**

Keeping Your Business  
SAFE from Attack:

# Patch Management

By Jeff Felling

**Microsoft<sup>®</sup>**



## Contents

<b>Chapter 3 The Dry Run: Setting Up a Lab to Test Patches and Updates and Using Microsoft Baseline Security Analyzer to Scan for Missing Patches</b> .....	<b>38</b>
<b>The Test Lab</b> .....	<b>39</b>
Creating Your Lab: Using Virtual Machines vs. Dedicated Hardware .....	39
Configuring Forests, Domains, and DCs .....	40
Patch Deployment Software .....	40
Network Considerations .....	41
Living Dangerously: Using Production as Your Test Lab .....	41
<b>The Test Plan</b> .....	<b>42</b>
<b>Verifying Installation and Scanning for Missing Patches with MBSA</b> .....	<b>43</b>
MBSA Compatibility .....	43
MBSA Installation and Configuration .....	44
Start Scanning .....	45
MBSA Command Line .....	47
Viewing Reports .....	47
MBSA as HFNetChk Replacement .....	49
MBSA Limitations .....	51
<b>The Timeline from Test to Production</b> .....	<b>51</b>

## Chapter 3:

# The Dry Run: Setting Up a Lab to Test Patches and Updates and Using Microsoft Baseline Security Analyzer to Scan for Missing Patches

As with just about everything, practice makes perfect and patching systems makes no exception. Ok, so installing a patch doesn't require too much practice. But your patch deployment tactics as a whole will directly benefit from careful planning and execution, which can be facilitated by dry runs and patching drills. If you manage an organization with many different systems performing a variety of roles, you'll find practice and planning of your patch deployment process invaluable to minimizing downtime and unpleasant surprises when rolling out patches to all your production servers and workstations.

The extent and ability to test patches depends on your organization's size and varies by system. For example, heavy-load Web servers or large database servers might be hard to replicate in a lab environment, but perhaps smaller versions of the same equipment can be tested. Also, plan to spend more time testing critical servers or servers running custom applications. Third-party applications will not have been tested as a part of Microsoft's prerelease patch quality assurance (QA) process, so you will want to ensure that a new patch does not adversely affect your non-Microsoft applications. For example, you should test an external Web site running custom code differently than a more or less vanilla file and print server running only the Windows OS.

This chapter examines different ideas and approaches to putting together a test environment to help with predeployment testing to minimize the risk of a patch adversely affecting your production systems. Consider the ideas in this chapter as guidelines that you can mold to fit your particular organization and server topology. In an ideal world, we would have a comprehensive lab that matches our production systems exactly and no time constraints so that we could perform full software and patch regression testing. Not surprisingly, you will have to make day-to-day decisions including not only when and where to deploy a patch, but also what level of testing you will undergo. You must continuously weigh the risk of an unpatched server against a shortened test schedule.

Lastly, this chapter examines Microsoft Baseline Security Analyzer (MBSA) a freely downloadable tool from Microsoft that you can use in your lab and production environment to find missing patches and confirm patch deployment success. If you are using a deployment tool that does not support user-initiated scans of targeted systems, then you will find MBSA an especially important tool in your security update toolkit.

## The Test Lab

Creating a test lab that emulates your production systems provides an important foundation for constructing your patch management program. This test lab will let you deploy patches, test new patches for compatibility with your existing applications, and let you conduct load testing, penetration testing, or other specialized testing not feasible to perform on production systems.

Ideally your test lab will consist of a subset of servers with each representing a type of server in your production environment. Include at least one of every type of server that you have in production. You may have multiple Exchange servers, a cluster of Web servers, or multiple SQL Servers configured to replicate between each other. Best case, you need to configure all these types of components and interactions in your lab. If you have a Web farm of many servers, you might need to use only one or two computers (depending on how they interact with each other) to represent this configuration in a lab. If you use clustering for load balancing or high availability or use Application Center to configure your production servers, then you will likely want to configure a similar system in your lab comprised of fewer servers but that uses these same services. Even though creating a lab to this level of detail can be difficult or time consuming, you will find reward the first time a patch breaks something in your lab and you discover it there instead of production.

Your lab computers need to run on hardware platforms similar to production to ensure that the BIOS, hardware drivers, monitoring software, array software, and other drivers closely match those used in production. You might not want to match the hardware exactly due to cost, size, or other reasons. Most server manufactures produce a line of servers that use many of the same components throughout the family. Check whether you can represent your more powerful servers by less expensive counterparts in the lab. Perhaps your quad-processor server with 4GB memory and a massive disk array can be represented in the lab by a much smaller dual-processor server of the same line.

Also, consider the similitude of your lab to production. Configure your domain controllers (DCs) and domains in your lab like your DCs and domains in production, mimicking any Group Policy Object (GPO) settings including security templates.

