the CENTER for INTERNET SECURITY

Security Configuration Benchmark For

Microsoft IIS 7.0

Version 1.0.0

December 29th, 2010

Copyright 2001-2010, The Center for Internet Security
http://cisecurity.org
feedback@cisecurity.org

Background.

CIS provides benchmarks, scoring tools, software, data, information, suggestions, ideas, and other services and materials from the CIS website or elsewhere ("**Products**") as a public service to Internet users worldwide. Recommendations contained in the Products ("**Recommendations**") result from a consensus-building process that involves many security experts and are generally generic in nature. The Recommendations are intended to provide helpful information to organizations attempting to evaluate or improve the security of their networks, systems and devices. Proper use of the Recommendations requires careful analysis and adaptation to specific user requirements. The Recommendations are not in any way intended to be a "quick fix" for anyone's information security needs.

No representations, warranties and covenants.

CIS makes no representations, warranties or covenants whatsoever as to (i) the positive or negative effect of the Products or the Recommendations on the operation or the security of any particular network, computer system, network device, software, hardware, or any component of any of the foregoing or (ii) the accuracy, reliability, timeliness or completeness of any Product or Recommendation. CIS is providing the Products and the Recommendations "as is" and "as available" without representations, warranties or covenants of any kind.

User agreements.

By using the Products and/or the Recommendations, I and/or my organization ("**we**") agree and acknowledge that:

No network, system, device, hardware, software or component can be made fully secure;
We are using the Products and the Recommendations solely at our own risk;

We are not compensating CIS to assume any liabilities associated with our use of the Products or the Recommendations, even risks that result from CIS's negligence or failure to perform;

We have the sole responsibility to evaluate the risks and benefits of the Products and Recommendations to us and to adapt the Products and the Recommendations to our particular circumstances and requirements;

Neither CIS, nor any CIS Party (defined below) has any responsibility to make any corrections, updates, upgrades or bug fixes or to notify us if it chooses at it sole option to do so; and

Neither CIS nor any CIS Party has or will have any liability to us whatsoever (whether based in contract, tort, strict liability or otherwise) for any direct, indirect, incidental, consequential, or special damages (including without limitation loss of profits, loss of sales, loss of or damage to reputation, loss of customers, loss of software, data, information or emails, loss of privacy, loss of use of any computer or other equipment, business interruption, wasted management or other staff resources or claims of any kind against us from third parties) arising out of or in any way connected with our use of or our inability to use any of the Products or Recommendations (even if CIS has been advised of the possibility of such damages), including without limitation any liability associated with infringement of intellectual property, defects, bugs, errors, omissions, viruses, worms, backdoors, Trojan horses or other harmful items.

Grant of limited rights.

CIS hereby grants each user the following rights, but only so long as the user complies with all of the terms of these Agreed Terms of Use:

Except to the extent that we may have received additional authorization pursuant to a written agreement with CIS, each user may download, install and use each of the Products on a single computer;

Each user may print one or more copies of any Product or any component of a Product that is in a .txt, .pdf, .doc, .mcw, or .rtf format, provided that all such copies are printed in full and are kept intact, including without limitation the text of this Agreed Terms of Use in its entirety.

Retention of intellectual property rights; limitations on distribution.

The Products are protected by copyright and other intellectual property laws and by international treaties. We acknowledge and agree that we are not acquiring title to any intellectual property rights in the Products and that full title and all ownership rights to the Products will remain the exclusive property of CIS or CIS Parties. CIS reserves all rights not expressly granted to users in the preceding section entitled "Grant of limited rights." Subject to the paragraph entitled "Special Rules" (which includes a waiver, granted to some classes of CIS Members, of certain limitations in this paragraph), and except as we may have otherwise agreed in a written agreement with CIS, we agree that we will not (i) decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any software Product that is not already in the form of source code; (ii) distribute, redistribute, encumber, sell, rent, lease, lend, sublicense, or otherwise transfer or exploit rights to any Product or any component of a Product; (iii) post any Product or any component of a Product on any website, bulletin board, ftp server, newsgroup, or other similar mechanism or device, without regard to whether such mechanism or device is internal or external, (iv) remove or alter trademark, logo, copyright or other proprietary notices, legends, symbols or labels in any Product or any component of a Product; (v) remove these Agreed Terms of Use from, or alter these Agreed Terms of Use as they appear in, any Product or any component of a Product; (vi) use any Product or any component of a Product with any derivative works based directly on a Product or any component of a Product; (vii) use any Product or any component of a Product with other products or applications that are directly and specifically dependent on such Product or any component for any part of their functionality, or (viii) represent or claim a particular level of compliance with a CIS Benchmark, scoring tool or other Product. We will not facilitate or otherwise aid other individuals or entities in any of the activities listed in this paragraph.

We hereby agree to indemnify, defend and hold CIS and all of its officers, directors, members, contributors, employees, authors, developers, agents, affiliates, licensors, information and service providers, software suppliers, hardware suppliers, and all other persons who aided CIS in the creation, development or maintenance of the Products or Recommendations ("**CIS Parties**") harmless from and against any and all liability, losses, costs and expenses (including attorneys' fees and court costs) incurred by CIS or any CIS Party in connection with any claim arising out of any violation by us of the preceding paragraph, including without limitation CIS's right, at our expense, to assume the exclusive defense and control of any matter subject to this indemnification, and in such case, we agree to cooperate with CIS in its defense of such claim. We further agree that all CIS Parties are third-party beneficiaries of our undertakings in these Agreed Terms of Use.

Special rules.

CIS has created and will from time to time create special rules for its members and for other persons and organizations with which CIS has a written contractual relationship. Those special rules will override and supersede these Agreed Terms of Use with respect to the users who are covered by the special rules. CIS hereby grants each CIS Security Consulting or Software Vendor Member and each CIS Organizational User Member, but only so long as such Member remains in good standing with CIS and complies with all of the terms of these Agreed Terms of Use, the right to distribute the Products and Recommendations within such Member's own organization, whether by manual or electronic means. Each such Member acknowledges and agrees that the foregoing grant is subject to the terms of such Member's membership arrangement with CIS and may, therefore, be modified or terminated by CIS at any time.

Choice of law; jurisdiction; venue.

We acknowledge and agree that these Agreed Terms of Use will be governed by and construed in accordance with the laws of the State of Maryland, that any action at law or in equity arising out of or relating to these Agreed Terms of Use shall be filed only in the courts located in the State of Maryland, that we hereby consent and submit to the personal jurisdiction of such courts for the purposes of litigating any such action. If any of these Agreed Terms of Use shall be determined to be unlawful, void, or for any reason unenforceable, then such terms shall be deemed severable and shall not affect the validity and enforceability of any remaining provisions. We acknowledge and agree that we have read these Agreed Terms of Use in their entirety, understand them and agree to be bound by them in all respects.

Table of Contents

Overview

This document, *Security Configuration Benchmark for Microsoft IIS 7*, provides prescriptive guidance for establishing a secure configuration posture for Microsoft IIS version 7.0 running on Microsoft Windows Server 2008. This guide was tested against Microsoft IIS 7 on Microsoft Windows Server 2008 RTM. To obtain the latest version of this guide, please visit http://cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

Consensus Guidance

This benchmark was created using a consensus review process comprised of volunteer and contract subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been released to the public Internet. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in to the CIS benchmark. If you are interested in participating in the consensus review process, please send us a note to feedback@cisecurity.org.

Intended Audience

This benchmark is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate Microsoft IIS 7 on a Microsoft Windows Server 2008 platform.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Authors
Derek M. Knicker

Contributors
Shailesh Athalye, *Symantec, Inc.*
Vern Perryman, *Hewlett-Packard*
Marco Shaw, *Atlantic Lottery*
Art Stricek, *County of Berks, PA*
John Wenning, *Tripwire, Inc*

Typographic Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<*italic font in brackets*>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Configuration Levels

This section defines the configuration levels that are associated with each benchmark recommendation. Configuration levels represent increasing levels of security assurance.

Level-I Benchmark settings/actions

Level-I Benchmark recommendations are intended to:

- be practical and prudent;
- provide a clear security benefit; and
- do not negatively inhibit the utility of the technology beyond acceptable means

Level-II Benchmark settings/actions

Level-II Benchmark recommendations exhibit one or more of the following characteristics:

- are intended for environments or use cases where security is paramount
- acts as defense in depth measure
- may negatively inhibit the utility or performance of the technology

Scoring Status

This section defines the scoring statuses used within this document. The scoring status indicates whether compliance with the given recommendation is discernable in an automated manner.

Scorable

The platform's compliance with the given recommendation can be determined via automated means.

Not Scorable

The platform's compliance with the given recommendation cannot be determined via automated means.

