

IIS Lockdown and Urlscan

Nishchal Bhalla and Rohyt Belani 2004-01-05

The security posture of a web application can be severely undermined if the underlying web server software is vulnerable. The web server software is the most visible and easy to exploit part of a web application. Even if the web application itself is impregnable it can be subject to serious security breaches if the underlying web server platform is insecure.

As one of the more widely deployed web servers, Microsoft's IIS has been a frequent target for attackers over the last few years. It has been beleaguered by vulnerabilities such as source code disclosure attacks like \$DATA, information exposures through sample scripts like showcode.asp, and easily exploited buffer overflow vulnerabilities which have fueled Internet-borne worms like Code Red and NIMDA. Such attacks emphasize the importance of web server security and more specifically IIS security. This article discusses two important vendor-provided tools (IIS Lockdown and Urlscan) that target significant security-related configuration problems for IIS versions 6.0, 5.0, and earlier.

IIS Lockdown

The default installation of most web servers do not satisfy the security needs of all administrators. Microsoft's IIS (particularly versions 5.0 and earlier) is no exception. It is packaged with several sample scripts, minimal file-system permissions and a plethora of file handlers. Vendors adopt this strategy to provide administrators the flexibility to tailor the security configuration to the business needs of their organization. The IIS administrator can accomplish this task either by manually configuring the server or by utilizing Microsoft's [IIS Lockdown tool](#).

The tool provides a centralized GUI interface to un-map a specific list of Directories, Methods and Services which could pose a security threat to the web server and hence the resident web applications. In this section we will cover some basic configurations of the IIS Lockdown tool.

The novice user may opt to apply one of the many default templates provided by IIS Lockdown. These include - Small Business Server 2000, Exchange Server, FrontPage Server Extensions, Dynamic Web Server and Static Web Server. Based on the selected template, only select ISAPI DLL mappings are retained. It is however important to note that IIS Lockdown only un-maps these ISAPI extensions, it does not uninstall or un-register the DLLs. Thus, the configuration of

the server can be restored to its original format by simply re-running the IIS Lockdown tool against the web server. Note that IIS 6 installs without any extras by default, and will only serve static pages until you configure it otherwise.

The more advanced user may, however, choose to manually configure the security settings to be subsequently applied by IIS Lockdown. These settings include:

1. **Disabling unnecessary services**

IIS allows for three basic services - the Web Service, the FTP Service and the SMTP Service. IIS Lockdown provides an option to disable and/or remove one or all of these services that are not required. This is a necessary step in securing the server as the existence of un-patched, unused services may well go unnoticed by the administrator and prove to be a playground for potential attackers.

2. **Un-mapping unused file handlers**

The functionality embodied in the various ISAPI DLLs can be invoked simply by requesting a file with the appropriate extension from IIS. Out of the box, IIS 5.0 and its precursors are provided with a large number of potentially unused DLLs. The extensions mapped by default include .htw, .ida, .idq, .asp, .cer, .cdx, .asa, .htr, .idc, .shtm, .shtml, .stm and .printer. The most commonly used ones are .asp (for server side scripts that help generate dynamic HTML), .asa (global configuration file, generally contains global variables and connect strings to the back end database), .cer (used for https communication) and .cdx (used for https communication). The other extensions such as .printer (provides internet printing capability) are rarely used. The DLLs supporting these extensions should be un-mapped, thus depriving erstwhile hackers with a myriad of different functionality available for exploit via malicious input. IIS Lockdown allows the user to perform this operation with relative ease through the user-friendly GUI.

3. **Un-mapping sample scripts and their directories**

Sample scripts and applications included in the default IIS installation pose a serious threat to the security posture of the server and its resident applications. This is due to the fact that the primary purpose of sample programs is to exhibit functionality and they do not incorporate the highest level of security. Thus, they do not include input validation

routines to prevent the acceptance of malicious input from a potential attacker. This could result in compromises such as disclosure of the source code of critical applications and arbitrary command execution on the IIS server.

This issue is addressed by the IIS Lockdown tool by un-mapping these sample files and the directories in which they are contained. It is important to note that the tool does not delete the files and they can be restored if need be.

4. Modifying critical file access permissions

IIS Lockdown modifies access permissions to the web root directory (InetPub\wwwroot). This action denies an anonymous IIS user the ability to create or delete files and folders, create or modify data and change file attributes within this directory. The permissions pre- and post-execution of IIS Lockdown are

Pre Installation	Post Installation
Everyone Read-Only	Machine\Web DENIED
NT AUTHORITY\SYSTEM Full Control	Applications DwawE
BUILTIN\Administrators Full Control	Machine\Web DENIED
	Anonymous Users DwawE
	Everyone Read-Only
	NT AUTHORITY\SYSTEM Full Control
	BUILTIN\Administrators Full Control

The tool also changes file permissions in the %Windir% directory. These changes prevent the remote execution of system utilities like cmd.exe, tftp.exe and edit.com. Even though an anonymous IIS user cannot access these utilities remotely, the exploitation of a new or existing vulnerability could provide a channel for such access. However, the explicit removal of permissions (by the IIS Lockdown tool) plugs the hole punched by such vulnerabilities.