### ***Creating Your Lab: Using Virtual Machines vs. Dedicated Hardware***

Advances in virtual PC/server software such as Microsoft Virtual PC and VMware Workstation or VMware Server let you host multiple and independent PC instances on one server. In essence you can install a DC, Web server, and other member servers all as separate virtual computers hosted from one hardware platform. The OS within virtual instance thinks it's running on its hardware.

These virtual PC technologies can help lower costs and make it quicker to restore the states of computers after a lab exercise. (For example, you can generally make a backup of the files that *define* the virtual computer, then simply copy them back to restore the original state of the computer. If you use Microsoft Virtual PC, then you can use an undo disk (directly from the Virtual PC application) to return to a known state.

Use virtual machines to test nonhardware-related patches or to build up a lab infrastructure that you will not use for direct patch testing. However, do not rely on virtual machines for end-to-end patch testing because the hardware in the virtual computer will be different than your production server hardware.

For example, a virtual PC can be beneficial to patch testing when you want to deploy patches to several Web applications each running on separate Microsoft Internet Information Server (IIS) servers. You can install these Web applications on multiple virtual servers, then test how a patch might affect your Web application code. But this testing examines the patch against only your code—not the OS code and its interaction with the hardware platform. For this complete end-to-end testing you need to test on a platform base consisting of the same server platform family (or when feasible, testing on the same server model), OS version, and similar ancillary software installed. Don't forget to also install your monitoring software or other agent-based software possibly installed on your production servers. You need to consider each of these software components, although possibly very small, when creating your lab because they can affect the success or failure of your patch installation.

As a guide during your security update triage exercises, remember to consider what components a specific patch affects. The Microsoft security bulletin lists each of the files that a patch replaces and updates so that you can use this information to help construct or confirm that your test lab is adequate in appropriately relevant areas. For example, if you plan to apply a security update that corrects a problem in the SNMP subsystem, you will want to be sure that you have all of your monitoring software (such as HP OpenView or other SNMP-based monitoring software) installed and running on your test systems before applying the patch. After installing the update, you can confirm that the systems continue to operate as expected.

### ***Configuring Forests, Domains, and DCs***

Install a test lab with a forest structure that matches production. Configure user accounts similar to your production domain. If you rebuild your lab (which is a good idea to do from time to time), consider writing a script to programmatically create your user accounts and groups and assign security group memberships. The more that you can automate, the quicker it will be to restore a lab—especially if it is used for purposes other than security update testing. Also don't forget to update or extend your lab schema to match production, such as running Adprep. Again, the closer your lab matches production, the higher the chances of discovering patch deployment problems before they adversely affect production.

You can use a virtual computer environment to build out your forest in which you functionally represent multiple domains by installing multiple DCs on one virtual computer hosting server. However, remember to keep in mind the caveats described earlier and avoid patching these virtual computers as a legitimate test. You can use the virtual computer servers to help you build out your domain structure, but do not use them as members of the test bed. (Of course, you will want to patch these computers anyway to harden them from possible exploits just as if they were in your production environment. Don't forget to patch your virtual computer host servers as well!)

### ***Patch Deployment Software***

Use the same patch deployment software in your lab that you will use in production. If you use Microsoft Software Update Services (SUS) to deploy your patches in your production environment, include an SUS server in your lab and use it to deploy the updates in your lab. Following your production process as closely as possible will help identify any problems associated with the mechanics of rolling out the patches and give you a *feel* for the particular deployment. Consider watching for these characteristics in the lab:

- Observe the time taken for patch deployment from start and end.
- Make notes of any dialog boxes that might prompt during the rollout process and how to handle them.
- Ensure that your patching software successfully downloads the proper updates, stages them for deployment, then copies and installs the patches on the target systems.
- Conduct a postinstallation scan with your patch management software or another tool like MBSA to confirm that each test patch was deployed correctly. (The last part of this chapter describes MBSA and how to use it to scan for missing updates.)

Another benefit of using the same production and lab patch deployment software is to ensure proper configuration for deploying updates to the wide variety of software that might need updates. For example, some Microsoft Office patches require access to the Office application files distribution point from which to copy source files. Using your patch deployment software in the lab, you can uncover these types of requirements early in testing. When you are ready to roll out to production, you will have already vetted your deployment process from start to finish. Scanning new software with antivirus software to ensure that viruses are not introduced into your network is also a good idea.

### ***Network Considerations***

In some cases with more sophisticated labs, you might want to simulate deployments across network connections like those used in the production network. For example, you can use a network latency simulator to emulate the characteristics of a slow WAN connection between your main office and a branch office. Testing your patch deployment software across a slow link such as this will help you identify network-related concerns early on. An example of a network latency simulator is the FreeBSD DUMMYNET program. This emulator lets you simulate a variety of network scenarios, which might be useful when you are testing the deployment of a service pack to several computers in a remote office and you wonder what the effect might be on the WAN link or how long a remote deployment might take when using your deployment software.

