Sitecore® Experience Platform™ 7.5

Sitecore Security Hardening Guide

Recommendations for making Sitecore more secure

Table of Contents

Chapter 1

Security Improvements

This chapter describes the steps you can take to improve security in your Sitecore installation.

This chapter contains the following sections:

- Introduction
- General Security Information
- Limiting Access to .XML, .XSLT, and .MRT Files
- Protecting Folders in the IIS
- Restrict Access to Client Interfaces
- The Structure of the Website Folder
- Turn off Auto Complete of Username in the Login Page
- Making the Sitecore Login Page Available to SSL Requests Only
- Controlling File Upload
- Secure the Telerik Controls
- Protecting Media Requests
- Security and Client RSS Feeds
- Removing Headers from Responses
- Recommended Reading

1.1 Introduction

The Security Hardening Guide is designed to help you make your Sitecore® Experience Platform™ installation as secure as possible.

Sitecore is of course subjected to rigorous testing before each release and any bugs or security threats that may exist are fixed and removed as soon as they are discovered. We also release updates whenever necessary.

However, the way you implement your Sitecore installation has a significant effect on the security of your website.

This document contains details of our best practices and recommendations for ensuring that your Sitecore installation is as secure as possible.

Sitecore is not responsible for the security of any other software products that you use with your website. We strongly recommend that you install *every* available service pack and update for *all* of the software products that you use.

It is important to remember that secure software is a goal that we are constantly trying to achieve but may never reach.

Security is risk management; it is about understanding the risks and concrete threats to your environment and mitigating against them. You must analyze the threats and risks that your installation faces and then do your utmost to secure your installation against these threats.

This document does not describe the Sitecore Security system. For more information about the Sitecore security system, see the *Security Administrators Cookbook*.

1.2 General Security Information

Although Sitecore can run on several different operating systems, we recommend that you use the newest operating systems with the most up-to-date security features. Use the Windows update / Automatic update service to keep all your client computers and servers up-to-date with the most recent security updates and service packs.

You should also create a disaster recovery plan to ensure the rapid resumption of services should a disaster occur. The recovery program should include:

- A plan for acquiring new or temporary equipment.
- A plan for restoring backups.
- Testing the recovery plan.

When you use the installation program to install Sitecore, all of the appropriate security settings are set. However, if you install Sitecore from a `.zip` file or if you install a website on a server without running the `setup.exe`, there are a number of settings that you will have to set manually. These settings are described in detail in the *Sitecore CMS Installation Guide*.

1.2.1 Change the Administrator Password

Before you deploy your Sitecore installation, you *must* change the administrator password to a strong password. Changing the password prevents unauthorized users from using the default password to access the admin account.

1.3 Limiting Access to .XML, .XSLT, and .MRT Files

To improve the security of your Sitecore installation, you must edit the `web.config` file. This file is stored in the `\WebSite` folder of you installation, for example at `C:\Inetpub\wwwroot\YourWebsite\WebSite`

To limit access to `.XML`, `.XSLT`, and `.MRT` files:

1. Open the `web.config` file.
2. Add the following lines to the `<system.webServer><handlers>` section:

```
<system.webServer>
  <handlers>

    <!-- Add managed handler for IIS Classic Mode in order to prevent access to files
         Notice: Must correspond to the handlers defined in <httpHandlers> section -->
    <add path="*.xml" name="xml Handler (classic)" verb="*" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework\ v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0" />
    <add path="*.xslt" name="xslt Handler (classic)" verb="*" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework\ v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0" />
    <add path="*.config.xml" name="config.xml handler (classic)" verb="*" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework\ v4.0.30319\aspnet isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0" />
    <add path="*.mrt" name="mrt handler (classic)" verb="*" modules="IsapiModule"
scriptProcessor="%windir%\Microsoft.NET\Framework\ v4.0.30319\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv4.0" />

    <!-- Prevent files from being served in IIS Integrated Mode -->
    <add path="*.xml" verb="*" type="System.Web.HttpForbiddenHandler" name="xml (integrated)"
preCondition="integratedMode"/>
    <add path="*.xslt" verb="*" type="System.Web.HttpForbiddenHandler" name="xslt (integrated)"
preCondition="integratedMode"/>
    <add path="*.config.xml" verb="*" type="System.Web.HttpForbiddenHandler" name="config.xml
(integrated)" preCondition="integratedMode"/>
    <add path="*.mrt" verb="*" type="System.Web.HttpForbiddenHandler" name="mrt (integrated)"
preCondition="integratedMode"/>
```

