DotNetNuke Security Notice

September 18, 2010

Background

A critical vulnerability in ASP.NET was publically disclosed late Friday at a security conference in Argentina. We recommend all DotNetNuke users immediately apply a workaround (described below) to prevent attackers from using this vulnerability against your DotNetNuke (and any other ASP.NET) applications.

We initially became aware of the potential existence of this vulnerability Tuesday and reached out to the two researchers involved. The researchers responded and made us aware that the vulnerability was an ASP.NET issue. They had successfully used it against a DotNetNuke install and intended to demonstrate that during a conference on Friday, September 17th. They also confirmed that original reports that changing the encryption scheme in the web.config of ASP.NET websites were incorrect and did not stop their exploit.

The DotNetNuke Security Team immediately began reviewing all the available material on this and similar oracle padding attacks and developed potential mitigations. In addition, we reached out to our contacts in the ASP.NET team and the Microsoft Vulnerability Research (MSVR) team to communicate the additional information we had and to ensure we would hear about any further details or workarounds they identified.

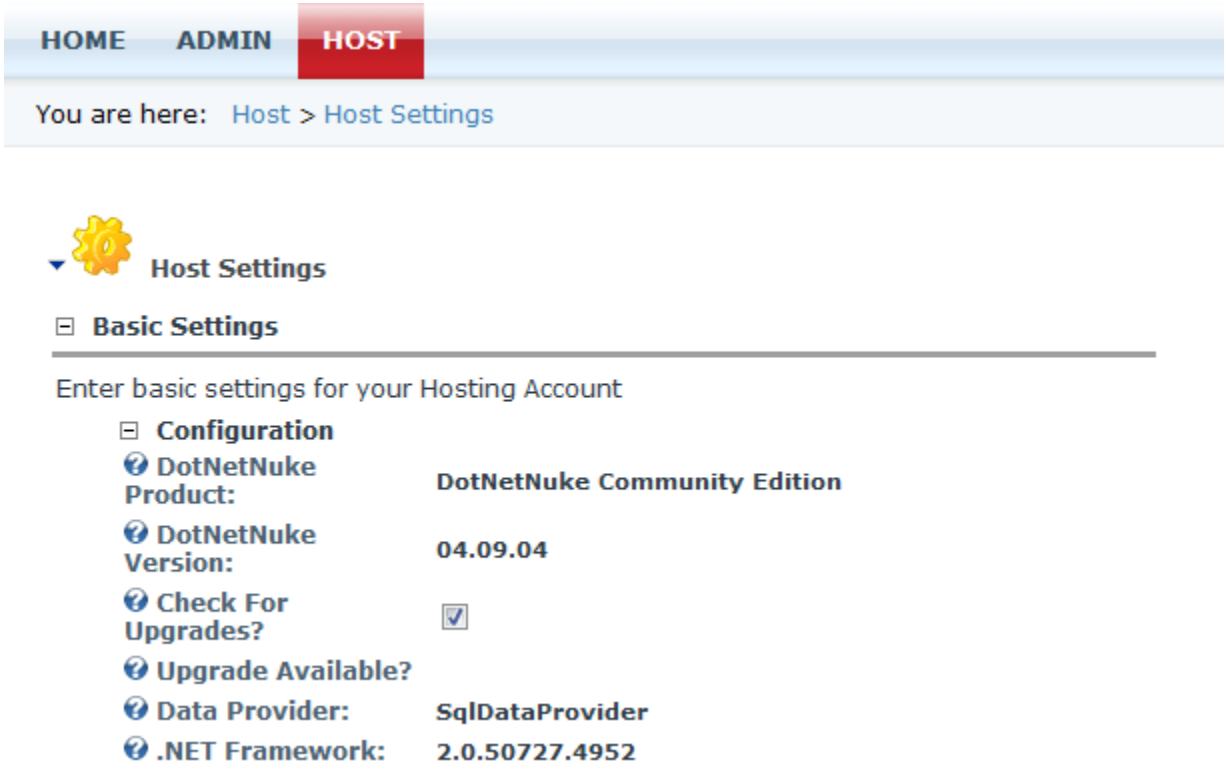
Immediately after the first public demonstration of the ASP.NET vulnerability (and a related video demonstration), Microsoft determined a workaround that will protect ASP.NET sites (including DotNetNuke sites), while they work on a more permanent solution, likely a server patch to resolve this at the machine/framework level.