Deploy firewalls in your lab the same as those in production, including a similar access control list (ACL). For example, if your production servers cannot access the Internet, then your lab computers need to have a similar restriction. These restrictions can help identify potential problems with your patch deployment process before they surprise you in production.

### ***Living Dangerously: Using Production as Your Test Lab***

Although not advisable, sometimes you must *test in production*. In some circumstances you might have a test lab (and have fully tested the patches to your satisfaction) but still prefer to roll out to production in a staged manner. These types of live deployments best work in environments that have many servers performing one role, such as a Web farm consisting of many load-balanced Web servers. For example, let's say you have 20 Web servers each sharing the incoming Web traffic load. Perhaps to build in redundancy, you added extra Web servers so that several could fail without affecting overall service availability. In essence you have a tolerance to risks associated with deploying an untested patch in production. (Of course, you will want to consider what happens if a rogue patch on even one server affects data integrity to a backend database or other downstream

server.) Even in this scenario end-to-end testing is preferable to deploying an untested patch to a production environment.

However, deploying a patch to a live production server will give you real-world data not available in any lab. This data will tell you exactly how the patch will perform in your daily environment. Again, however, this type of deployment is *final* and much more risky when performed in lieu of formal testing. For better results, test the patch fully in a lab, then deploy it to one or two servers in production to burn in the patch before deploying it to all your servers. However, there is no panacea. This extensive testing and burn in period is disadvantageous as it takes time, thereby increasing the risk to your unpatched servers, which are possibly vulnerable to exploit. Remember that patching a computer system is changing the software code running on it, so be sure that you have a reliable (and tested) backup and restore process or failover/high availability options for servers less tolerant to downtime and outages.

### The Test Plan

A well-defined test plan that exercises the functionality of your systems after deploying a patch is just as important as a properly configured test lab. Your test plan needs to flex and test key aspects of your servers and the applications that run on them. A plan can be very basic, such as deploying an update to a corporate workstation, then testing the functionality of the update by logging onto the network, running Office programs, and accessing the company intranet.

Other plans might be specific to a server function. For example, if your company hosts a Web application, you might be able to borrow regression tests from your QA department that test all components of the Web application. Depending on the sophistication of your application your company might even have developed automated testing scripts or programs that you can leverage for your patch testing.

Automated testing probes many different aspects of an application's functionality in a reliable and repeatable manner. To take this one step further, automated unit testing systematically tests the low-level functionality of many different systems—often times at the object level. Although unit testing might not be possible for all organizations, you might be surprised at what you can find already exists in your organization—especially if you already employ a QA department whose job it is to test server-based applications. Also, unit testing is not necessarily limited to a single server. The testing modules might be able to test everything from the Web application to backend data-processing components such as database servers or n-tier servers.

A test plan for conducting regression testing on patches is an important component in reducing risk when deploying new patches in your environment. Microsoft releases patches monthly, so you will be more efficient if you devise a repeatable plan that you can use for every patch deployment. The test plan should not only exercise the functions of the application before and after deploying the patch, but also search for signs of possible deployment problems, such as errors in the event log.

## Verifying Installation and Scanning for Missing Patches with MBSA

Scanning your test and production systems is an important component to confirming that the patch has been installed successfully. Most patch management software products provide this type of scanning because it is an important first step towards the deployment of the patches. An exception to this is SUS, which uses the client-based Automatic Updates to determine whether or not a patch is deployed. SUS also lacks a target-based scanner. Therefore, using this tool to determine your level of patch compliance for target systems in your organization is somewhat difficult. For example, if you have configured SUS to download but prompt users to install the patch, you can't easily determine how many users have installed the security update.

To complement Microsoft's patch-deployment systems—or to implement a simple update-detection system—you can use the small, yet agile, and feature-packed MBSA to regularly scan your network. MBSA not only scans local and remote systems for patch-update status but also performs more than 65 vulnerability-scanning tests specific to Microsoft products. And although MBSA doesn't patch your systems or plug your holes, the product's fast and lightweight approach provides a quick and efficient method for canvassing your systems for common vulnerabilities.

No stranger to update scanning, MBSA provides the scanning engine that the enterprise-focused Microsoft Systems Management Server (SMS) uses. MBSA also supports vintage HFNetChk functionality. HFNetChk, MBSA's predecessor, enables local and remote scanning of Microsoft OS security updates, as well as updates for Microsoft's enterprise applications such as Microsoft Exchange Server and Microsoft SQL Server. To extend HFNetChk's functionality, MBSA features product security scans that search for known OS misconfigurations that can result in system vulnerabilities.

MBSA includes both a graphical front-end version for ad hoc scanning and a script-friendly command-line interface version. MBSA saves its scan results in an easy-to-read XML format, further increasing the product's usefulness by writing custom reports that fit your needs.