3. Add the following lines to the `<system.web><httpHandlers>` section:

```
<system.web>
  <httpHandlers>

    <!-- Prevent files from being served in IIS Classic Mode -->
    <add path="*.xml" verb="*" type="System.Web.HttpForbiddenHandler" validate="true" />
    <add path="*.xslt" verb="*" type="System.Web.HttpForbiddenHandler" validate="true" />
    <add path="*.config.xml" verb="*" type="System.Web.HttpForbiddenHandler" validate="true" />
    <add path="*.mrt" verb="*" type="System.Web.HttpForbiddenHandler" validate="true" />
```

Sitecore under Windows x64

If Sitecore is running under Windows x64, you must set the scriptProcessor attribute to the Framework64 folder: `scriptProcessor="%windir%\Microsoft.NET\Framework64\....."` .

1.3.1 Providing Access to Specific Files

The above configuration restricts access to all files with described extensions. To allow a specific file path to be accessed in an unrestricted manner (such as `/sitemap.xml`), add the following changes:

1. Open the `Web.config` file.

2. Add the following line to the `<system.webServer><handlers>` section:
   ```
   <add path="sitemap.xml" verb="GET" type="System.Web.StaticFileHandler"
   name="xml allow" />
   ```
3. Add the following lines to the `<system.web><httpHandlers>` section:
   ```
   <add path="sitemap.xml" verb="GET" type="System.Web.StaticFileHandler"
   name="xml allow" />
   ```

1.4 Protecting Folders in the IIS

You can improve security by preventing anonymous users from accessing certain key folders.

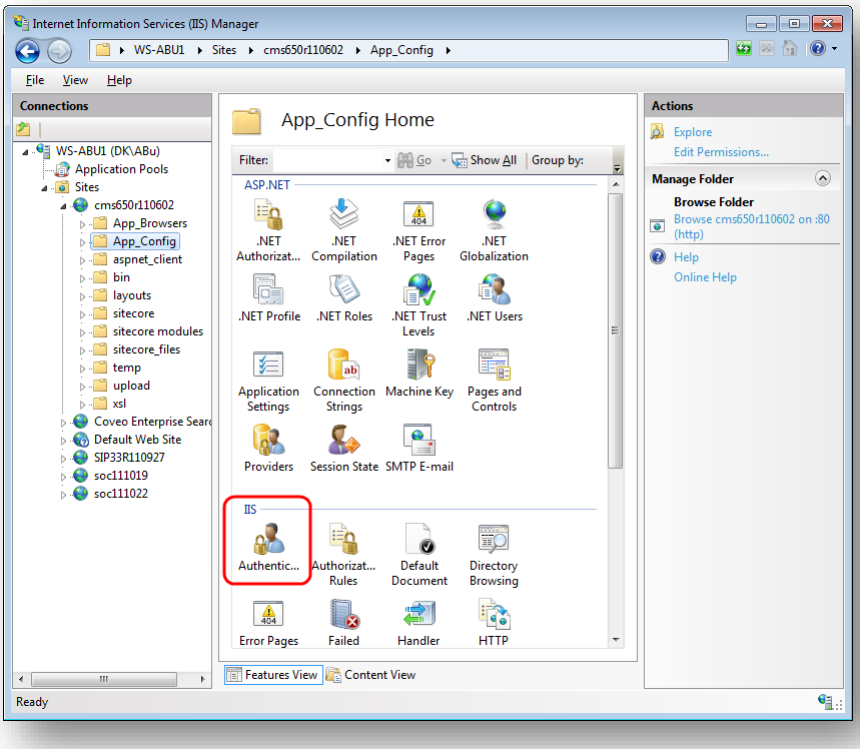
You should prevent anonymous users from accessing the following folders:

- */App_Config*
- */sitecore/admin*
- */sitecore/debug*
- */sitecore/shell/WebService*

1.4.1 Limiting Anonymous Access to Folders

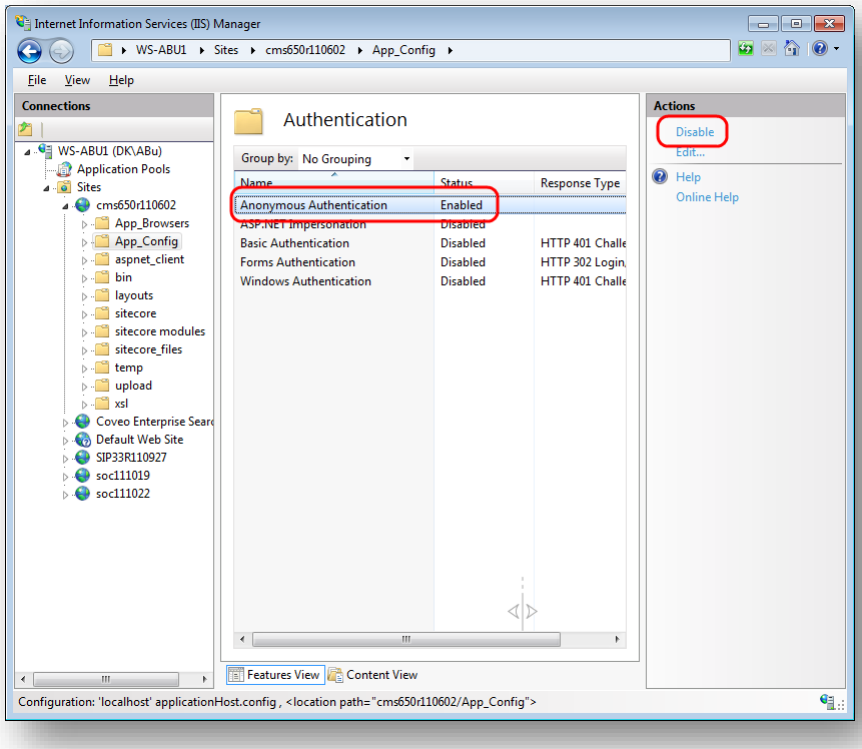
To limit anonymous access to the /`App_Config` folder:

1. Open the IIS.
2. Navigate to the `Web Sites\Default Web Site\App_Config` folder.



3. In **Features View**, double-click **Authentication**.

4. In the **Authentication** window, select **Anonymous Authentication** and in the **Actions** panel, click **Disable**.



5. Restart IIS.

Repeat this process for the `/sitecore/admin`, `/sitecore/debug` and `/sitecore/shell/WebService` folders.

1.5 Restrict Access to Client Interfaces

To prevent unauthorized access to the Sitecore client interfaces, you must restrict access to the client interfaces on every Sitecore content delivery server.

To restrict access to the client interfaces, we recommend that you implement IP-based security restrictions or disable Anonymous IIS access to the `/sitecore/admin`, `/sitecore/login`, `/sitecore/shell` folders and to the `/sitecore/default.aspx` page.

The `/sitecore/service` folder should be excluded from the IIS restrictions because it contains a number of service ASPX pages that are used by Sitecore to report various conditions that can occur in the application, such as, *404 Page Not Found* or *403 Forbidden* to the web client.

However, you can move the files from the `/sitecore/service` folder to the `/sitecore` folder. If you move the files, you must also update the following settings in the `web.config` file:

- `ErrorPage`
- `NoAccessUrl`
- `NoLicenseUrl`
- `LayoutNotFoundUrl`
- `ItemNotFoundUrl`
- `LinkItemNotFoundUrl`

For more information about configuring IP-based security restrictions in IIS 7 and later, see http://www.iis.net/ConfigReference/system.webServer/security/ipSecurity.