5. Editing WebDAV access permissions

Web Distributed Authoring and Versioning (WebDAV) is a facility that allows users to remotely collaborate and manage files on a particular IIS web server. This functionality is

provided by httpext.dll. The IIS Lockdown tool denies the "Everyone" group permission to execute this DLL. This action decreases the risk of unauthorized users uploading malware to and deleting critical files on the web server.

Although IIS Lockdown tackles most of the security concerns of an IIS administrator, it is not a comprehensive solution for IIS security. The validation of client input, for example, is a major issue not tackled by IIS Lockdown. Maliciously formed URLs can result in serious security hazards. This can be prevented by using Microsoft's Urlscan (now a part of IIS Lockdown), a tool that monitors and filters the content of URLs before they are processed by the server.

Urlscan

Many attacks launched against web servers involve a maliciously crafted URL. The URL may be unusually long, may be encoded by an alternate character set or may include character sequences which are not common to a legitimate request. Such URLs, if processed by the IIS server, may cause severe damage to the server and/or the web site hosted by it. In order to prevent this, Microsoft has developed a tool known as [Urlscan](#).

As the name suggests, Urlscan scans all incoming URL requests. Based on a set of pre-established rules, Urlscan filters out the request and sends only valid data to the server process. It provides an option to store the filtered requests in a log file.

It is important to note that Urlscan is practically obsolete with IIS 6.0. Most of the features provided by Urlscan have either been implemented by default in IIS or can be enabled by simply modifying registry keys. Therefore, this part of the article will apply primarily to versions 5.0 and earlier.

Urlscan consists of two key files: urlscan.dll and urlscan.ini which reside in the %systemroot%\inetrv\Urlscan directory. The default urlscan.ini is archived [here](#).

Urlscan.dll is an ISAPI filter that is self-registered when installed through IIS Lockdown/Urlscan. It can be manually registered through the Internet Services Manager interface as well. It pre-processes all requests to the IIS server looking for malicious input as defined in the **Urlscan.ini** configuration file. Rejected requests are logged to the **Urlscan.log** file located in the same directory as the other Urlscan files.

The Urlscan.ini file holds the key to successful prevention of attacks against the IIS server. The

remainder of this article thus focuses on these configurations due to their paramount importance.

The Urlscan.ini file has two main parts: options and implementation. The options part of the file allows the user to enable or disable a particular option while the latter supports the actual configuration of the enabled options.

Options and Implementations:

The options take a value of either "0" or "1". Typically "1" is to enable the option and "0" to disable the option, or "1" is to explicitly allow certain implementations, whereas "0" would deny only the actions specified in the list of implementations and allow all the other default actions. Additionally the semicolon ";" is used to mark the beginning of a comment in the urlscan.ini file.

UseAllowVerbs

Some typical verbs for a web-server are "GET", "HEAD", "POST", "DEBUG", "TRACE", "OPTIONS", "PUT", "DELETE". Additionally, there are the WebDAV (Web Document Authoring and Versioning) verbs like "PROPFIND", "MOVE" etc. This option works in conjunction with the implementation section's "AllowVerbs" and "DenyVerbs".

The default value for "UseAllowVerbs" is "1". If the value is set to "1", then only the verbs that are explicitly specified in the "AllowVerbs" section are passed on to the web-server and other verbs are rejected. If the value is set to "0" then all the verbs are passed on to the web-server except for those specified in the "DenyVerbs".

The value should be set to "1". A typical web site needs only "GET", "HEAD" and "POST" requests. However, to find the list of verbs that are typically used by a website, an administrator can review the website logs, in the default location "%system32%\Logfiles\W3SVC1*.log".

This option could help prevent an attacker from using a verb to attack a remote system by disallowing the verb from

being used (either explicitly or implicitly). An example of an attack that could be prevented is the TRACE cross site scripting vulnerability or the PROPFIND vulnerability, whereby an attacker can enumerate the internal IP address of a web server by simply making a "PROPFIND" request with the "Content-Length:" of zero.

UseAllowExtensions

Some of the extensions that are mapped on a server are ".asp", ".aspx", ".html", ".exe", ".bat", ".cmd", ".com", ".htr", ".printer". Many of these extensions need only be executed on the server and don't need to be actually sent to the web.