### ***MBSA Compatibility***

Microsoft released MBSA 1.2.1 in August 2004. Although you must run this version of MBSA from a Windows Server 2003, Windows XP, or Windows 2000 (Win2K) computer, you can remotely scan Windows 2003, XP, Win2K, and Windows NT 4.0 systems. Using one scanner to scan multiple Microsoft products presents a challenge because of update-format compatibility problems. Microsoft uses multiple update engines and processes for many of its products, and until the company concentrates on one method, many of its update tools will work with only specific products. (For example, the Microsoft Office update tools work differently from the Windows Update tools, resulting in incompatible update-distribution methods.) Despite these challenges, MBSA supports security and update scanning for the Microsoft products that Table 3-1 lists.

**Table 3-1 MBSA-Supported Microsoft Products**

OSs	Server Software	Desktop Applications	Tools
Windows 2003	BizTalk Server	Internet Explorer (IE)	Microsoft Data Access Components (MDAC)
Windows XP	Commerce Server	Office	MSXML
Windows 2000	Content Management Server (CMS)	Windows Media Player (WMP)	Microsoft Virtual Machine (VM)
Windows NT 4.0	Exchange Server		
	Host Integration Server (HIS)		
	IIS		
	SQL Server		

## ***MBSA Installation and Configuration***

MBSA installation is a snap. Download the MBSASetup-EN.msi Windows Installer (.msi) file from the MBSA Web site at <http://www.microsoft.com/technet/security/tools/mbsahome.msp>. This site contains detailed information about MBSA, including descriptions of the MBSA scans and an FAQ that addresses how MBSA interoperates with other patch-deployment systems (e.g., SUS).

By default, the setup program installs MBSA in the C:\Program Files\Microsoft Baseline Security Analyzer directory. This folder contains the MBSA executables `mbsa.exe` and `mbsacli.exe`, which provide the GUI and command-line interface to the scanning application. The installation directory also contains the HTTP and Extensible Style Language Transformations (XSLT) templates that MBSA uses to format and display the builtin reports. A Help directory provides comprehensive descriptions of each test that MBSA performs.

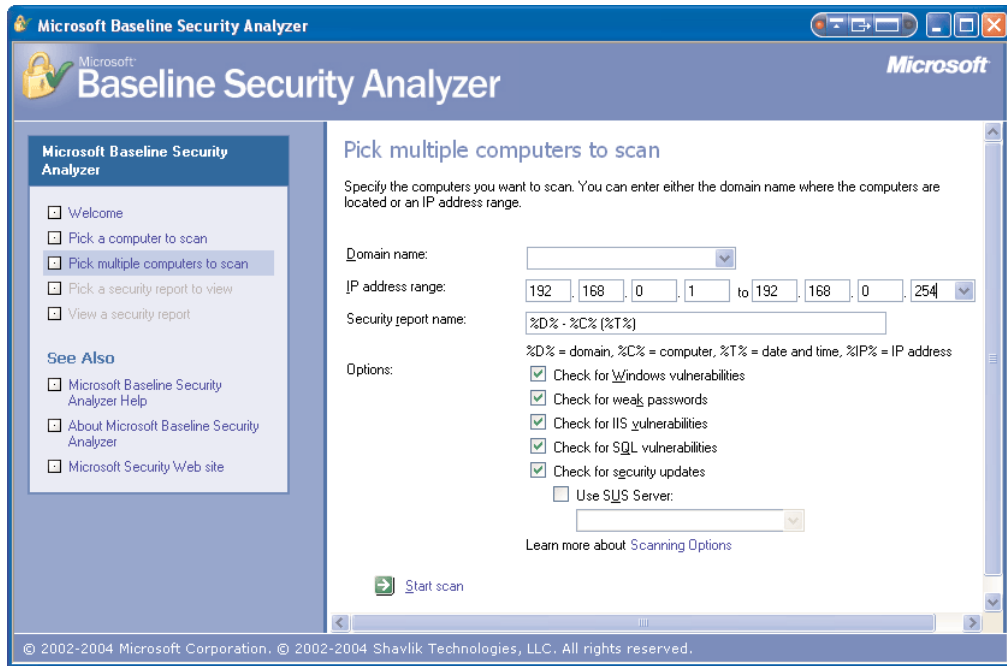
Every time you run an MBSA scan, the program attempts to download a file called `mssecure.cab` from the Microsoft Web site. If your computer is not connected to the Internet, you will need to download the XML file manually to update MBSA and have new patches reflected in the reports. This compressed XML file contains all the most recent software updates. Optionally, if you host an SUS server, you can direct MBSA to obtain its list of approved updates from that server instead of directly from the Microsoft Web site. Consequently, your reports will reflect only updates that you approved with your SUS server. MBSA's ability to reference and use your list of previously approved SUS updates helps you enforce your corporate update policy without the distractions of false positives from unapproved updates. For example, you might use SUS as the gatekeeper to manage the rollout of new updates to your end users. After you've assessed the applicability and tested the compatibility of a particular update, you approve its deployment through SUS. Then, depending on your environment's SUS configuration, end users' computers will either automatically download and install the patch or download and prompt them for manual installation. To enforce your update policy, use the graphical MBSA or the command-line `mbsacli` utility to scan your end users' computers for missing updates. Schedule MBSA to run weekly and pull its list of updates to check from your SUS server's list of approved updates. The resulting XML reports will show you which systems haven't been successfully updated with your specifically approved updates.