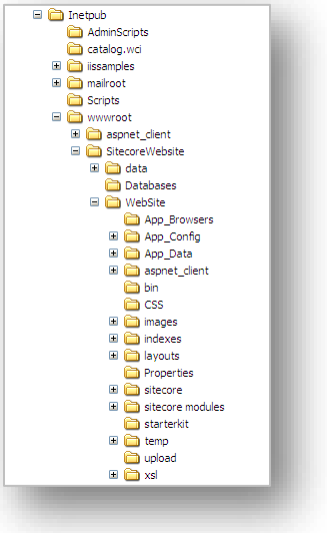
For more information about configuring IP-based security restrictions in IIS 6 and earlier, see http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/128d26dd-decb-42f9-8efb-30724d1a2f29.mspx?mfr=true.

1.6 The Structure of the Website Folder

You can improve security by placing the following folders outside the website root folder:

- */data*
- */indexes*

After moving the `/data` folder, you must edit the `web.config` file to point to the new location. You must also configure permissions for ASP.NET requests. For more information, see the section *File System Permissions for ASP.NET Requests* in the *CMS Installation Guide*.



You can install Sitecore using:

- The installation program.
- A `.zip` file.

Using the Installation Program

If you use the installation program to install Sitecore, the `/data` folder is created outside the website root folder and the `web.config` file is edited to point to this location. The `/indexes` folder is placed in the `/data` folder.

This is the recommended configuration and you don't need to make any changes.

Using a .zip File

If you use a `.zip` file to install Sitecore, the data folder is created outside the website root folder, but the `web.config` file is not edited to point to this location. The `/indexes` folder is placed in the `/data` folder. When you run Sitecore for the first time, it creates another data folder in the `/WebSite` folder.

We therefore recommend that you edit the `web.config` file to point to the correct location.

The `web.config` file should look like this:

```
        <sitecore database="SqlServer">
          <sc.variable name="dataFolder" value="C:\Inetpub\wwwroot\SitecoreWebsite\data\" />
          <sc.variable name="mediaFolder" value="/upload" />
          <sc.variable name="tempFolder" value="/temp" />
```

1.7 Turn off Auto Complete of Username in the Login Page

You can specify that Sitecore should not complete the username of users automatically when they log in. This is useful, for example, if you do not want user names to be disclosed when content authors log into Sitecore on a shared or public computer. In addition, you can disable the Remember me checkbox.

- To disable auto complete of user names, open the web.config file and set the Login.DisableAutoComplete setting to true. This disables autocomplete on the Sitecore login forms on the /sitecore/login/default.aspx and /sitecore/admin/login.aspx pages.
- To disable the Remember me checkbox on the login page, open the web.config file and set the Login.DisableRememberMe setting to true. This also ignores any existing Remember Me cookies, and all users have to log in again.

1.8 Making the Sitecore Login Page Available to SSL Requests Only

You can configure the Sitecore CMS to use only SSL requests for the Sitecore login page.

Create a custom redirect processor that will redirect from *http://hostname/sitecore/login* to *https://hostname/sitecore/login*, and redirect all other pages from *https* to *http*.

Use the following code as an example:

```
        public class SslLogin
            {
                public void Process(PipelineArgs args)
                {
                    string absUrl = HttpContext.Current.Request.Url.AbsoluteUri;
                    string localUrl = HttpContext.Current.Request.Url.LocalPath;

                    if (localUrl.StartsWith("/sitecore/login") && absUrl.StartsWith("http://")
&& !Context.IsLoggedIn)
                    {
                        HttpContext.Current.Response.Redirect(absUrl.Replace("http://",
"https://"));
                        return;
                    }
                    if (!localUrl.StartsWith("/sitecore/login") && absUrl.StartsWith("https://")
&& Context.IsLoggedIn)
                    {
                        HttpContext.Current.Response.Redirect(absUrl.Replace("https://",
"http://"));
                    }
                }
            }
```

1.9 Controlling File Upload

You can strengthen security of your Sitecore installation by controlling access to files that are uploaded by the users.