This option works in conjunction with the implementation section's "AllowExtensions" and "DenyExtensions". The default value for "UseAllowExtensions" is "0". This value allows all extensions to be processed by the server except for the extensions listed in the "DenyExtensions" implementation section. If the value is "1", then the "AllowExtensions" implementation section is processed, which will only allow particular file extension requests to be passed to the web server and deny all the other extensions.

The value should be set to "1". A typical web site needs to allow only ".asp", ".aspx", ".cer", ".cdx", ".asa", ".html", ".js", ".htm", ".jpg", ".jpeg", ".gif" extensions. To determine the extensions that should be allowed, an administrator should not only review the logs to determine the files being requested, but also review the directory tree of the website.

This option could help prevent an attacker from abusing extensions to attack a remote system by disallowing those extensions from being processed. An example of an attack that could be prevented would be the ".+htr" attack, where by an attacker could view the contents of the "global.asa" file by modifying the request to "global.asa+.htr".

Similar to a firewall ruleset, it is recommended that one

always try and deny everything by default and then explicitly allow specific verbs (UseAllowVerbs - AllowVerbs) and extensions (UseAllowExtensions - AllowExtensions).

NormalizeUrlBeforeScan Some of the requests that could be sent back to the server could be either hex encoded, URL Encoded or UTF encoded. For example, %20 indicates the hexadecimal value for a space character in ASCII. Files on a web server can be requested using alternate representation. The use of "NormalizeUrlBeforeScan" causes the URL to be canonicalized before processing. All the characters would be decoded/normalized before a request is processed. The default value is "1" which would normalize the request before passing it to the server process.

The value should be set to "1", however it is known to break various web applications. The cause of this failure is typically because the application expects to receive encoded characters and tries to process regular characters as encoded characters.

This option could help prevent an attacker from requesting information from a server by encoding the request in order to bypass access controls in the applications. An example of an attack that could be prevented would be the "cgi-bin/..%c0%af../..c0%af../..c0%af../..c0%af../..c0%af../winnt/system32/cmd.exe?/c+dir+c:\", which would list the directory contents of the "c:\", the Unicode attack.

VerifyNormalization

Some of the requests that could be sent back to the server could be encoded multiple times. For example, %255c is double encoded for "\". The hex encode character for "\" is %5c. The hex encoded character for "%" is "%25". Thus the double encoded character for "\" would yield "%255c".

When "VerifyNormalization" value is set to "1", the default value, it causes the URL to be canonicalized twice to verify the resulting value with the result of the previous canonicalization. If the two differ the request is rejected.

The value should be set to "1", however it is known to break various web applications. The cause of this failure is typically because the application expects to receive encoded characters and tries to process regular characters as encoded characters.

This option could help prevent an attacker from requesting information from a server by double-encoding the request in order to bypass access controls in the applications. An example of an attack that could be prevented would be the scripts/..%255c..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\", which would list the directory contents of the "c:\", the double-encode attack.

AllowHighBitCharacters

Some of the requests that might be sent back to the server could contain non-ASCII characters. These characters would typically be UTF-8 encoded, and a site that requires international language support would use this character set.

When AllowHighBitCharacters is set to "0", the default value, the server won't process characters which are non-ASCII. However, if the value is set to "1", the server processes the request directly.

For a site that contains only ASCII characters, the value

should be set to "0", however for a site which might contain additional characters (multiple language support), the value should be set to "1". This option could help prevent an attacker from requesting information from a server by encoding the request in order to bypass access controls in the applications. An example of an attack that could be prevented would be the "cgi-bin/..%c0%af../..%c0%af../..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\", which would list the directory contents of the "c:\", the Unicode attack.

AllowDotInPath

Many requests that are sent back to the server contain "../". An example of such a request could be a simple reference in the source code itself "../images/company_logo.gif".

If this option is set to "0", the default value, Urlscan rejects any request that contains multiple periods, including the example shown above. Note that this does not negate the server IP address. If this option is set to "1" then Urlscan allows for file or directory requests with multiple periods to be processed by the server.

The default value should be set to "0", however it may break the functionality of many web applications and web sites because of the common use of relative references, such as "../images/company_logo.gif" to reference files versus literal references, like "/images/company_logo.gif". This option could help prevent an attacker from attempting to request information from a server by providing multiple "../" as an argument. An example of an attack that could be prevented would be the "cgi-bin/..%c0%af../..c0%af../..c0%af../..c0%af../..c0%af../..c0%af../winnt/system32/cmd.exe?/c+dir+c:\", which would list the directory contents of the "c:\", the Unicode attack.