To perform all the MBSA-supported scans, you need Local Administrator privileges on the target systems. Run `mbsa.exe` to launch the scanner's graphical version. This version provides a simple-to-

use interface so you can quickly specify which scans you want to run and which computers you want to run them on, as Figure 3-1 shows.

**Figure 3-1**

*Selecting computers to scan and which scans to perform*



First, to specify the targets of your scan, in the MBSA GUI click *Pick a computer to scan* or *Pick multiple computers to scan*, then enter an IP address, a range of IP addresses, or a domain name. Next, select the scan options, including *Check for Windows vulnerabilities*, *Check for weak passwords*, *Check for IIS vulnerabilities*, *Check for SQL vulnerabilities*, and *Check for security updates*. Optionally, you can specify an SUS server whose list MBSA will use to compare with each client. Otherwise, MBSA will use the list of all updates that Microsoft provides. By default, the graphical MBSA client performs what Microsoft calls a baseline scan, which scans for and reports on only critical updates (which Windows Update defines) as opposed to all security updates.

## Start Scanning

To begin a scan, click *Start scan*. The length of time a scan takes depends on which options you've chosen. For example, in my environment a comprehensive scan of a 16-computer network comprised of a variety of services, including IIS, SQL Server, and Exchange, took about 5 minutes to finish. By default, MBSA writes the security reports to the `\%userprofile%\securityscans` folder as XML files. MBSA creates a separate XML report for every computer it scans, each time it scans the computer. These reports are generally about 20KB in size.

After you run the scan, click *Pick a security report to view* and select the name of the report you want to view. Although MBSA lets you sort by computer name, IP address, and scan date, you might need to delete old reports to keep the list from cluttering your folder after running multiple scans. In Figure 3-2 an example report shows which critical security updates are missing from a computer.

**Figure 3-2**  
*Showing missing critical security updates*

**Microsoft Baseline Security Analyzer**

**8 critical security updates are missing. 1 products are using a service pack not at the latest version or have other warnings. 5 security updates could not be confirmed.**

**Result Details**

**Windows Security Updates**

Security updates confirmed as missing are marked with a red X

Score	Security Update	Description	Reason
X	<a href="#">MS04-022</a>	Vulnerability in Task Scheduler Could Allow Code Execution (841873)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\mstask.dll, 5.1.2600.1106 < 5.1.2600.1564]
X	<a href="#">MS04-023</a>	Vulnerability in HTML Help Could Allow Code Execution (840315)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\itss.dll, 5.2.3644.0 < 5.2.3790.185]
X	<a href="#">MS04-030</a>	Vulnerability in WebDav XML Message Handler Could Lead to a Denial of Service (824151)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\msxml3.dll, 8.30.9926.0 < 8.50.2162.0]
X	<a href="#">MS04-031</a>	Vulnerability in NetDDE Could Allow Remote Code Execution (841533)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\nddenb32.dll, 5.1.2600.1106 < 5.1.2600.1555]
X	<a href="#">MS04-032</a>	Security Update for Microsoft Windows (840987)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\basesrv.dll, 5.1.2600.1106 < 5.1.2600.1566]
X	<a href="#">MS04-034</a>	Vulnerability in Compressed (zipped) Folders Could Allow Code Execution (873376)	File version is less than expected. [\\192.168.0.104 \C\$\WINDOWS\system32\zipfldr.dll, 6.0.2800.1126 < 6.0.2800.1584]

## ***MBSA Command Line***

The command-line version of MBSA supports two syntax structures: a command-line equivalent of MBSA and a syntax that matches the popular command-line patch-checking tool HFNetChk. (In fact, MBSA replaces the standalone HFNetChk tool.) Run the

```
mbsacli /?
```

command for a listing of command-line options and the

```
mbsacli /hf /?
```

command for a listing of arguments that this improved HFNetChk supports.

