

## IIS 6.0 Security

by [Rohyt Belani](#) and [Michael Muckin](#)

last updated March 5, 2004

The popularity of web servers as a prime target for crackers and worm writers around the globe made IIS a natural place for Microsoft to focus its Trustworthy Computing Initiative. As a result, IIS has been completely redesigned to be secure by default and secure by design. This article discusses the major default configuration and design changes incorporated in IIS 6.0 to make it a more secure platform for hosting critical web applications.

### Secure by Default

In the past, vendors including Microsoft packaged the default installations of their web servers with an array of sample scripts, file handlers and minimal file-system permissions to provide administrators the necessary flexibility and ease of use. However, this approach tended to increase the available attack surface and was the basis of several attacks against IIS. As a result, IIS 6.0 is designed to be more secure out-of-the-box than its precursors. The most noticeable change is that IIS 6.0 is not installed by default with Windows Server 2003. Other changes include:

- **Default installation is only a static HTTP server**

The default installation of IIS 6.0 is configured to serve static HTML pages only; dynamic content is not permitted. The following table compares the default features of IIS 5.0 and IIS 6.0.

IIS Component	IIS 5.0 default install	IIS 6.0 default install
Static file support	Enabled	Enabled
ASP	Enabled	Disabled
Server-side includes	Enabled	Disabled
Internet Data Connector	Enabled	Disabled
WebDAV	Enabled	Disabled
Index Server ISAPI	Enabled	Disabled
Internet Printing ISAPI	Enabled	Disabled
CGI	Enabled	Disabled
Microsoft FrontPage® server extensions	Enabled	Disabled
Password change interface	Enabled	Disabled
SMTP	Enabled	Disabled
FTP	Enabled	Disabled
ASP.NET	N/A	Disabled
Background Intelligence Transfer Service	N/A	Disabled

- **No sample applications installed**

IIS 6.0 does not include any sample scripts and applications like showcode.asp and codebrws.asp. These programs were originally designed to let programmers quickly look at their database connection code in order to debug it. However, Showcode.asp and codebrws.asp do not correctly check the input to ensure that the file being requested is within the web root directory. This allows the attacker to read any file (including those containing sensitive information and insightful configuration settings) on the system by traversing back to it. Refer to the following link for more information regarding these vulnerabilities:

<http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS99-013.asp>

- **Improved file-system access controls**

Anonymous users no longer have write access to the home directory of the Web server. In addition, FTP users are isolated in their own home directories. These limitations prevent a user from uploading malicious files to other parts of the server's file system. Such attacks may include web site defacement by uploading files to the web document root and remote command execution via the execution of malicious executables that may be uploaded to the /scripts directory.

- **No Executable Virtual Directories**

No virtual directories have executable permissions on them. This prevents exploitation of the numerous directory traversal, code upload and MDAC exploits that have existed in the past.

- **Sub-authentication module removed**

The IISSUBA.dll has been removed from IIS 6.0. Any accounts that required this functionality in previous versions of IIS required the "access this computer from the network" privilege. Removal of this DLL removes this dependency and thus reduces the attack surface by forcing all authentications directly to the SAM or Active Directory.

- **Parent Paths are disabled**

Access to parent paths in the file system is disabled by default. This is to prevent directory traversal attacks that may allow an attacker to break out of the web document root and gain access to sensitive files on the file system, such as the SAM file. Note that this can however cause problems for migrated applications that used parent paths on previous versions of IIS.

## **Secure by Design**

The fundamental design changes incorporated in IIS 6.0 include improved data validation, enhanced logging, rapid-fail protection, application isolation and adherence to the principle of least privilege.

### **Improved data validation**

A principal new feature incorporated in the design of IIS 6.0 is the kernel-mode HTTP driver, HTTP.sys. It is not only tuned to enhance the web server's performance and scalability characteristics, but also to significantly strengthen the security posture of the server. HTTP.sys acts as the gateway for user requests to the web server. It first parses the request and then dispatches it to the appropriate user-level worker processes. The restriction of the worker processes to the user-mode prevents them from accessing privileged resources in the system kernel. Thus the target space of an attacker intending to gain privileged access to the server is greatly limited.

The kernel-mode driver incorporates several security mechanisms to augment the inherent secure design of IIS 6.0. These features include protection against potential buffer overflows, improved logging mechanisms to aid the process of incident response and advanced URL parsing to check for the validity of user requests.