1. Recommendations

1.1 Basic Configurations

1.1.1 Ensure Web Content is on Non-System Partition (Level 1, Scorable)

Description:
Web resources published through IIS are mapped, via Virtual Directories, to physical locations on disk. It is recommended to map all Virtual Directories to a non-system disk volume.

Rationale:
Isolating web content from system files may reduce the probability of

- web sites/applications exhausting system disk space.
- file IO vulnerability in the web site/application from affecting the confidentiality and/or integrity of system files.

Remediation:

1) Browse to web content in `C:\inetpub\wwwroot\`
2) Copy or cut content onto a dedicated and restricted web folder on a non-system drive such as `D:\webroot\`
3) Change mappings for any applications or Virtual Directories to reflect the new location

   To change the mapping for the application named `app1` which resides under the Default Web Site, open IIS Manager:

1) Expand the server node
2) Expand Sites
3) Expand Default Web Site
4) Click on `app1`
5) In the Actions pane, select Basic Settings
6) In the Physical path text box, put the new location of the application, `D:\wwwroot\app1` in the example above

Audit:
Execute the following command to ensure no virtual directories are mapped to the system drive.

```
%systemroot%\system32\inetsrv\appcmd list vdir
```

Default Value:
The default location for web content is: `%systemdrive%\inetpub\webroot`.

References:
1) http://technet.microsoft.com/en-us/library/cc179961.aspx

1.1.2 Remove Or Rename Well-Known URLs (Level 1, Scorable)

Description:
IIS may install superfluous files, folders and Virtual Directories depending on the options chosen during setup and the features desired. Suck resources may increase the attack surface of IIS. It is recommended that all default content, including well-known URLs be removed or renamed.

Rationale:
URLs that are installed by default are well-known to everyone, including those with malicious intent. Removing unnecessary or extraneous files and folders will reduce the Web server's attack surface.

Remediation:
All default Virtual Directories and the files and folder they point to should be removed or renamed. In the case they can be removed:

1) Remove the `%systemdrive%\inetpub\AdminScripts` folder if it exists
2) Remove the `%systemdrive%\inetpub\scripts\IISSamples` folder if it exists
3) Remove the iissamples Virtual Directory mapping if it exists
4) Restrict access to the iisadmpwd Virtual Directory to Windows Authenticated users if exists or remove the Virtual Directory mapping
5) Remove the IISHelp Virtual Directory mapping if it exists
6) Remove the Printers Virtual Directory mapping if it exists
7) Remove the `%programfiles%\Common Files\System\msadc` folder if it exists

In the case the `IISHelp` virtual directory, for example, were to be renamed to `IISHelpVD`, use the following `appcmd` command:

```
%systemroot%\system32\inetsrv\appcmd set vdir "Default Web Site/IISHelp" -
path:/IISHelpVD
```

Audit:
1) Ensure nothing exists at `%systemdrive%\inetpub\AdminScripts`
2) Ensure nothing exists at `%systemdrive%\inetpub\scripts\IISSamples`
3) Ensure nothing exists at `http://localhost/iissamples`
4) Ensure nothing exists at `http://localhost/iisadmpwd`
5) Ensure nothing exists at `http://localhost/IISHelp`
6) Ensure nothing exists at `http://localhost/Printers`
7) Exists at `%programfiles%\Common Files\System\msadc`

Default Value:
These files and folders will exist depending on the features installed along with IIS,

References:
1) http://windows.stanford.edu/docs/IISseccheckList.htm

1.1.3 Require Host Headers on all Sites (Level 2, Scorable)

Description:
Host headers provide the ability to host multiple websites on the same IP address and port. It is recommended that host headers be configured for all sites.

Rationale:
Requiring a Host header for all sites may reduce the probability of

- DNS rebinding attacks successfully compromising or abusing site data or functionality. [2]
- IP-based scans successfully identifying or interacting with a target application hosted on IIS.

Remediation:
Obtain a listing of all sites by using the following `appcmd` command:

```
%systemroot%\system32\inetsrv\appcmd list sites
```

Perform the following in IIS Manager to configure host headers for the Default Web Site:

1) Open IIS Manager
2) In the Connections pane expand the Sites node and select Default Web Site
3) In the Actions pane click Bindings
4) In the Site Bindings dialog box, select the binding for which host headers are going to be configured, Port 80 in this example
5) Click Edit
6) Under host name, enter the sites FQDN, such as *`www.yoursite.com`*
7) Click OK, then Close

Note: Requiring a Host header may impair site functionality for HTTP/1.0 clients.

Audit:
Execute the following command to identify sites that are not configured to require host headers:

```
%systemroot%\system32\inetsrv\appcmd list sites
```

All sites will be listed as such:
```
SITE "Default Web Site" (id:1,bindings:http/*:80:test.com,state:Started)
SITE "badsite" (id:3,bindings:http/*:80:,state:Started)
```

For all non-SSL sites, ensure that the *IP:port:host* binding triplet contains a host name. In the example above, the first site is configured as recommended given the `Default Web Site` has a host header of `test.com`. `badsite`, however, does not have a host header configured – it shows `*:80:` which means all IPs over port 80, with no host header.

Default Value:

Host headers are not required or set up by default.

References:

1) http://technet.microsoft.com/en-us/library/cc753195%28WS.10%29.aspx
2) http://crypto.stanford.edu/dns/dns-rebinding.pdf
3) http://www.sslshopper.com/article-ssl-host-headers-in-iis-7.html
4) http://blogs.iis.net/thomad/archive/2008/01/25/ssl-certificates-on-sites-with-host-headers.aspx

1.1.4 Disable Directory Browsing (Level 1, Scorable)

Description:
Directory browsing allows the contents of a directory to be displayed upon request from a web client. If directory browsing is enabled for a directory in Internet Information Services, users receive a page that lists the contents of the directory when the following two conditions are met:

1) No specific file is requested in the URL
2) The Default Documents feature is disabled in IIS, or if it is enabled, IIS is unable to locate a file in the directory that matches a name specified in the IIS default document list

It is recommended that directory browsing be disabled.

Rationale:
Ensuring that directory browsing is disabled may reduce the probability of disclosing sensitive content that is inadvertently accessible via IIS.

Remediation Steps:
Directory Browsing can be set by using the UI, running `appcmd.exe` commands, by editing configuration files directly, or by writing WMI scripts. To disable directory browsing at the server level using an `appcmd.exe` command:

```
%systemroot%\system32\inetsrv\appcmd set config /section:directoryBrowse
/enabled:false
```

Audit Steps:
Perform the following to verify that Directory Browsing has been disabled at the server level:

```
%systemroot%\system32\inetsrv\appcmd list config /section:directoryBrowse
/enabled:false
```

If the server is configured as recommended, the following will be displayed:

```
<system.webServer>
  <directoryBrowse enabled="false" />
</system.webServer>
```

Value:

In IIS 7.0, Directory browsing is disabled by default.

References:

1) http://technet.microsoft.com/en-us/library/cc725840%28WS.10%29.aspx
2) http://technet.microsoft.com/en-us/library/cc731109%28WS.10%29.aspx

1.1.5 Set Default Application Pool Identity To Least Privilege Principal (Level 1, Scorable)

Description:
Application Pool Identities are the actual users/authorities that will run the worker process – `w3wp.exe`. Assigning the correct principal will help ensure that applications can function properly, while not giving overly permissive permissions on the system. These identities can further be used in ACLs to protect system content.

IIS 7.0 has additional built-in least privilege identities intended for use by Application Pools. It is recommended that the default Application Pool Identity be changed to a least privilege principle other than Network Service. It is recommended that all application pool identities be assigned a unique least privilege principal.

Rationale:
Setting Application Pools to use least privilege identities reduces the potential harm the identity could cause if the application becomes compromised.

Remediation:
The default Application Pool identity may be set for an application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, directly editing the configuration files, or by writing WMI scripts. Perform the following to change the default identity to the built-in `ApplicationPoolIdentity` in the IIS Manager GUI:

1) Open the IIS Manager GUI
2) In the connections pane, expand the server node and click Application Pools
3) On the Application Pools page, select the `DefaultAppPool`, and then click Advanced Settings in the Actions pane
4) For the Identity property, click the `'...'` button to open the Application Pool Identity dialog box
5) Select the Built-in account option choose `ApplicationPoolIdentity` from the list
6) Restart IIS

To change the `DefaultAppPool` identity to the built-in `ApplicationPoolIdentity` using `AppCmd.exe`, run the following from a command prompt:

```
%systemroot%\system32\inetsrv\appcmd set config /section:applicationPools
/[name='DefaultAppPool'].processModel.identityType:ApplicationPoolIdentity
```

Note: If using a custom defined Windows user such as a dedicated service account, that user will need to be a member of the `IIS_IUSRS` group. The `IIS_IUSRS` group has access to all the necessary file and system resources so that an account, when added to this group, can seamlessly act as an application pool identity.