1.9.1 Deny Execute Permissions on the Upload Folder

If you allow users to modify the content of the /`upload` folder, you also give them the permission to place scripts and executable programs in the folder. Executing these scripts and programs can cause an unexpected behavior on the server. You must therefore prevent an uploaded file from being executed on the server side when a user attempts to download it.

We recommend that you deny permissions to run scripts and executable files in the `/upload` folder.

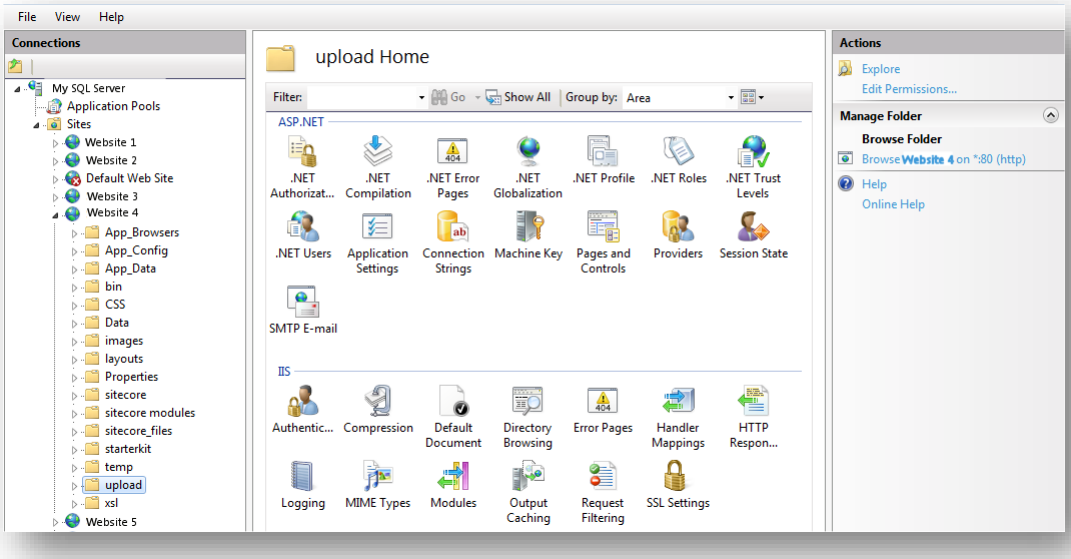
Note
You only need to perform this step if your configuration allows content authors to place files directly to the `/upload` folder. For example, if you use a shared directory or FTP server, content authors can quickly place a lot of media in the media library.

For more information about *Execute* permission in IIS, see http://support.microsoft.com/kb/313075.