The console-based mbsacli lets you use command-line arguments to specify most configuration options. Therefore, you can use any Windows scripting technology to script a wrapper that calls mbsacli to scan multiple systems or networks. You can even schedule a scan to regularly check the status of your domain or specific computers. For example, a scheduled scan that reports only missing updates on one computer might look like

```
mbsacli /n os+sql+iis+passwords /i 192.168.0.151
```

Microsoft understands that many people want to script or schedule such scans, so the product provides several output-suppression and output-redirection options. The following command redirects the scan output to the network share \\wkstn\logon and writes it to the scan.txt file:

```
mbsacli -f \\wkstn\logon\scan.txt -c sl-blvu\dc4
```

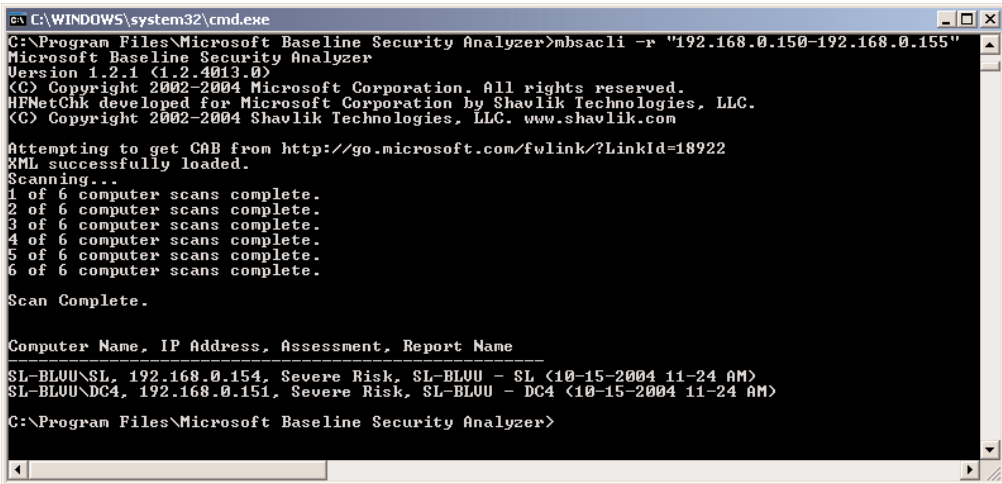
To configure MBSA to pull the list of updates to check for from your SUS server's list of approved updates use the command

```
mbsacli /sus "http://susserver" /i 192.168.0.10
```

## ***Viewing Reports***

Mbsacli doesn't display verbose scan details to the console, as HFNetChk does, but instead displays a summary of results, which Figure 3-3 shows.

**Figure 3-3**  
*Displaying summary scan results*



```

C:\WINDOWS\system32\cmd.exe
C:\Program Files\Microsoft Baseline Security Analyzer>mbsacli -r "192.168.0.150-192.168.0.155"
Microsoft Baseline Security Analyzer
Version 1.2.1 (1.2.4013.0)
(C) Copyright 2002-2004 Microsoft Corporation. All rights reserved.
HFNetChk developed for Microsoft Corporation by Shavlik Technologies, LLC.
(C) Copyright 2002-2004 Shavlik Technologies, LLC. www.shavlik.com

Attempting to get CAB from http://go.microsoft.com/fwlink/?LinkId=18922
XML successfully loaded.
Scanning...
1 of 6 computer scans complete.
2 of 6 computer scans complete.
3 of 6 computer scans complete.
4 of 6 computer scans complete.
5 of 6 computer scans complete.
6 of 6 computer scans complete.

Scan Complete.

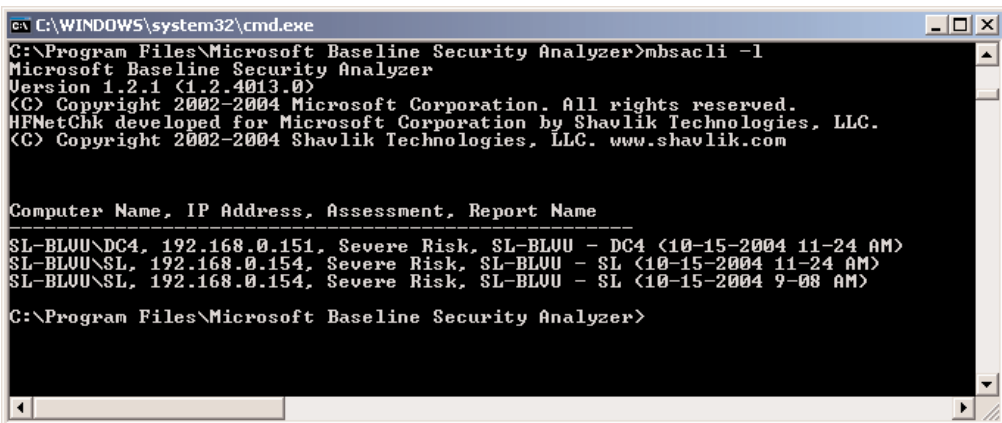
Computer Name, IP Address, Assessment, Report Name
-----
SL-BLUU\SL, 192.168.0.154, Severe Risk, SL-BLUU - SL (10-15-2004 11-24 AM)
SL-BLUU\DC4, 192.168.0.151, Severe Risk, SL-BLUU - DC4 (10-15-2004 11-24 AM)

C:\Program Files\Microsoft Baseline Security Analyzer>

```

However, `mbsacli` generates the same XML reports that the graphical version of MBSA creates and also supports command-line arguments for listing and displaying these reports. For example, as Figure 3-4 shows, the `mbsacli -l` command lists all XML reports that reside under the user profile of the person running the command (`%userprofile%\securityscans`).