Audit:
Execute the following command to determine if the `DefaultAppPool` identity has been changed to `ApplicationPoolIdentity`:

```
%systemroot%\system32\inetsrv\appcmd list config /section:applicationPools
```

Default Value:
The `DefaultAppPool` in is configured to use the Network Service account.

References:

1) http://technet.microsoft.com/en-us/library/cc771170%28WS.10%29.aspx
2) http://learn.iis.net/page.aspx/140/understanding-built-in-user-and-group-accounts-in-iis-7/

1.1.6 Ensure Application Pools Run Under Unique Identities (Level 1, Scorable)

Application Pool Identities are the actual users/authorities that will run the worker process – `w3wp.exe`. Assigning the correct user authority will help ensure that applications can function properly, while not giving overly permissive permissions on the system. These identities can further be used in ACLs to protect system content. It is recommended that each Application Pool run under a unique identity.

Rationale:
Setting Application Pools to use unique identities reduces the potential harm the identity could cause if the application becomes compromised.

Remediation:

1) Open IIS Manager
2) Open the Application Pools node underneath the machine node; select Application Pool(s) to be changed
3) Right click Application Pool and select Advanced Settings…
4) Select the Identity list item under the Process Model section
5) Once selected, click the ellipsis in the right hand column
6) Select `ApplicationPoolIdentity` from the Built-in account drop down box

Additionally, `ApplicationPoolIdentity` can be made the default for all Application Pools by using the Set Application Pool Defaults action on the Application Pools node.

Audit:
The following `appcmd` command will give a listing of all applications configured and which application pool is serving them:

```
%systemroot%\system32\inetsrv\appcmd list app
```

The output of this command will be similar to the following:

```
APP "Default Web Site/" (applicationPool:DefaultAppPool)
```

1) Ensure a unique application pool for each app listed, which can be found in the parentheses

Default Value:
By default, all Sites created will use the Default App Pool (`DefaultAppPool`).

References:
1) http://technet.microsoft.com/en-us/library/cc753449%28WS.10%29.aspx
2) http://blogs.iis.net/tomwoolums/archive/2008/12/17/iis-7-0-application-pools.aspx
3) http://learn.iis.net/page.aspx/624/application-pool-identities/

1.1.7 Ensure Unique Application Pools for Sites (Level 1, Scorable)

Description:
IIS 7.0 introduced a new security feature called Application Pool Identities that allows Application Pools to be run under unique accounts without the need to create and manage local or domain accounts. It is recommended that all Sites run under unique, dedicated Application Pools.

Rationale:
By setting sites to run under unique Application Pools, resource-intensive applications can be assigned to their own application pools which could improve server and application performance. In addition, it can help maintain application availability: if an application in one pool fails, applications in other pools are not affected. Last, isolating applications helps mitigate the potential risk of one application being allowed access to the resources of another application.

Remediation:

1) Open IIS Manager
2) Open the Sites node underneath the machine node
3) Select the Site to be changed
4) In the Actions pane, select Basic Settings
5) Click the Select... box next to the Application Pool text box
6) Select the desired Application Pool
7) Once selected, click OK

Audit:
The following `appcmd` command will give a listing of all applications configured, which site they are in, which application pool is serving them and which application pool identity it's running under:

```
%systemroot%\system32\inetsrv\appcmd list app
```

The output of this command will be similar to the following:

```
APP "Default Web Site/" (applicationPool:DefaultAppPool)
```

1) Ensure a unique application pool for each site listed

Default Value:
By default, all Sites created will use the Default Application Pool (`DefaultAppPool`).

References:
1) http://technet.microsoft.com/en-us/library/cc753449%28WS.10%29.aspx
2) http://blogs.iis.net/tomwoolums/archive/2008/12/17/iis-7-0-application-pools.aspx
3) http://learn.iis.net/page.aspx/624/application-pool-identities/

1.1.8 Configure Anonymous User Identity To Use Application Pool Identity (Level 1, Scorable)

Description:
To achieve isolation in IIS 7.0, application pools can be run as separate identities. IIS can be configured to automatically use the application pool identity if no anonymous user account is configured for a Web site. This can greatly reduce the number of accounts needed for Web sites and make management of the accounts easier. It is recommended the Application Pool Identity be set as the Anonymous User Identity.

Rationale:
Configuring the anonymous user identity to use the application pool identity will help ensure site isolation – provided sites are set to use the application pool identity. Since a unique principal will run each application pool, it will ensure the identity is least privilege. Additionally, it will simplify Site management.

Remediation:
To configure the Web server to use the application pool identity as the anonymous identity:
1) Open a command prompt and run the following:

```
%windir%\system32\inetsrv\appcmd set config -section:anonymousAuthentication
/username:"" --password
```

Audit:

1) Open IIS Manager
2) Open the Sites node underneath the machine node
3) In the Actions pane, click Set Web Site Defaults
4) The Application Pool line under General should now read DefaultAppPool

Default Value:
The default identity for the anonymous user is the IUSR virtual account.

References:
1) http://learn.iis.net/page.aspx/202/application-pool-identity-as-anonymous-user/
2) http://learn.iis.net/page.aspx/624/application-pool-identities/

1.2 Configure Authentication

1.2.1 Configure Global Authorization Rule to restrict access (Level 1, Not Scorable)

Description:
IIS 7 introduced URL Authorization, which allows the addition of Authorization rules to the actual URL, instead of the underlying file system resource, as a way to protect it. Authorization rules can be configured at the server, web site, folder (including Virtual Directories), or file level. The native URL Authorization module applies to all requests, whether they are .NET managed or other types of files (e.g. static files or ASP files). It is recommended that URL Authorization be configured to only grant access to the necessary security principals.

Rationale:
Configuring a global Authorization rule that restricts access ensures inheritance of the settings down through the hierarchy of web directories; if that content is copied elsewhere, the authorization rules flow with it. This will ensure access to current and future content is only granted to the appropriate principals, mitigating risk of accidental or unauthorized access.

Remediation:
To configure URL Authorization at the server level using IIS Manager:

1) Connect to Internet Information Services(IIS) Manager
2) Select the server
3) Select Authorization Rules
4) Remove the "Allow All Users" rule
5) Click Add Allow Rule…
6) Allow access to the user(s), user groups, or roles that are authorized across all of the web sites and applications (e.g. the Administrators group)

Audit:
At the web site or application level, verify that the authorization rule configured has been applied:

1) Connect to Internet Information Services(IIS) Manager
2) Select the site or application where Authorization was configured
3) Select Authorization Rules and verify the configured rules were added

To verify an authorization rule specifying no access to all users except the Administrators group, browse to and open the `web.config` file for the configured site/application/content:

```
<configuration>
   <system.webServer>
      <security>
          <authorization>
               <remove users="*" roles="" verbs="" />
               <add accessType="Allow" roles="administrators" />
```

```
            </authorization>
        </security>
    </system.webServer>
</configuration>
```

Default Value:
The server-level setting is to allow all users access.

References:

1) http://learn.iis.net/page.aspx/142/understanding-iis-70-url-authorization/
2) http://learn.iis.net/page.aspx/110/changes-between-iis6-and-iis7-security/

1.2.2 Ensure Access to Sensitive Site Features Is Restricted To Authenticated Principals Only (Level 1, Scorable)

Description:
IIS 7 supports both challenge-based and login redirection-based authentication methods. Challenge-based authentication methods, such as Integrated Windows Authentication, require a client to respond correctly to a server-initiated challenge. A login redirection-based authentication method such as Forms Authentication relies on redirection to a login page to determine the identity of the principal. Challenge-based authentication and login redirection-based authentication methods cannot be used in conjunction with one another.

Public servers/sites are typically configured to use Anonymous Authentication. This method typically works, provided the content or services is intended for use by the public. When sites, applications, or specific content containers are not intended for anonymous public use, an appropriate authentication mechanism should be utilized. Authentication will help confirm the identity of clients who request access to sites, application, and content. IIS 7.0 provides the following authentication modules by default:

1) Anonymous Authentication – allows anonymous users to access sites, applications, and/or content
2) Integrated Windows Authentication – authenticates users using the NTLM or Kerberos protocols; Kerberos v5 requires a connection to Active Directory
3) ASP.NET Impersonation – allows ASP.NET applications to run under a security context different from the default security context for an application
4) Forms Authentication - enables a user to login to the configured space with a valid user name and password which is then validated against a database or other credentials store
5) Basic authentication – requires a valid user name and password to access content
6) Client Certificate Mapping Authentication – allows automatic authentication of users who log on with client certificates that have been configured; requires SSL
7) Digest Authentication – uses Windows domain controller to authenticate users who request access

Note: none of the challenge-based authentication modules can be used at the same time Forms Authentication is enabled for certain applications/content. Forms Authentication does not rely on IIS authentication, so anonymous access for the ASP.NET application can be configured if Forms Authentication will be used.