In order to impede the exploitation of a potential buffer or memory overflow vulnerability that may arise at a later point in time, Microsoft has resorted to the defense-in-depth principle of security in the design of IIS 6.0. This has been accomplished by adding specific URL parsing capabilities to the repertoire of features incorporated in HTTP.sys. These capabilities can be further fine-tuned by appropriately modifying specific registry values. The following table provides a brief overview of vital registry keys (found at the following path:

HKLM\System\CurrentControlSet\Services\HTTP\Parameters):

AllowRestrictedChars	This key accepts a Boolean value, which if non-zero allows HTTP.sys to accept hex-encoded characters in the request URL. The default value for this key is 0. This is also the recommended value as it facilitates the task of input validation at the server-level. If set to 1, potentially malicious characters may be hex-encoded by the attacker in an attempt to bypass input validation routines.
MaxFieldLength	This key allows the administrator to set an upper limit (in bytes) for each header. Its default value is 16KB.
MaxRequestBytes	This key establishes the upper limit on the total size of the request line and the headers. Its default value is also 16KB.
UrlSegmentMaxCount	This key determines the maximum number of URL path segments accepted by the server. It effectively limits the number of slashes that can be included by the user in a request URL. It is recommended that one set fairly stringent limits on this value based on the depth of the web document root tree to protect the server from a file system traversal attack. The default value for this key is 255.
UrlSegmentMaxLength	This key sets an upper bound on the maximum number of characters in any URL path segment. This value can also be customized in accordance with the normal operation of the hosted applications to prevent the acceptance of unusually long segments that may cause the application to behave in an anomalous manner. The default value for this key is 260.
EnableNonUTF8	The value of this key controls the character set that is permitted by HTTP.sys. The default value of 1 permits HTTP.sys to accept ANSI- and DBCS-encoded URLs in addition to those encoded in the UTF8 format.

## Enhanced logging mechanisms

Comprehensive logging is a fundamental requirement for the successful detection of, and response to, a security

incident. Microsoft has recognized this need and implemented an extensive and reliable logging mechanism in HTTP.sys. HTTP.sys writes to the log file before dispatching the request to the specific worker process. This ensures that an error condition is logged even if it causes the worker process to terminate. An entry in the log file consists of the date and time stamp of the occurrence of the error condition, the source and destination IP addresses and ports, the protocol version, HTTP verb, the URL, protocol status, the site ID and the HTTP.sys reason phrase. The reason phrase provides detailed information about why the error occurred - whether it can be attributed to a timeout condition or a connection being abandoned by the application pool due to the unexpected termination of the worker process.

An example HTTP.sys log file entry can be found at the following link:

[http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/iis/iis6/proddocs/resguide/iisrg\\_log\\_qlow.asp](http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/iis/iis6/proddocs/resguide/iisrg_log_qlow.asp)

## **Rapid-Fail Protection**

In addition to tweaking the registry, an IIS 6.0 administrator can also configure the server to automatically shutdown or restart worker processes whose applications have failed repeatedly (a set number of times) within a specific period of time. This is an additional safeguard to protect the application against repeated failures, which may be an indication of attack. This feature is called *Rapid-Fail Protection*.

Rapid-Fail protection can be configured through IIS manager as follows:

1. In IIS Manager, expand local computer.
2. Expand **Application Pools**
3. Right click on application pool
4. Click on **Properties**
5. On the **Health** tab, check the **Enable rapid-fail protection** box
6. In the **Failures** box, type the number of worker process failures to be tolerated (before shutting down the process).
7. In the **Time Period** box, specify the number of minutes for which worker process failures are allowed to accumulate.

## **Application isolation**

In previous versions of IIS (version 5.0 and earlier), the performance penalty for segregating web applications into independent units made it infeasible to do so. Thus, more often than not the failure or compromise of one web application had a cascading effect to the other applications resident on the same web server. However, performance enhancements coupled with design changes to the request processing architecture of IIS 6.0 have made it viable to isolate applications into self-contained units called application pools (without affecting performance). Each application pool is served by one or more independent worker processes. This allows for the localization of failure, preventing the malfunction of one worker process from affecting the others. This boosts the reliability of the server and in turn that of the applications hosted by it.

## **Adherence to principle of least privilege**

IIS 6.0 adheres to a fundamental tenet of security - the principle of least privilege. This is achieved by including all the code that needs to run with Local System (high-level) privileges in HTTP.sys. All the worker processes execute as Network Service, a new type of account built-in to Windows 2003 with extremely limited operating system rights. Further, IIS 6.0 only allows system administrators to execute command-line tools, thus preventing malicious exploits from using these tools. These design changes reduce the exposure due the compromise of the server via a potential vulnerability. Apart from these fundamental design changes, some simple configuration modifications include denying anonymous users write access to the home directory of the web server and isolating FTP users into their own home directories also greatly enhance the security posture of IIS 6.0.

IIS 6.0 is a step in the right direction, by Microsoft, to help organizations improve their security postures. It provides a reliable and secure infrastructure to host web applications. The improved security can be attributed to the secure default configuration, the evident emphasis to security in the design process and enhanced monitoring and logging capabilities of the IIS 6.0 server. However, administrators should not acquire a false sense of security by simply migrating to this platform but couple it with the implementation of multiple layers of security. This would be in accordance with the defense-in-depth tenet of security to safeguard against the ever-looming threat of another Code Red or Nimda.

### **Author Credit**

This document is co-authored by [Rohyt Belani](#) and [Michael Muckin](#).