**Figure 3-4**  
*Listing XML reports that reside under the user's profile*



```

C:\WINDOWS\system32\cmd.exe
C:\Program Files\Microsoft Baseline Security Analyzer>mbsacli -l
Microsoft Baseline Security Analyzer
Version 1.2.1 (1.2.4013.0)
(C) Copyright 2002-2004 Microsoft Corporation. All rights reserved.
HFNetChk developed for Microsoft Corporation by Shavlik Technologies, LLC.
(C) Copyright 2002-2004 Shavlik Technologies, LLC. www.shavlik.com

Computer Name, IP Address, Assessment, Report Name
-----
SL-BLUU\DC4, 192.168.0.151, Severe Risk, SL-BLUU - DC4 (10-15-2004 11-24 AM)
SL-BLUU\SL, 192.168.0.154, Severe Risk, SL-BLUU - SL (10-15-2004 11-24 AM)
SL-BLUU\SL, 192.168.0.154, Severe Risk, SL-BLUU - SL (10-15-2004 9-08 AM)

C:\Program Files\Microsoft Baseline Security Analyzer>

```

You can use the `-ld` report name option to access the reports. For example, using the data from Figure 3-4, you can use the following command to display the most recent scan of the computer called `dc4`:

```
mbsacli -ld "SL-BLUU - DC4 (10-15-2004 11-24 AM)"
```

As Figure 3-5 shows, this command displays a text interpretation of the XML report that you can parse. However, using XML scripting technologies to directly extract the data provides greater flexibility and control over the data.

**Figure 3-5**  
*Displaying a text interpretation of an XML report*

```

C:\WINDOWS\system32\cmd.exe
C:\Program Files\Microsoft Baseline Security Analyzer>mbsacli -id "SL-BLUN - DC4 (10-15-2004 11-24 AM)"
Microsoft Baseline Security Analyzer
Version 1.2.1 (1.2.4013.0)
(C) Copyright 2002-2004 Microsoft Corporation. All rights reserved.
HFNetChk developed for Microsoft Corporation by Shavlik Technologies, LLC.
(C) Copyright 2002-2004 Shavlik Technologies, LLC. www.shavlik.com

Attempting to get CAB from http://go.microsoft.com/fwlink/?LinkId=18922
XML successfully loaded.

Computer name: SL-BLUN-DC4
IP address: 192.168.0.151
Security report name: SL-BLUN - DC4 (10-15-2004 11-24 AM)
Scan date: 2004-10-15 11:24:19
Security update database version: 2004.10.12.0

Security assessment: Severe Risk

Security Updates Scan Results
Issue: Windows Security Updates
Score: Severe Risk
Result: 7 critical security updates are missing. 3 security updates could not be confirmed.
Detail:
Security Update | Description |
MS04-023 | Vulnerability in HTML Help Could Allow Code Execution (840315) |
MS04-030 | Vulnerability in WebDav XML Message Handler Could Lead to a Denial of Service (82
MS04-031 | Vulnerability in NetDDE Could Allow Remote Code Execution (841533) |
MS04-032 | Security Update for Microsoft Windows (846987) |
MS04-034 | Vulnerability in Compressed (zipped) Folders Could Allow Code Execution (873376)
MS04-037 | Vulnerability in Windows Shell Could Allow Remote Code Execution (841356) |
MS04-038 | Cumulative Security Update for Internet Explorer (834707) |
MS04-039 | Unchecked Buffer in DirectX Could Enable System Compromise (819696) |
MS04-016 | Vulnerability in DirectPlay Could Allow Denial of Service (839643) |
MS04-028 | Buffer Overrun in JPEG Processing (GDI+) Could Allow Code Execution (833987) |

Issue: Office Security Updates
Score: Check Not Performed
Result: This scan can only be performed on a local machine.
Issue: Windows Media Player Security Updates
Score: Strong Security
Result: No critical security updates are missing.
Issue: MDAC Security Updates
Score: Strong Security
Result: No critical security updates are missing.
Issue: MSXML Security Updates
Score: Potential Risk
Result: 2 products are using a service pack not at the latest version or have other warnings.
Detail:

```

## ***MBSA as HFNetChk Replacement***

Although HFNetChk doesn't provide the security checking or XML reporting that MBSA offers, it does provide a quick and easy method of listing all missing updates on a specific computer. Whereas MBSA defaults to performing baseline scans, HFNetChk scans all security-related updates. Using the `-b` switch to force HFNetChk to perform a baseline scan looks like

```
mbsacli -hf -b
```

If you want to simply view all security updates missing on a specific server, run the command

```
mbsacli -hf -h sl
```

where `-hf` instructs `mbsacli` to use the HFNetChk argument parser and `-h sl` specifies the host named `sl`. The output of this command displays all missing updates, as Figure 3-6 shows.