It is recommended that sites containing sensitive information, confidential data, or non-public web services be configured with a credentials-based authentication mechanism.

Rationale:
Configuring authentication will help mitigate the risk of unauthorized users accessing data and/or services, and in cases reduce the potential harm that can be done to a system.

Remediation:
Enabling authentication can be performed by using the user interface (UI), running `Appcmd.exe` commands in a command-line window, editing configuration files directly, or by writing WMI scripts. To verify an authentication mechanism is in place for sensitive content using the IIS Manager GUI:

1) Open IIS Manager and navigate to level with sensitive content
2) In Features View, double-click Authentication
3) On the Authentication page, make sure an authentication module is enabled, while anonymous authentication is enabled (Forms Authentication can have anonymous as well)
4) If necessary, select the desired authentication module, then in the Actions pane, click Enable

Note: When configuring an authentication module for the first time, each mechanism must be further configured before use.

Audit:
To verify that the authentication module is enabled for a specific site, application, or content, browse to and open the `web.config` file pertaining to the content. Verify the configuration file now has a mode defined within the `<authentication>` tags. The example below shows that Forms Authentication is configured, cookies will always be used, and SSL is required:

```
<system.web>
    <authentication>
         <forms cookieless="UseCookies" requireSSL="true" />
    </authentication>
</system.web>
```

Default Value:
The default installation of IIS 7.0 supports Anonymous and Integrated Windows Authentication by default.

References:
1) http://learn.iis.net/page.aspx/377/using-aspnet-forms-authentication/rev/1
2) http://learn.iis.net/page.aspx/244/how-to-take-advantage-of-the-iis7-integrated-pipeline/
3) http://technet.microsoft.com/en-us/library/cc733010%28WS.10%29.aspx

4) http://msdn.microsoft.com/en-us/library/aa480476.aspx

1.2.3 Require SSL in Forms Authentication (Level 1, Scorable)

Description:
Forms-based authentication can pass credentials across the network in clear text. It is therefore imperative that the traffic between client and server be encrypted using SSL, especially in cases where the site is publicly accessible. It is recommended that communications with any portion of a site using Forms Authentication be encrypted using SSL.

Rationale:
Requiring SSL for Forms Authentication will protect the confidentiality of credentials during the login process, helping mitigate the risk of stolen user information.

Remediation:

1) Open IIS Manager and navigate to the appropriate tier
2) In Features View, double-click Authentication
3) On the Authentication page, select Forms Authentication
4) In the Actions pane, click Edit
5) Check the Requires SSL checkbox in the cookie settings section, click ok

Audit:
To verify that the authentication module is enabled for a specific site, application, or content, browse to and open the `web.config` file for the level in which forms authentication was enabled. Verify the tag `<forms requireSSL="true" />`:

```
<system.web>
    <authentication>
         <forms requireSSL="true" />
    </authentication>
</system.web>
```

Default Value:
SSL is not required when forms authentication is enabled.

References:
1) http://technet.microsoft.com/en-us/library/cc771077(WS.10).aspx

1.2.4 Configure Forms Authentication to Use Cookies (Level 2, Scorable)

Description:
Forms Authentication can be configured to maintain the site visitor's session identifier in either a URI or cookie. It is recommended that Forms Authentication be set to use cookies.

Rationale:
Using cookies to manage session state may help mitigate the risk of session hi-jacking attempts by preventing ASP.NET from having to move session information to the URL.

Moving session information identifiers into the URL may cause session IDs to show up in proxy logs, browsing history, and be accessible to client scripting via `document.location`.

Remediation:

1) Open IIS Manager and navigate to the level where Forms Authentication is enabled
2) In Features View, double-click Authentication
3) On the Authentication page, select Forms Authentication
4) In the Actions pane, click Edit
5) In the Cookie settings section, select Use cookies from the Mode dropdown

Audit:
Locate and open the `web.config` for the configured application. Verify the presence of `<forms cookieless="UseCookies" />`.

```
<system.web>
        <authentication>
            <forms cookieless="UseCookies" requireSSL="true" timeout="30" />
        </authentication>
</system.web>
```

Default Value:
The default setting for Cookie Mode is Auto Detect which will only use cookies if the device profile supports cookies.

References:

1) http://technet.microsoft.com/en-us/library/cc732830%28WS.10%29.aspx

1.2.5 *Configure Cookie Protection Mode for Forms Authentication (Level 1, Scorable)*

Description:
The cookie protection mode defines the protection Forms Authentication cookies will be given within a configured application. The four cookie protection modes that can be defined are:

1) Encryption and validation - Specifies that the application use both data validation and encryption to help protect the cookie. This option uses the configured data validation algorithm (based on the machine key) and triple-DES (3DES) for encryption, if available and if the key is long enough (48 bytes or more)
2) None - Specifies that both encryption and validation are disabled for sites that are using cookies only for personalization and have weaker security requirements
3) Encryption - Specifies that the cookie is encrypted by using Triple-DES or DES, but data validation is not performed on the cookie; cookies used in this manner might be subject to plain text attacks
4) Validation - Specifies that a validation scheme verifies that the contents of an encrypted cookie have not been changed in transit

It is recommended that cookie protection mode always encrypt and validate Forms Authentication cookies.

Rationale:
By encrypting and validating the cookie, the confidentiality and integrity of data within the cookie is assured. This helps mitigate the risk of attacks such as session hijacking and impersonation.

Remediation:
Cookie protection mode can be configured by using the user interface (UI), by running `Appcmd.exe` commands in a command-line window, by editing configuration files directly, or by writing WMI scripts. Using IIS Manager:

1) Open IIS Manager and navigate to the level where Forms Authentication is enabled
2) In Features View, double-click Authentication
3) On the Authentication page, select Forms Authentication
4) In the Actions pane, click Edit
5) In the Cookie settings section, verify the dropdown for Protection mode is set for Encryption and validation

Audit:
Locate and open the `web.config` for the configured application. Verify the presence of `<forms protection="All" />`.

```
<system.web>
        <authentication>
            <forms cookieless="UseCookies" protection="All" />
        </authentication>
</system.web>
```

Note: the `protection="All"` property will only show up if cookie protection mode was set to something different, and then changed to Encryption and Validation. To truly verify the `protection="All"` property in the `web.config`, the protection mode can be changed, and then changed back. Conversely, the `protection="All"` line can be added to the `web.config` manually.

Default Value:
When cookies are used for Forms Authentication, the default cookie protection mode is `All`, meaning the application encrypts and validates the cookie.

References:
1) http://technet.microsoft.com/en-us/library/cc731804%28WS.10%29.aspx

1.2.6 *Ensure passwordFormat Credentials Element Not Set To Clear (Level 1, Scorable)*

Description:
The `<credentials>` element of the `<authentication>` element allows optional definitions of name and password for IIS Manager User accounts within the configuration file. IIS Manager Users can use the administration interface to connect to sites and applications in which they've been granted authorization. Note that the `<credentials>` element only applies when the default provider, `ConfigurationAuthenticationProvider`, is configured

as the authentication provider. It is recommended that `passwordFormat` be set to a value other than `Clear`, such as `SHA1` or `MD5`.

Rationale:
Authentication credentials should always be protected to reduce the risk of stolen authentication credentials.

Remediation:
Authentication mode is configurable at the `machine.config`, root-level `web.config`, or application-level `web.config`.

1) Locate and open the configuration file where the credentials are stored
2) Find the `<credentials>` element
3) If present, ensure `passwordFormat` is not set to `Clear`
4) Change `passwordFormat` to `SHA1` or `MD5`

```
Note: The clear text passwords will need to be replaced with the appropriate
hashed version.
```

Audit:
Locate and open the configuration file for the configured application. Verify the `passwordFormat` is not set to `Clear`:

```
<configuration>
   <system.web>
      <authentication mode="Forms">
         <forms name="SampleApp" loginUrl="/login.aspx">
            <credentials passwordFormat = "SHA1">
               <user
                  name="UserName1"
                  password="SHA1EncryptedPassword1"/>
               <user
                  name="UserName2"
                  password="SHA1EncryptedPassword2"/>
            </credentials>
         </forms>
      </authentication>
    </system.web>
</configuration>
```

Default Value:
The default passwordFormat method is `SHA1`.