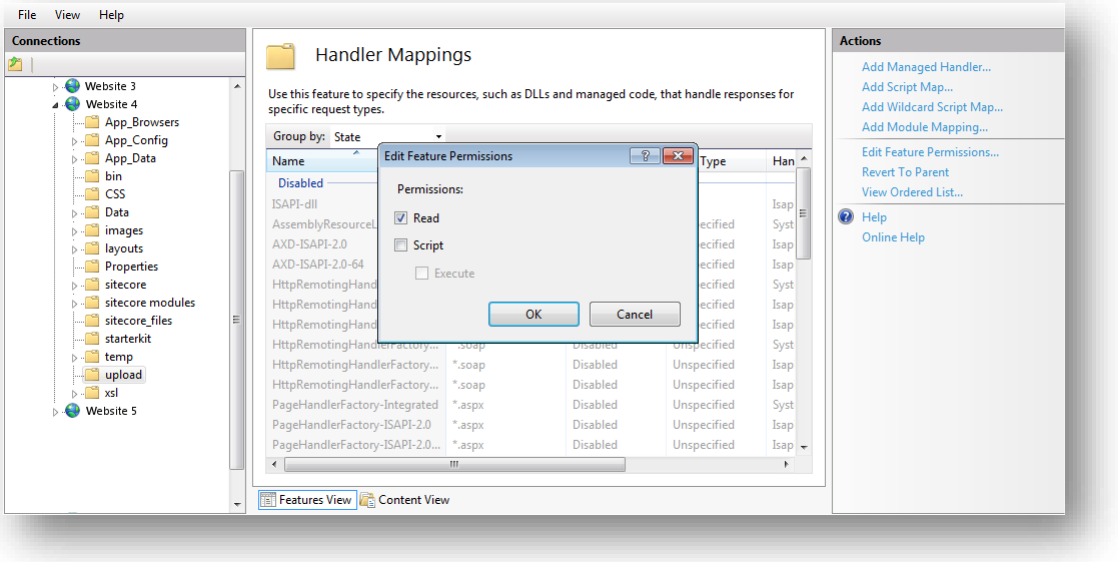
Denying Execute Permission in IIS

You must deny both *Script* and *Execute* permission to the upload folder.

1. Navigate to the `/upload` folder for the database that you are interested in.



2. Select the `/upload` folder and click **Handler Mappings** and then in the **Actions** pane, click **Edit Feature Permissions**.



3. In the **Edit Feature Permissions** dialog box, clear the **Script** and **Execute** check boxes.

Prevent Users from Uploading Files to the Temp Folder

You should also deny users *Script* and *Execute* permission to the `/temp` folder. This prevents them from uploading files to the `/temp` folder.

Note
This step is primarily needed if your configuration allows content authors to place files directly to the `/temp` folder using a shared directory or a FTP server. But we recommend that you perform this step in any case to avoid potential security problems if `.aspx` files for some reason end up being saved in the `/temp` folder (for example from custom code).

1.9.2 Disabling the Upload Watcher

We recommend that you disable **Upload Watcher** and thereby ensure that the only way to upload files is from the **Media Library**. This ensures that you can only upload files from within the Sitecore client and have control over the files that are uploaded.

When **Upload Watcher** is disabled, files that are placed in the `/upload` folder are not automatically uploaded to the **Media Library**.

To disable the **Upload Watcher**, remove the following line from the `<modules>` section of the `Web.config` file.

```
  <system.webServer>
    <modules>
      <remove name="ScriptModule"/>
      <add type="Sitecore.Nexus.Web.HttpModule,Sitecore.Nexus" name="SitecoreHttpModule"/>
      <add type="Sitecore.Resources.Media.UploadWatcher, Sitecore.Kernel"
name="SitecoreUploadWatcher"/>
```

1.9.3 The Upload Filter Tool

If you want to have complete control and prevent users from uploading certain file types, you should use the **Upload Filter** tool.

The **Upload Filter** tool lets you prevent certain file types from being uploaded, for example `.exe`, `.dll`, and so on.

You can download the Upload Filter tool — `Upload Filter-1.0.0.2.zip` — from the Sitecore Developer Network where it is available as a Sitecore package file along with this Security Hardening Guide.

The Sitecore package contains the following files:

File Name	Destination Folder
`UploadFilter.config`	`Website\App_Config\Include\`
`UploadFilter.dll`	`WebSite\bin\`

Installing the Upload Filter Tool

You must install the Sitecore package file before you can use the **Upload Filter** tool.

To install Upload Filter tool:

1. In the **Sitecore Desktop**, click **Sitecore**, **Control Panel**.
2. In the **Sitecore Control Panel**, click **Administration**, **Install a Package**.
3. The wizard will guide you through the installation process.

Configuring the Upload Filter Tool

After you install the package, you must configure the tool.

