Sitecore CMS 6

Microsoft ASP.NET Security Vulnerability 2416728

Sitecore's recommendations for working around this vulnerability

Last updated: 2010-09-27

Microsoft ASP.NET Security Vulnerability

2416728

Sitecore recommends that you follow these instructions to address the Microsoft ASP.NET Security Vulnerability 2416728.

http://www.microsoft.com/technet/security/advisory/2416728.mspx

or

http://weblogs.asp.net/scottgu/archive/2010/09/18/important-asp-net-security-vulnerability.aspx

Important

The steps described in this article are a temporary solution and should be rolled back when Microsoft releases an official fix.

Applying this solution may have the following side effects when processing requests that are configured in IIS to be forwarded to the ASP.NET process:

1. Going through Sitecore pipelines adds a minor overhead when serving static files.
2. The application will attempt to resolve any request going to the ASP.NET process as a Sitecore item regardless of the requested extension. For example, both `/home.aspx` and `/home.html` will resolve to an item with the name “Home”.

An example of a modified `web.config` file (from the clean installation of Sitecore 6.2.0 rev. 100507) and the source code of the `Sitecore.Support.MSA2416728v6000.dll` file can be found in the package that is attached to this article.

Instructions

1. Find the `FilterUrlExtensions` processor of the `preprocessRequest` pipeline in the `web.config` file and make the following changes:

```
<preprocessRequest help="Processors should derive from
Sitecore.Pipelines.PreprocessRequest.PreprocessRequestProcessor">
   ...
   <processor type="Sitecore.Pipelines.PreprocessRequest.FilterUrlExtensions,
Sitecore.Kernel">
        <param desc="Allowed extensions (comma separated)">*</param>
        <param desc="Blocked extensions (comma separated)"></param>
        <param desc="Blocked extensions that stream files (comma separated)">*</param>
        <param desc="Blocked extensions that do not stream files (comma separated)">
        </param>
   </processor>
   ...
</preprocessRequest>
```

2. Add a `CustomErrorPage.aspx` file to your application that contains the appropriate HTML contents. This file will be displayed every time an error occurs within the web application. An

example of the `CustomErrorPage.aspx` can be found in the package that is attached to this article.

3. Add the following code to the `Page_Load()` server event handler in the `CustomErrorPage.aspx` file. This adds a random, small sleep delay.

```
<script type="text/C#" runat="server">
protected void Page_Load(object sender, EventArgs e)
{
            Response.Clear();
    Response.Cache.SetCacheability(HttpCacheability.NoCache);
    byte[] delay = new byte[1];
    System.Security.Cryptography.RandomNumberGenerator prng = new
System.Security.Cryptography.RNGCryptoServiceProvider();
    prng.GetBytes(delay);
    System.Threading.Thread.Sleep((int)delay[0]);
    IDisposable disposable = prng as IDisposable;
    if (disposable != null) { disposable.Dispose(); }
}
</script>
```

4. Set the following Sitecore settings in the `web.config` file to point to `/CustomErrorPage.aspx` file:

`ErrorPage, ItemNotFoundUrl, LayoutNotFoundUrl, LinkItemNotFoundUrl, NoAccessUrl, NoLicenseUrl.`

For example:

```
<setting name="ErrorPage" value="/CustomErrorPage.aspx" />
```

5. In the `web.config` file, set the value of the `RequestErrors.UseServerSideRedirect` setting to `true`.
6. Add the `Sitecore.Support.MSA2416728v6000.dll` file from the package that is attached to this article the `\bin` folder of your Sitecore solution.
7. If you are running .NET 3.5 with SP1, make the following modifications to the `customErrors` section of the `web.config` file:

```
   <!-- The "defaultRedirect" attribute should contain the path to your custom error page
-->
   <customErrors mode="On" redirectMode="ResponseRewrite"
defaultRedirect="/CustomErrorPage.aspx" />
```

8. If you are running .NET 3.5 without SP1, make the following modifications to the `customErrors` section of the `web.config` file:

```
   <!-- The "defaultRedirect" attribute should contain the path to your custom error page
-->
   <customErrors mode="On" defaultRedirect="/CustomErrorPage.aspx" />
```

9. In the `web.config` file, add the following HTTP module definitions to both the `/configuration/system.web/httpModules` section and the `/configuration/system.webServer/modules` section:

```
   <system.webServer>
    <modules>
      ...
      <add type="Sitecore.Support.MSA2416728v6000.RedirectOnError,
Sitecore.Support.MSA2416728v6000" name="SitecoreRedirectOnError" />
    </modules>
    ...
   <system.webServer>

   <system.web>
    ...
    <httpModules>
      ...
      <add type="Sitecore.Support.MSA2416728v6000.RedirectOnError,
Sitecore.Support.MSA2416728v6000" name="SitecoreRedirectOnError" />
    </httpModules>
```

```
...
</system.web>
```

10. In the `web.config` file, in both the `/configuration/system.web/httpHandlers` section and the `/configuration/system.webServer/handlers` section, replace the `Sitecore.Resources.Media.MediaRequestHandler, Sitecore.Kernel` type reference with `Sitecore.Support.Resources.Media.MediaRequestHandler, Sitecore.Support.MSA2416728v6000`.
11. Install IIS UrlScan and add a custom rule as described in the Microsoft Security Advisory (after the *Using UrlScan* heading in the *Workarounds* section):

    http://www.microsoft.com/technet/security/advisory/2416728.mspx

    If you are using IIS7.5, you can configure IIS request filtering as an alternative to installing IIS UrlScan. This approach is described in the Microsoft Security Advisory (after the *Using IIS request filtering* heading in the *Workarounds* section):

    http://www.microsoft.com/technet/security/advisory/2416728.mspx

    For more information about UrlScan, see the following Microsoft ASP.NET blog post:

    http://weblogs.asp.net/scottgu/archive/2010/09/24/update-on-asp-net-vulnerability.aspx