References:

1) http://msdn.microsoft.com/en-us/library/e01fc50a.aspx
2) http://www.iis.net/ConfigReference/system.webServer/management/authentication/credentials
3) http://msdn.microsoft.com/en-us/library/bb422401%28VS.90%29.aspx

1.2.7 Lock Down Encryption Providers (Level 2, Scorable)

Description:
By default, whenever a property is encrypted, IIS 7.0 uses the `defaultProvider` for encryption defined in `machine.config`. The IIS 7.0 local system process (WAS) runs under the context of `LOCALSYSTEM` and needs access to the application pool passwords. However, by default the `IIS_IUSRS` security group is granted read access. It is recommended that the `IIS_IUSRS` group have access to the `iisWasKey` revoked.

Rationale:
The `iisWasKey` is intended for access only by Administrators and `SYSTEM`. Since the `IIS_IUSRS` group is granted read access, an attacker compromising an application set to use a principal in the `IIS_ISURS` group could potentially gain access to the encryption key(s). Revoking this unnecessary privilege will reduce attack surface and help maintain confidentiality and system/application integrity.

Remediation:
Removing access to the `iisWasKey` can be done by using an `AppCmd.exe` command. The syntax is as follows, and is dependent on the version of .NET being used:

```
%systemroot%\Microsoft.NET\Framework\<framework_version>\aspnet_regiis.exe -
pr iisWasKey IIS_IUSRS
```

To remove read access to the `IIS_IUSRS` security group on a system using .NET Framework v2.0:

1) Open an elevated command prompt
2) Run the following aspnet_regiis.exe command:

```
%systemroot%\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe -pr
iisWasKey IIS_IUSRS
```

3) If running a 64-bit system, also run the following:

```
%systemroot%\Microsoft.NET\Framework64\v2.0.50727\aspnet_regiis.exe -pr
iisWasKey IIS_IUSRS
```

Note: A unique version of `aspnet_regiis.exe` is included with each version of the .NET Framework. Since each version of the tool applies only to its associated version of the .NET Framework, be sure to use the appropriate version of the tool.

Audit:
1) To verify the permissions have been removed, obtain the machine GUID located in: `%ALLUSERSPROFILE%\Microsoft\Crypto\RSA\MachineKeys\`
2) Next, open a command prompt and run the following:

```
icacls command:Icacls
ALLUSERSPROFILE%\Microsoft\Crypto\RSA\MachineKeys\<MachineGUID>
```

3) Ensure that `BUILTIN\IIS_IUSR(R)` has been removed.

Default Value:
The `IIS_IUSRS` account has read access to the `iisWasKey` encryption provider.

References:

1) http://learn.iis.net/page.aspx/141/using-encryption-to-protect-passwords/

1.3 ASP.NET Configuration Recommendations

1.3.1 Set Deployment Method to Retail (Level 1, Scorable)

Description:
The `<deployment retail>` switch is intended for use by production IIS 7.0 servers. This switch is used to help applications run with the best possible performance and least possible security information leakages by disabling the application's ability to generate trace output on a page, disabling the ability to display detailed error messages to end users, and disabling the debug switch. Often times, switches and options that are developer-focused, such as failed request tracing and debugging, are enabled during active development. It is recommended that the deployment method on any production server be set to `retail`.

Rationale:
Utilizing the switch specifically intended for production IIS servers will eliminate the risk of vital application and system information leakages that would otherwise occur if tracing or debug or were to be left enabled, or `customErrors` were to be left off.

Remediation:

1) Open the machine.config file located in:
`%windir%\Microsoft.NET\Framework\<framework_version>\CONFIG`
2) Add the `line <deployment retail="true" />` within the `<system.web>` section
3) If systems are 64-bit, do the same for the machine.config located:
`%windir%\Microsoft.NET\Framework64\<framework_version>\CONFIG`

Audit:
After the next time IIS is restarted, open the `machine.config` file and verify that `<deployment retail="true" />` remains set to true.

```
<system.web>
    <deployment retail="true" />
</system.web>
```

Default Value:
The `<deployment retail>` tag is not included in the `machine.config` by default.

References:

1) http://msdn.microsoft.com/en-US/library/ms228298%28VS.80%29.aspx

1.3.2 Turn Debug Off (Level 2, Scorable)

Description:
Developers often enable the debug mode during active ASP.NET development so that they do not have to continually clear their browsers cache every time they make a change to a resource handler. The problem would arise from this being left "on" or set to "true." Compilation debug output is displayed to the end user, allowing malicious persons to obtain detailed information about applications.

This is a defense in depth recommendation due to the `<deployment retail="true" />` in the `machine.config` configuration file overriding any debug settings. It is recommended that debugging still be turned off.

Rationale:
Setting compilation debug to `false` ensures that detailed error information does not inadvertently display during live application usage, mitigating the risk of application information leakage falling into unscrupulous hands.

Remediation:
To use the UI to make this change:

1) Open IIS Manager and navigate desired server, site, or application
2) In Features View, double-click .NET Compilation
3) On the .NET Compilation page, in the Behavior section, ensure the Debug field is set to False
4) When finished, click Apply in the Actions pane

Audit:
Browse to and open the `web.config` file pertaining to the server or specific application that has been configured. Locate the `<compilation debug>` switch and verify it is set to `false`.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.web>
      <compilation debug="false" />
  </system.web>
</configuration>
```

Note: The `<compilation debug>` switch will not be present in the `web.config` file unless it has been added manually, or has previously been configured using the IIS Manager GUI.

Default Value:
The compilation of debug binaries is not enabled by default.

References:
1) http://technet.microsoft.com/en-us/library/cc725812%28WS.10%29.aspx

1.3.3 Ensure Custom Error Messages are not Off (Level 2, Scorable)

Description:
When an ASP.NET application fails and causes an HTTP/1.x 500 Internal Server Error, or a feature configuration (such as Request Filtering) prevents a page from being displayed, an error message will be generated. Administrators can choose whether or not the application should display a friendly message to the client, detailed error message to the client, or detailed error message to localhost only. The `<customErrors>` tag in the `web.config` has three modes:

1) On: Specifies that custom errors are enabled. If no `defaultRedirect` attribute is specified, users see a generic error. The custom errors are shown to the remote clients and to the local host.
2) Off: Specifies that custom errors are disabled. The detailed ASP.NET errors are shown to the remote clients and to the local host.
3) RemoteOnly: Specifies that custom errors are shown only to the remote clients, and that ASP.NET errors are shown to the local host. This is the default value.

This is a defense in depth recommendation due to the `<deployment retail="true" />` in the `machine.config` file overriding any settings for `customErrors` to be turned `Off`. It is recommended that `customErrors` still be turned to `On` or `RemoteOnly`.

Rationale:
`customErrors` can be set to `On` or `RemoteOnly` without leaking detailed application information to the client. Ensuring that `customErrors` is not set to `Off` will help mitigate the risk of malicious persons learning detailed application error and server configuration information.

Remediation:
`customErrors` may be set for a server, site, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, directly editing the configuration files, or by writing WMI scripts. Perform the following to set the `customErrors` mode to `RemoteOnly` or `On` for a web site in the IIS Manager GUI:

1) Open the IIS Manager GUI and navigate to the site to be configured
2) In Features View, find and double-click the Error Pages icon
3) In the Actions Pane, click Edit
4) Select one of the following in the Edit Custom Error Page dialog box:
    a. Insert content from static file into the error response if content is static
    b. Execute a URL on this site if error content is dynamic, such as an .asp file
    c. Respond with a 302 redirect a custom error page if redirecting a client browser to a different URL

Audit:
Find and open the `web.config` file for the application/site and verify that the tag has either `<customErrors mode="RemoteOnly" />` or `<customErrors mode="On" />` defined.

If using Insert content from static file into the error response, the `customErrors` message/page itself can be viewed by opening a browser and typing the path of the custom error page in the File path text box. If using either the Execute a URL on this site or Respond with a 302 redirect path, type the URL of the custom error page into a browser.

Default Value:
The default value is `<customErrors mode= "RemoteOnly" />`.

References:

1) http://technet.microsoft.com/en-us/library/cc771076(28WS.10)29.aspx
2) http://technet.microsoft.com/en-us/library/cc753103%28WS.10%29.aspx

1.3.4 Ensure Failed Request Tracing is not Enabled (Level 2, Scorable)

Description:
The `trace` element configures the ASP.NET code tracing service that controls how trace results are gathered, stored, and displayed. When tracing is enabled, each page request generates trace messages that can be appended to the page output or stored in an application trace log. In order to use tracing, it must be installed as a role service under the Health and Diagnostics section of the Web Server role.

This is a defense in depth recommendation due to the `<deployment retail="true" />` in the `machine.config` file overriding any settings for Failed Request Tracing to be left on. It is recommended that Failed Request Tracing still be turned off.

Rationale:
In an active Web Site, tracing should not be enabled because it can display sensitive configuration and detailed stack trace information to anyone who views the pages in the site. If necessary, the `localOnly` attribute can be set to true to have trace information displayed only for localhost requests. Ensuring that Failed Request Tracing is not on will help mitigate the risk of malicious persons learning detailed stack trace information.

Remediation:
To disable tracing for the Default Web Site using `AppCmd.exe`, run the following command at an elevated prompt:

```
%windir%\system32\inetsrv\appcmd configure trace "Default Web Site"
/disablesite
```

Note that tracing is configurable at numerous levels:

1) `Machine.config`
2) Root-level `web.config`
3) Application-level `web.config`
4) Virtual or physical directory–level `web.config`

Audit:
Verify Failed Request Tracing is turned off by using the IIS Manager GUI:

1) On the taskbar, click Start, point to Administrative Tools, and then click Internet Information Services (IIS) Manager
2) In the Connections pane, select the server connection, site, application, or directory on which failed request tracing will be configured
3) In the Actions pane, click Failed Request Tracing...
4) In the Edit Web Site Failed Request Tracing Settings dialog box, verify that the Enable check box is not checked

Default Value:
Failed Request Tracing is not enabled by default.

References:
1) http://www.iis.net/ConfigReference/system.webServer/tracing
2) http://technet.microsoft.com/en-us/library/cc730944%28WS.10%29.aspx

1.3.5 *Configure Use Cookies Mode for Session State (Level 2, Scorable)*

Description:
A session cookie associates session information with client information for that session, which can be the duration of a user's connection to a site. The cookie is passed in an HTTP header together with all requests between the client and server.

Session information can also be stored in the URL. However, storing session information in this manner has security implications that can open attack vectors such as session hijacking. An effective method used to prevent session hijacking attacks is to force web applications to use cookies to store the session token. This is accomplished by setting the `cookieless` attribute of the `sessionState` node to `UseCookies` or `False` which will in turn keep session state data out of URI. It is recommended that session state be configured to `UseCookies`.

Rationale:
Cookies that have been properly configured help mitigate the risk of attacks such as session hi-jacking attempts by preventing ASP.NET from having to move session information to the URL; moving session information in URI causes session IDs to show up in proxy logs, and is accessible to client scripting via `document.location`.

Remediation:
`SessionState` can be set to `UseCookies` by using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, directly editing the configuration files, or by writing WMI scripts. Perform the following to set the `cookieless` attribute of the `sessionState` node to `UseCookies` in the IIS Manager GUI:

1) Open the IIS Manager GUI and navigate desired server, site, or application
2) In Features View, find and double-click the Session State icon
3) In the Cookie Settings section, choose Use Cookies from the Mode dropdown
4) In the Actions Pane, click Apply

To use AppCmd.exe to configure sessionState at the server level, the command would look like this:

```
%windir%\system32\inetsrv\appcmd set config /commit:WEBROOT
/section:sessionState /cookieless:UseCookies /cookieName:ASP.NET_SessionID
/timeout:20
```

Note: When Appcmd.exe is used to configure the <sessionState> element at the global level in IIS 7.0, the /commit:WEBROOT switch must be included so that configuration changes are made to the root web.config file instead of ApplicationHost.config.

Audit:
Find and open the web.config file for the application/site and verify that the sessionState tag is set to use cookies:

```
<system.web>
            <sessionState cookieless="UseCookies" />
</system.web>
```

Default Value:
By default, IIS maintains session state data for a managed code application in the worker process where the application runs e.g. In Process.

References:
1) http://technet.microsoft.com/en-us/library/cc770672%28WS.10%29.aspx
2) http://msdn.microsoft.com/en-us/library/h6bb9cz9%28VS.71%29.aspx

1.3.6 Ensure Cookies Are Set With HttpOnly Attribute (Level 2, Scorable)

Description:
The httpOnlyCookies attribute of the httpCookies node determines if IIS will set the HttpOnly flag on HTTP cookies it sets. The HttpOnly flag indicates to the user agent that the cookie must not be accessible by client-side script (i.e document.cookie). It is recommended that the httpOnlyCookies attribute be set to true.

Rationale:
When cookies are set with the HttpOnly flag, they cannot be accessed by client side scripting running in the user's browser. Preventing client-side scripting from accessing cookie content may reduce the probability of a cross site scripting attack materializing into a successful session hijack.

Remediation:
1) Locate and open the application's web.config file
2) Add the <httpCookies httpOnlyCookies="true" /> tag within <system.web>:

```
<configuration>
 <system.web>
   <httpCookies httpOnlyCookies="true" />
 </system.web>
</configuration>
```

Setting the value of the `httpOnlyCookies` attribute of the `httpCookies` element to `true` will add the `HttpOnly` flag to all the cookies set by the application.

Note: This attribute, `HttpOnly`, is currently only recognized by modern versions of Internet Explorer; older versions will either treat them as normal cookies or simply ignore them altogether.

Audit:
After the next time IIS is restarted, browse to and open the `web.config` for the application in which `httpOnly` cookies have been turned on. Confirm the `httpOnlyCookies` attribute is set to true: `<httpCookies httpOnlyCookies="true" />.`

Default Value:
ASP.NET 2.0 does not force cookies to `httpOnly`.

References:
1) http://authors.aspalliance.com/aspxtreme/aspnet/syntax/httpCookies.aspx
2) http://www.asp101.com/tips/index.asp?id=160
3) https://tools.ietf.org/wg/httpstate/charters

1.3.7 Configure MachineKey Validation Encryption (Level 1, Not Scorable)

Description:
The `machineKey` element of the ASP.NET `web.config` specifies the algorithm and keys that ASP.NET will use for encryption. The Machine Key feature can be managed to specify hashing and encryption settings for application services such as view state, Forms authentication, membership and roles, and anonymous identification.

The following encryption methods are available:
1) Advanced Encryption Standard (AES) is relatively easy to implement and requires little memory. AES has a key size of 128, 192, or 256 bits. This method uses the same private key to encrypt and decrypt data, whereas a public-key method must use a pair of keys
2) Message Digest 5 (MD5) is used for digital signing of applications. This method produces a 128-bit message digest, which is a compressed form of the original data. MD5 can provide some protection against computer viruses and Trojan Horses
3) Secure Hash Algorithm (SHA1) is considered more secure than MD5 because it produces a 160-bit message digest.
4) Triple Data Encryption Standard (TripleDES) is a minor variation of Data Encryption Standard (DES). It is three times slower than regular DES but can be more secure because it has a key size of 192 bits. If performance is not a primary consideration, consider using TripleDES

It is recommended that SHA1 encryption be configured for use at the global level.

Rationale:
Secure Hash Algorithm (SHA1) is considered more secure than MD5 because it produces a 160-bit message digest. SHA1 encryption should be used whenever possible. Ensuring a strong encryption method can mitigate the risk of data tampering in crucial functional areas such as Forms authentication cookies or view state.

Remediation Steps:
Machine key encryption can be set by using the UI, running `appcmd.exe` commands, by editing configuration files directly, or by writing WMI scripts. To set the Machine Key encryption at the global level using an `appcmd.exe` command:

```
%systemroot%\system32\inetsrv\appcmd set config /commit:WEBROOT
/section:machineKey /validation:SHA1
```

Note: when `Appcmd.exe` is used to configure the `<machineKey>` element at the global level in IIS 7.0, the `/commit:WEBROOT` switch must be included so that configuration changes are made to the root `web.config` file instead of `ApplicationHost.config`.

Audit Steps:
To verify the Machine Key encryption method using IIS Manager:
1) Open IIS Manager and navigate to the level that was configured, the WEBROOT, or server in this case
2) In the features view, double click Machine Key
3) On the Machine Key page, verify that SHA1 is selected in the Encryption method dropdown

Default Value:
The default Machine Key encryption method is SHA1.

References:
1) http://technet.microsoft.com/en-us/library/cc772271%28WS.10%29.aspx
2) http://technet.microsoft.com/en-us/library/cc772287%28WS.10%29.aspx

1.3.8 Configure Global .NET Trust Level (Level 1, Not Scorable)

Description:
An application's trust level determines the permissions that are granted by the ASP.NET code access security (CAS) policy. CAS defines two trust categories: full trust and partial trust. An application that has full trust permissions may access all resource types on a server and perform privileged operations, while applications that run with partial trust have varying levels of operating permissions and access to resources.

The possible values for the Level property of the TrustSection class are:
1) Full: Specifies unrestricted permissions and grants the ASP.NET application permissions to access any resource that is subject to operating system security; all privileged operations are supported
2) High: specifies a high level of code access security which limits the application from doing the following:

    a. Call unmanaged code
    b. Call serviced components
    c. Write to the event log
    d. Access Microsoft Windows Message Queuing queues
    e. Access ODBC, OLD DB, or Oracle data sources
3) Medium: specifies a medium level of code access security, which means that in addition to the restrictions for High, the ASP.NET application cannot do any of the following things:
    a. Access files outside the application directory
    b. Access the registry
4) Low: specifies a low level of code access security, which means that in addition to the restrictions for Medium, the application is prevented from performing any of the following actions:
    a. Write to the file system
    b. Call the `System.Security.CodeAccessPermission.Assert` method to expand permissions to resources
5) Minimal: specifies a minimal level of code access security, which means that the application has only execute permission

It is recommended that the global .NET Trust Level be set to Medium or lower.

Rationale:
The CAS determines the permissions that are granted to the application on the server. Setting a minimal level of trust that is compatible with the applications will limit the potential harm that a compromised application could cause to a system.

Remediation Steps:
Trust level can be set by using the UI, running `appcmd.exe` commands, by editing configuration files directly, or by writing WMI scripts. To set the .Net Trust Level to Medium at the server level using an `appcmd.exe` command:

```
%systemroot%\system32\inetsrv\appcmd set config /commit:WEBROOT
/section:trust /level:Medium
```

Note: when `Appcmd.exe` is used to configure the `<trust>` element at the global level in IIS 7.0, the `/commit:WEBROOT` switch must be included so that configuration changes are made to the root `web.config` file instead of `ApplicationHost.config`.

Audit Steps:
To verify the global .NET Trust Level using IIS Manager:

4) Open IIS Manager and navigate to the level that was configured, the server in this example
5) In the features view, double click .NET Trust Levels
6) On the .NET Trust Levels page, verify that `Medium (web_mediumtrust.config)` is selected in the Trust Level dropdown

Default Value:

By default ASP.NET web applications run under the full trust setting.

References:

1) http://technet.microsoft.com/en-us/library/cc772237(WS.10).aspx
2) http://msdn.microsoft.com/en-us/library/ms691448%28VS.90%29.aspx
3) *Professional IIS 7 by Ken Schaefer, Jeff Cochran, Scott Forsyth, Rob Baugh, Mike Everest, Dennis Glendenning*

1.4 Request Filtering and Restrictions

Request Filtering is a powerful new module implemented in IIS 7.0 which provides a configurable set of rules that enables administrators to allow or reject the types of requests that they determine should be allowed or rejected at the server, web site, or web application levels.

Earlier versions of Internet Information Services provided the tool UrlScan, which was provided as an add-on to enable system administrators to enforce tighter security policies on their web servers. For IIS 7.0, all of the core features of URLScan have been incorporated into the Request Filtering module. Due to the close nature of functionality in these two tools, reference to legacy URLScan settings will be made where applicable.

Note: Request Filtering must be installed as a role service under IIS in order to configure any of its features. Additionally, to edit Request Filtering settings with the IIS Management GUI requires the IIS 7.0 Administration Pack, which is provided as an IIS Extension and available on the Web.

1.4.1 Configure MaxAllowedContentLength Request Filter (Level 2, Not Scorable)

Description:
The `maxAllowedContentLength` Request Filter is the maximum size of the http request, measured in bytes, which can be sent from a client to the server. Configuring this value enables the total request size to be restricted to a configured value. It is recommended that the overall size of requests be restricted to a maximum value appropriate for the server, site, or application.

Rationale:
Setting an appropriate value that has been tested for the `maxAllowedContentLength` filter will lower the impact an abnormally large request would otherwise have on IIS and/or web applications. This helps to ensure availability of web content and services, and may also help mitigate the risk of buffer overflow type attacks in unmanaged components.

Remediation:
The `MaxAllowedContentLength` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, click on the server, site, application, or directory to be configured
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the Request Limits section, key the maximum content length in bytes that will allow applications to retain their intended functionality, such as 30000000 (approx. 28.6 MB)

Audit:
Upon exceeding the configured value set for the Request Filter, IIS will throw a Status Code 404.13.

To manually verify the change, locate and open the `web.config` for the web site or application in which the request filter was set. Ensure the value defined for `maxAllowedContentLength` is what was set. The 9.3MB max example would show:

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering>
    <requestLimits
       maxAllowedContentLength="30000000" />
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
When request filtering is installed on a system, the default value is:
`maxAllowedContentLength="30000000"`, which is approximately 28.6MB.

References:
1) http://www.iis.net/ConfigReference/system.webServer/security/requestFiltering/requestLimits
2) http://learn.iis.net/page.aspx/143/use-request-filtering/

1.4.2 Configure maxURL Request Filter (Level 2, Scorable)

Description:
`The maxURL` attribute of the `<requestLimits>` property is the maximum length (in Bytes) in which a requested URL can be (excluding query string) in order for IIS to accept. Configuring this Request Filter enables administrators to restrict the length of the requests that the server will accept. It is recommended that a limit be put on the length of URI.

Rationale:
With a properly configured Request Filter limiting the amount of data accepted in the URL, chances of undesired application behaviors affecting the availability of content and services are reduced.

Remediation:
The `MaxURL` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, click on the connection, site, application, or directory to be configured
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the Request Limits section, key the maximum URL length in bytes that has been tested with web applications

Audit:
IIS will log a 404.14 HTTP status if the requested URL was rejected because it exceeded the length defined in the filter.

To manually verify the change, locate and open the `web.config` for the web site or application in which the request filter was set. Verify the value defined for `maxURL`.

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering>
    <requestLimits
       maxURL="4096" />
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
When Request Filtering is installed on a system, the default value for `maxURL="4096"`.

References:
1) http://www.iis.net/ConfigReference/system.webServer/security/requestFiltering/requestLimits
2) http://learn.iis.net/page.aspx/143/use-request-filtering/

1.4.3 Configure MaxQueryString Request Filter (Level 2, Scorable)

Description:
The `MaxQueryString` Request Filter describes the upper limit on the length of the query string that the configured IIS server will allow for websites or applications. It is recommended that values always be established to limit the amount of data will can be accepted in the query string.

Rationale:

With a properly configured Request Filter limiting the amount of data accepted in the query string, chances of undesired application behaviors such as app pool failures are reduced.

Remediation:
The `MaxQueryString` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, go to the connection, site, application, or directory to be configured
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the Request Limits section, key in a safe upper bound in the Maximum query string (Bytes) textbox

Audit:
If a request is rejected because it exceeds the value set in the `maxQueryString` request filter, a 404.15 HTTP status is logged to the IIS log file.

To manually verify the change, locate and open the `web.config` for the web site or application in which the filter was set. Ensure the value defined for `maxQueryString` is what was configured.

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering>
    <requestLimits
      maxQueryString="2048" />
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
When request filtering is installed on a system, the default value is `maxQueryString="2048".`

References:
1) http://www.iis.net/ConfigReference/system.webServer/security/requestFiltering/requestLimits
2) http://learn.iis.net/page.aspx/143/use-request-filtering/

1.4.4 Do not allow non-ASCII characters in URLs (Level 2, Scorable)

Description:
This feature is used to allow or reject all requests to IIS 7 that contain non-ASCII characters. When using this feature, Request Filtering will deny the request if high-bit characters are

present in the URL. The UrlScan equivalent is `AllowHighBitCharacters`. It is recommended that requests containing non-ASCII characters be rejected, where possible.

Note: Disallowing high-bit ASCII characters in the URL may negatively impact the functionality of sites requiring international language support.

Rationale:
This feature can help defend against canonicalization attacks, reducing the potential attack surface of servers, sites, and/or applications.

Remediation:
The `AllowHighBitCharacters` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, go to the connection, site, application, or directory to be configured
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the General section, uncheck Allow high-bit characters

Audit:
If a request is rejected because it contains a high-bit character, a 404.12 HTTP status is logged to the IIS log file.

To manually verify the change, locate and open the `web.config` for the web site or application in which the request filter was set. Ensure the value defined for the filter is false, as such:

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering
     allowHighBitCharacters="false">
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
When request filtering is installed on a system, the default behavior is to allow high-bit characters in URI.

References:
1) http://learn.iis.net/page.aspx/143/use-request-filtering/
2) http://learn.iis.net/page.aspx/936/urlscan-1-reference/
3) *Professional IIS 7 By Ken Schaefer, Jeff Cochran, Scott Forsyth, Rob Baugh, Mike Everest, Dennis Glendenning*

1.4.5 Ensure Double-Encoded Requests will be rejected (Level 1, Scorable)

Description:
This request filter feature prevents attacks that rely on double-encoded requests and applies if an attacker submits a double-encoded request to IIS. When the double-encoded requests filter is enabled, IIS 7 will go through a two iteration process of normalizing the request. If the first normalization differs from the second, the request is rejected and the error code is logged as a 404.11. The double-encoded requests filter was the `VerifyNormalization` option in UrlScan. It is recommended that double-encoded requests be rejected.

Rationale:
This feature will help prevent attacks that rely on URLs that have been crafted to contain double-encoded request(s).

Remediation:
The `allowDoubleEscaping` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, select the site, application, or directory to be configured
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the General section, uncheck Allow double escaping

Note: If a filename in a URL includes "+" then `allowDoubleEscaping` must be set to `true`.

Audit:
If a request is rejected because it contains a double-encoded request, a 404.11 HTTP status is logged to the IIS log file.

To manually verify the change, locate and open the `web.config` for the web site or application in which the request filter was set. Ensure the value defined for `allowDoubleEscaping` is `false`:

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering
     allowDoubleEscaping="false">
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
When request filtering is installed on a system, the default behavior is to not allow double-encoded requests.

References:

1) http://www.iis.net/ConfigReference/system.webServer/security/requestFiltering/requestLimits
2) http://learn.iis.net/page.aspx/143/use-request-filtering/

1.4.6 Disallow Unlisted File Extensions (Level 1, Scorable)

Description:
The `FileExtensions` Request Filter allows administrators to define specific extensions their web server(s) will allow and disallow. The property `allowUnlisted` will cover all other file extensions not explicitly allowed or denied. Often times, extensions such as `.config`, `.bat`, `.exe`, to name a few, should never be served. The `AllowExtensions` and `DenyExtensions` options are the UrlScan equivalents. It is recommended that all extensions be unallowed at the most global level possible, with only those necessary being added.

Rationale:
Disallowing all but the defined types of file extensions can greatly reduce the attack surface applications and servers.

Remediation:
The `allowUnlisted` Request Filter may be set for a server, website, or application using the IIS Manager GUI, using `AppCmd.exe` commands in a command-line window, and/or directly editing the configuration files. To configure at the server level using the IIS Manager GUI:

1) Open Internet Information Services (IIS) Manager
2) In the Connections pane, select the server
3) In the Home pane, double-click Request Filtering
4) Click Edit Feature Settings... in the Actions pane
5) Under the General section, uncheck Allow unlisted file name extensions

   To set this Request Filter using an `AppCmd.exe` command, run this command at an elevated command prompt:

```
%windir%\system32\inetsrv\appcmd set config /section:requestfiltering
/fileExtensions.allowunlisted:false
```

Audit:
When IIS 7 rejects a request based on a file extensions filter, the error code logged is 404.7.

To manually verify the change, locate and open the `web.config` for the web site or application in which the request filter was set. Ensure `<fileExtensions allowUnlisted="false">`. The following `web.config` will disallow any requests for files that do not have `.asp`, `.aspx`, or `.html` as their extension:

```
<configuration>
 <system.webServer>
  <security>
   <requestFiltering>
    <fileExtensions allowUnlisted="false" >
     <add fileExtension=".asp" allowed="true"/>
     <add fileExtension=".aspx" allowed="true"/>
     <add fileExtension=".html" allowed="true"/>
    </fileExtensions>
   </requestFiltering>
  </security>
 </system.webServer>
</configuration>
```

Default Value:
The default Request Filtering configuration allows all unlisted file extensions to be requested.

References:

1) http://www.iis.net/ConfigReference/system.webServer/security/requestFiltering/requestLimits

1.5 IIS Logging Recommendations

1.5.1 Move Default IIS Web Log Location (Level 1, Scorable)

Description:
IIS will log relatively detailed information every request. These logs are usually the first item looked at in a security response, and can be the most valuable. Malicious users are aware of this, and will often try to remove evidence of their activities. It is therefore recommended that the default location for IIS log files be changed to a restricted, non-system drive.

Rationale:
Moving IIS logging to a restricted, non-system drive will help mitigate the risk of logs being maliciously altered, removed, or lost in the event of system drive failure(s).

Remediation:
Moving the default log location can be easily accomplished using the Logging feature in the IIS Management UI or `AppCmd.exe`. To change to D:\LogFiles, for example:

```
%windir%\system32\inetsrv\appcmd set config -section:sites -
siteDefaults.logfile.directory:"D:\LogFiles"
```

Moving log file stores to a non-system drive or partition separate from where web applications run and/or content is served is preferred. Additionally, folder-level NTFS permissions should be set as restrictive as possible. Administrators and SYSTEM are typically the only principals requiring access.

Note: while standard IIS logs can be moved and edited using IIS Manager, additional management tool add-ons are required in order to manage logs generated by other IIS features, such as Request Filtering and IIS Advanced Logging. These add-ons can be obtained using the Web Platform Installer or from Microsoft's site.

Audit:

To verify web logs are being logged to the new location, open Windows Explorer and browse to the path that was defined. Depending on how the logging was configured, there will be either a) a folder containing .log files or b) .log files in the root of the specified directory.

Default Value:
The default location for web logs in IIS 7.0 is: `%SystemDrive%\inetpub\logs\LogFiles`

References:
1) http://learn.iis.net/page.aspx/579/advanced-logging-for-iis-70---custom-logging#open

1.5.2 Enable Advanced IIS Logging (Level 1, Scorable)

Description:
IIS Advanced Logging is a module which provides flexibility in logging requests and client data. It provides controls that allow businesses to specify what fields are important, easily add additional fields, and provide policies pertaining to log file rollover and Request Filtering. HTTP request/response headers, server variables, and client-side fields can be easily logged with minor configuration in the IIS management console. It is recommended that Advanced Logging be enabled, and the fields which could be of value to the type of business or application in the event of a security incident, be identified and logged.

Rationale:
Many of the fields available in Advanced Logging many can provide extensive, real-time data and details not otherwise obtainable. Developers and security professionals can use this information to identify and remediate application vulnerabilities/attack patterns.

Remediation:
IIS Advanced Logging can be configured for servers, Web sites, and directories in IIS Manager. To enable Advanced Logging using the UI:

1) Open Internet Information Services (IIS) Manager
2) Click the server in the Connections pane
3) Double-click the Advanced Logging icon on the Home page
4) Click Enable Advanced Logging in the Actions pane

The fields that will be logged needs to be configured using the Edit Logging Fields action. As with IIS's standard log files, their location should be changed.

Audit:
Browse to the location of the AdvancedLogs and verify .log files are being generated. Note that logs will be written to disk after a non-determined period of time. They can be written into their specified directory immediately if, in the Log Definition, the Publish real-time events and Write to disk options are selected.

Default Value:
IIS Advanced Logging is not enabled.

References:

1) http://learn.iis.net/page.aspx/579/advanced-logging-for-iis-70---custom-logging#open
2) http://technet.microsoft.com/en-us/library/cc732826%28WS.10%29.aspx

1.6 FTP Requests

1.6.1 Encrypt FTP Requests (Level 1, Not Scorable)

Description:
The new FTP Publishing Service for IIS 7.0 supports adding an SSL certificate to an FTP site. Using an SSL certificate with an FTP site is also known as FTP-S or FTP over Secure Socket Layers (SSL). FTP-S is an RFC standard (RFC 4217) where an SSL certificate is added to an FTP site and thereby making it possible to perform secure file transfers.

Note: the new FTP service for IIS 7.0 is not installed by default nor is it available for enabling. FTP 7.5 for IIS 7.0 is available from Microsoft's web site.

Rationale:
By using SSL, the FTP transmission is encrypted and secured from point to point and all FTP traffic as well as credentials are thereby guarded against interception.

Remediation:
To secure an existing FTP site using a SSL Certificate, a certificate must first be installed on the system. Production systems should always use a third party certificate from a trusted root, such as VeriSign. Once that certificate is installed for use in IIS, follow the steps below to configure the FTP site for SSL:

1) Open IIS Manager, select the FTP server and choose FTP SSL Settings in the Features View pane
2) Under the SSL Certificate dropdown, choose the SSL certificate to be configured for use
3) In the SSL Policy section, click the radio button next to Require SSL connections; it is important to require SSL, because allow SSL still permits non-SSL FTP
4) Click Apply in the Actions pane

Audit:
The FTP site will now require the use of FTP-S; test this by attempting to use an FTP client which either does not support FTP-S or is not configured to use FTP-S. If setup was successful, the request will fail. Conversely, open a command prompt from the server and type `ftp localhost`. After entering credentials the server should return and Access is denied message.

Default Value:
By default, the FTP site is not SSL enabled.

References:
1) http://www.windowsnetworking.com/articles_tutorials/IIS-FTP-Publishing-Service-Part3.html
2) http://learn.iis.net/page.aspx/304/using-ftp-over-ssl/#03

2 Appendix B: Change History

Date	Version	Changes for this version
December 29th, 2010	1.0.0	Public Release