To configure the **Upload Filter** tool:

1. Open the `UploadFilter.config` file.

```
<processors>
      <uiUpload>
          <processor mode="on" type="Sitecore.Pipelines.Upload.CheckExtension,
Sitecore.UploadFilter" patch:before="*[1]">
              <param desc="Allowed extensions (comma separated)"></param>
              <param desc="Blocked extensions (comma separated)">exe,dll</param>
          </processor>
      </uiUpload>
</processors>
```

2. In the `Allowed extensions` parameter, enter a comma-separated list of the file extension types that can be uploaded.

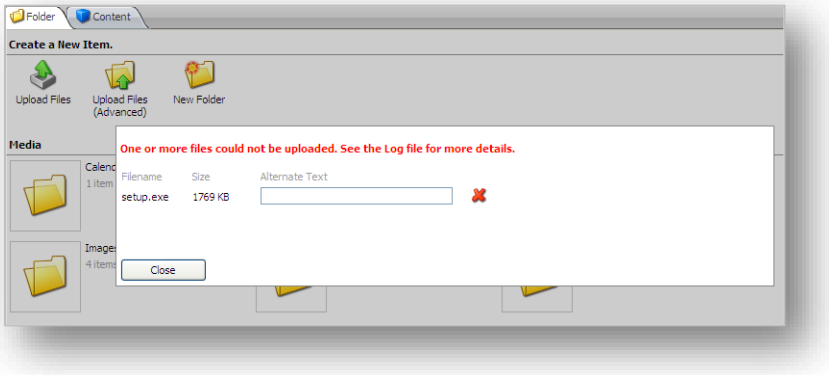
   Or

   In the `Blocked extensions` parameter, enter a comma-separated list of the file extension types that cannot be uploaded.

   You must enter the file extension without the dot.

   Important
   If you set the `Allowed extensions` parameter, the `Blocked extensions` parameter is ignored.

3. If you try to upload a file type that is on the blocked list, you see the following message:



1.10 Secure the Telerik Controls

Sitecore uses some UI controls from Telerik. These controls are only used in a Content Management environment.

To reduce the attack surface area:

1. In a Content Delivery environment, in the `web.config` file, remove the following nodes:

```
<add name="Telerik_Web_UI_DialogHandler_aspx" verb="*" preCondition="integratedMode"
path="Telerik.Web.UI.DialogHandler.aspx" type="Telerik.Web.UI.DialogHandler" />

<add name="Telerik Web UI SpellCheckHandler axd" verb="*" preCondition="integratedMode"
path="Telerik.Web.UI.SpellCheckHandler.axd" type="Telerik.Web.UI.SpellCheckHandler" />

<add name="Telerik Web UI WebResource axd" verb="*" preCondition="integratedMode"
path="Telerik.Web.UI.WebResource.axd" type="Telerik.Web.UI.WebResource" />
```

2. In a Content Management environment, you must configure the encryption key that is used to secure the Telerik upload control.

   In the `web.config` file, in the `appSettings` section, create a node for the Telerik configuration encryption key and enter a `"STRONG-RANDOM-VALUE-UNIQUE-TO-YOUR-APP"`.

   For example:

```
<appSettings>
    <add key="Telerik.AsyncUpload.ConfigurationEncryptionKey" value="STRONG-RANDOM-VALUE-
UNIQUE-TO-YOUR-APP" />
</appSettings>
```

For more information, see the Telerik documentation.

1.11 Protecting Media Requests

To use the Sitecore media request protection feature optimally and make your solution more secure, open the `/App_Config/Include/Sitecore.Media.RequestProtection.config` file and change the `Media.RequestProtection.SharedSecret` setting to a random string.

In a multi-server setup, you must use the same value for the `Media.RequestProtection.SharedSecret` setting on every server. This ensures that dynamic image scaling will work correctly when the image URL is generated by one server and the request is handled by a different server.

1.12 Security and Client RSS Feeds

RSS technology is designed so that users who follow an RSS link can come directly to the item specified in the URL of the RSS feed. Most RSS readers do not support authentication. This means that users who subscribe to Sitecore client RSS feeds have direct access to the item specified in the URL of the RSS feed and do not have to identify themselves to the Sitecore security system when they view the RSS feed. However, the Sitecore security system verifies that they are authorized users when they try to perform any actions associated with the client feed.