Figure 3-6

Displaying all missing updates using the HFNetChk-based scanner

```

C:\WINDOWS\system32\cmd.exe
C:\Program Files\Microsoft Baseline Security Analyzer>mbsacli -hf -h s1
Microsoft Baseline Security Analyzer
Version 1.2.1 (1.2.4013.0)
(C) Copyright 2002-2004 Microsoft Corporation. All rights reserved.
HFNetChk developed for Microsoft Corporation by Shavlik Technologies, LLC.
(C) Copyright 2002-2004 Shavlik Technologies, LLC. www.shavlik.com

Please use the -v switch to view details for
Patch NOT Found, Warning and Note messages

Scanning s1
Attempting to get CAB from http://go.microsoft.com/fwlink/?LinkId=18922
XML successfully loaded.

Done scanning s1
-----
SL <192.168.0.154>
-----

* WINDOWS SERVER 2003, STANDARD EDITION GOLD

Note MS03-030 819696
Note MS04-016 839643
Patch NOT Found MS04-023 840315
Note MS04-028 833987
Patch NOT Found MS04-030 824151
Patch NOT Found MS04-031 841533
Patch NOT Found MS04-032 840987
Patch NOT Found MS04-034 873376
Patch NOT Found MS04-037 841356

* INTERNET EXPLORER 6.0 FOR WINDOWS SERVER 2003 GOLD

Patch NOT Found MS04-038 834707

* WINDOWS MEDIA PLAYER 9 SERIES FOR WINDOWS SERVER 2003 GOLD

Information
All necessary hotfixes have been applied.

* MDAC 2.8 GOLD

Information
All necessary hotfixes have been applied.

* MSXML 2.6 SP3

Information
There are no security updates available for this product.

* MSXML 3.0 SP4

Information
There are no security updates available for this product.

Warning
The latest service pack for this product is not installed.

```

Notice that MBSA reports several Note messages and one Warning message. MBSA (both the graphical and command-line version) displays these messages when it can't determine whether an update has been installed or to notify a user of a security problem that an update can't fix. For example, if an update exists for both Microsoft XML Core Services (MSXML) 4.0 and MSXML 3.0 and the target machine has MSXML 4.0 installed, MBSA might display a note informing you that the MSXML 3.0 update hasn't been installed. The Microsoft article "Microsoft Baseline Security Analyzer (MBSA) returns note messages for some updates" at <http://support.microsoft.com/?kbid=306460> lists the explanations behind many of these notes and warnings.

MBSA reports these notes for every scan. However, you can use the `-s n` argument to disable notes and exclude them from your reports. Use the `-s 1` argument to suppress notes and use `-s 2` to suppress both notes and warnings.

## **MBSA Limitations**

Although MBSA performs admirably as an all-in-one update-checking tool and basic Microsoft product security-configuration checker, it has limitations. MBSA doesn't scan for Office updates or updates that aren't related to security, so you'll need to rely on other tools to report those updates. MBSA is strictly a scanner and doesn't deploy patches or remediate misconfigurations. (However, it provides useful Help documents that walk you through the remediation of any discovered vulnerability.)

## **The Timeline from Test to Production**

Testing is a valuable step to reducing the risk involved in deploying a new security update. The two hurdles to an effective testing regiment are *cost* and *time*. A lab can be expensive in terms of server costs and the resources needed to build and manage the lab computers and supporting infrastructure. The second hurdle to a comprehensive testing program is the time needed to evaluate the update, install the update in the lab, perform adequate testing of the patch against your systems and applications, then formally approve the update for deployment to production. These steps usually take days to weeks, which can significantly increase the risk to your frontline systems—especially those susceptible to an attack vector.

Deciding what amount of testing is the right amount is difficult. As guidance you will find the most success in creating a lab that is representative of production and defining a set of test procedures that you can execute for new patches. Next, when updates are released, spend time to learn about their types, the vulnerabilities they address, and any mitigating factors. This knowledge will help you establish a timeline to work with and assess the risk of delaying the deployment while you execute your tests. Also, the security bulletins should provide an understanding of the intrusiveness of the updates. A service pack will require more testing than the update of one, less frequently used application. Lastly, execute your test procedures in full before and after applying the update. This step will give you confidence (as well as punctuate any tactical steps necessary during patch deployment) when you deploy the patch in production.