RemoveServerHeader

By default when a request is made to the server, the server responds with a list of options. Among the response there is a variable called "Server:" which specifies the exact version of server that is hosting the site.

The default value of this is set to "0" which responds back to the client and displays the server information. If the value is set to "1", the server name is no longer sent to the client. The entire line containing the server string is omitted.

The value should be set to "1". This option could not only help prevent remote attackers from randomly targeting your websites due to the version you are running but also could prevent worms from spreading to your website if the worms are designed to first read the header information before attempting the attack on the server. Note that there are other ways to identify and enumerate a web server, however this approach removes the easiest and most obvious method.

So far there aren't any known attacks against the server string directly, however, knowing the server type and version could make the server a potential target for a zero day exploit.

EnableLogging

This option starts the logging of all activities that are related to the Urlscan ISAPI. It not only logs the requests sent from the client, but also provides information on the settings that are being implemented by the ISAPI. The default value of this is set to "1" to enable logging, however if the value is set to "0", the logging is disabled.

The value should be set to "1". However, it is important to note that some parts of the logging will be duplicated with the standard IIS logs. This option could provide more information on the type of attacks a malicious user was

attempting against the website or it might also help troubleshoot other problems in the web application.

PerProcessLogging

This option could be viewed as an extension to logging. It provides a separate log file for each individual process ID, by appending the Process ID in each log file. This option may aid in debugging a process using the process ID.

The default value "0" doesn't separate the logs per process ID, however if the value is set to "1", the log files are generated on the basis of each and every Process ID.

The recommended value for PerProcessLogging is "0".

AllowLateScanning

This option allows the administrator to load the Urlscan ISAPI as a high or low priority filter. Note that this feature is not required for IIS 6.0 because IIS 6.0 does not depend on filter notifications for its lockdown mechanism.

The default value is "0" which will load the ISAPI as a high priority and thus will process all the requests before they are passed to the server process. When set to "1", Urlscan will be loaded later, first allowing other ISAPI Filters to be used before Urlscan screens the input.

The recommended value for "AllowLateScanning" is "0", however per Microsoft's Website, late scanning has to be enabled (set to "1") to allow for Front page extensions to work.

PerDayLogging

This option allows for either producing daily log files or letting the administrator rotate the logs. The default value which is set to "1" separates the daily logs. A new log file is automatically started at the end of the day (12:00 AM). The log files are labeled with the date in the name - Urlscan.<date>.log.

The recommended value is "1". This would help organize the data more systematically, however when the logs are centralized using a central log server it might require a single master file.

RejectResponseUrl

This option allows for a custom error message to be presented to the client.

The default value would reveal to the User Agent - that Urlscan is being run on the remote system. Note that XSS is common in custom error messages, therefore one should be careful in modifying this default value.

UseFastPathReject

This option is used in conjunction with the RejectResponseUrl. If the value of "UseFastPathReject" is set to "0", then the ISAPI will respond back with the value of the "RejectResponseUrl", however, if the value is set to "1", then the ISAPI will neither use the "RejectResponseUrl" nor log the request. If an ISA server is part of the website architecture, then the Proxy server will log both the request and the response.

The default value for "UseFastPathReject" is "1". Urlscan will ignore the "RejectResponseUrl" and returns a 500 error message to the browser. This is faster than processing RejectResponseUrl, but it does not permit as many logging options.

The recommended value is "1", if the value is set to "0", there is too much information logged.

AlternateServerName

This option allows to customize the "Server" value. This option works in conjunction with "RemoveServerHeader". If the value of "RemoveServerHeader" is "0", then AlternateServerName can be used to specify a replacement for IIS's built in 'Server' header.

The recommended value could be changed to either be a different vendor's name or something unique that would mislead the attacker.

DenyUrlSequences

This option allows you to specify a list of characters to be rejected in the URL. The default options here are "..", "./", "\", ":", "%", "&". Additional values recommended to add to this list are "#", "<", ">", "\$", "@", "!", ",", and "~".

This option could help prevent different attacks against the site including cross site scripting attacks on the URL.

Conclusion

The use of Urlscan in conjunction with IIS Lockdown blocks a multitude of attacks against IIS. However they do not provide a comprehensive solution for the IIS administrator. There are some security issues that still need to be explicitly addressed by the IIS administrator. These include: implementation of a patch management process to ensure that the both the IIS server and the underlying operating system are up to date with security fixes, hardening the underlying operating system security and the removal of un-mapped or unused DLLs to prevent their accidental re-mapping. However, IIS lockdown and Urlscan go a long way in securing an IIS server.

Reference

Default [Urlscan.ini](#) provided with Microsoft's Urlscan.

[Privacy Statement](#)

Copyright 2006, SecurityFocus