If someone else gains access to the URL of the RSS feed:

- They *can* follow the link and view all the content contained in the RSS feed even though their own security permissions do not give them access to this item.
- They *cannot* perform any actions on the content.
- They *cannot* view any other content.
- They *cannot* gain access to the username or password of the original owner of the RSS feed.
- They *cannot* modify the link to gain access to any other content.

Important
Sitecore users should not share RSS feeds.

1.12.1 Disabling Client RSS Feeds

If your Sitecore installation contains sensitive information that you want to protect, you can disable Sitecore client RSS feeds.

To disable Sitecore client feeds:

1. Open the `web.config` file.
2. Locate the `<httpHandlers>` section. Depending on your IIS pool, this section may be called Handlers.
3. Remove the following handler:

```
<add verb="*" path="sitecore_feed.ashx"
type="Sitecore.Shell.Feeds.FeedRequestHandler, Sitecore.Kernel"/>
```

Removing this handler disables all the client feeds that are available inside Sitecore. However, any public RSS feeds that you have created are still available to Website visitors.

1.13 Removing Headers from Responses

You can improve security and save a small amount of bandwidth by removing header information from each response sent by your web site.

These headers contain a number of infrastructure details about the framework used on your website that you do not need to publicize.

You can easily remove:

- The X-Aspnet-Version HTTP header.
- The X-Powered-By HTTP header.
- The X-AspNetMvc-Version HTTP header.

1.13.1 Removing the X-Aspnet-Version HTTP Header

By removing the X-Aspnet-Version HTTP header information from each web page, you save a little bandwidth and ensure that you are not publicizing which version of ASP.NET you are using.

To remove the X-Aspnet-Version HTTP header from each response from ASP.NET, add the following to the `web.config` file.

```
<system.web>
  <httpRuntime enableVersionHeader="false" />
</system.web>
```

For more information about removing the X-Aspnet-Version HTTP header, see http://www.dotnetperls.com/x-aspnet-version.

1.13.2 Removing the X-Powered-By HTTP Header

By removing the X-Powered-By HTTP header, you are not publicizing which version of ASP.NET you are using.

To remove the X-Powered-By HTTP header from each response from ASP.NET, add the following to the `web.config` file.

```
<system.webServer>
  <httpProtocol>
    <customHeaders>
      <remove name="X-Powered-By" />
    </customHeaders>
  </httpProtocol>
</system.webServer>
```

1.13.3 Removing the X-AspNetMvc-Version HTTP Header

By removing the X-AspNetMvc-Version HTTP header, you are not publicizing which version of ASP.NET MVC you are using.

To remove the X-AspNetMvc-Version HTTP header, add the following to the `Application_Start` method in the `Global.asax.cs` file:

```
protected void Application_Start(object sender, EventArgs e)
{
  MvcHandler.DisableMvcResponseHeader = true;
  // RegisterRoutes etc... and other stuff
}
```

1.14 Recommended Reading

For information about how to make Sitecore more secure, we recommend that you read the following documents:

- Sitecore Installation Guide.

  http://sdn.sitecore.net/Products/Sitecore%20V5/Sitecore%20CMS%207/Installation.aspx
- The upgrade instructions for the Sitecore version you are upgrading from.
- The `web.config` changes documentation for the Sitecore version you are upgrading to.

1.14.1 Other Resources

For more information about SQL Server security, visit:

http://technet.microsoft.com/en-us/library/bb545450.aspx

http://www.microsoft.com/en-us/sqlserver/solutions-technologies/mission-critical-operations/security-and-compliance.aspx

For more information about security in general, visit the Microsoft Security TechCenter:

http://technet.microsoft.com/en-us/security/default.aspx

MongoDB

Mongo DB is secure at the network level by default.

However, Sitecore recommends that customers follow MongoDB's best practices to harden their installation.

For more information about MongoDB security, visit:

http://www.mongodb.com/blog/post/mongodb-security-best-practices

http://docs.mongodb.org/manual/administration/security/?_ga=1.171844135.606011462.1424169294