Determining Framework Version

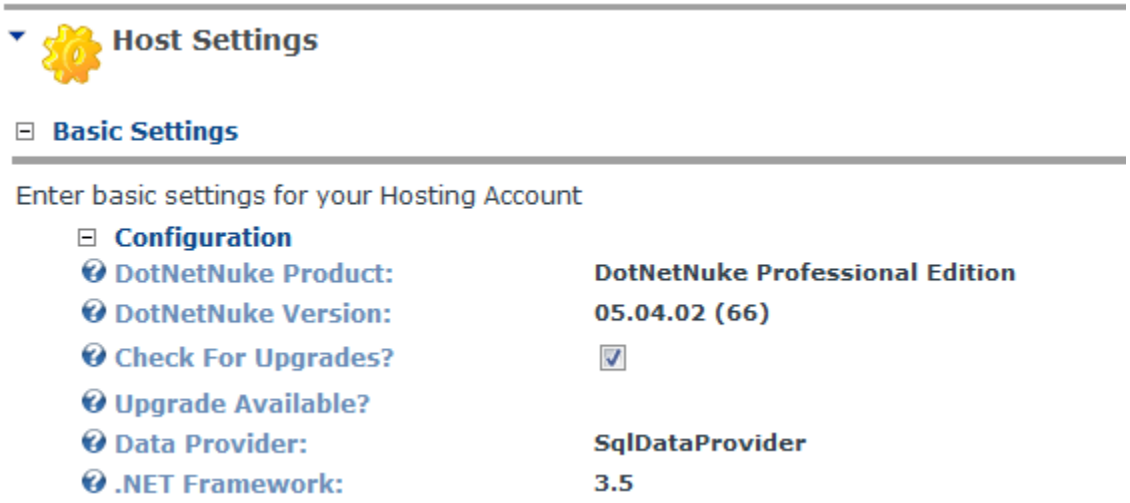
There are two separate workarounds dependent on the .NET framework version a website is using, so it is important that you determine the correct version to use before applying the workaround.

Identifying the Framework Version within DotNetNuke

To determine the.NET framework version a DotNetNuke site is running against, log in as host and go to Host->Host Settings. The version will be shown under the .NET framework field. In the example shown in the screen shot below, the site is running under .NET 2.0 and should use the first version of the fix.

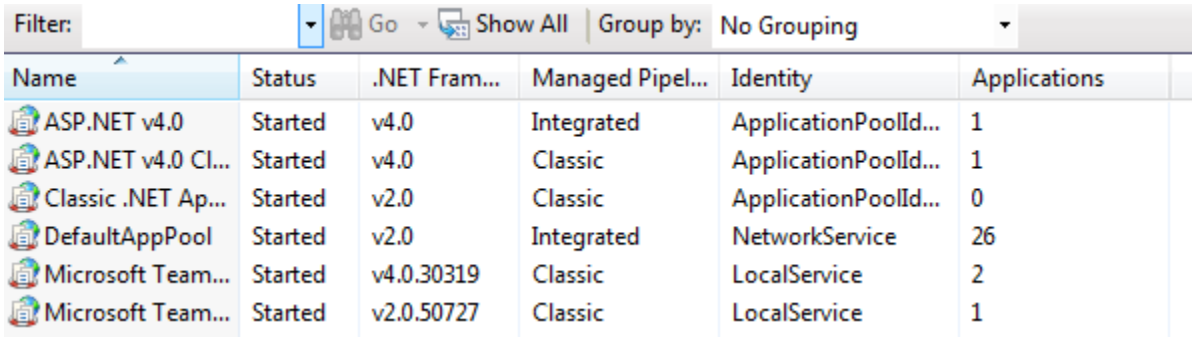


In the screen shot shown below, DotNetNuke is running under .NET 3.5 and should use the second variant of the fix.

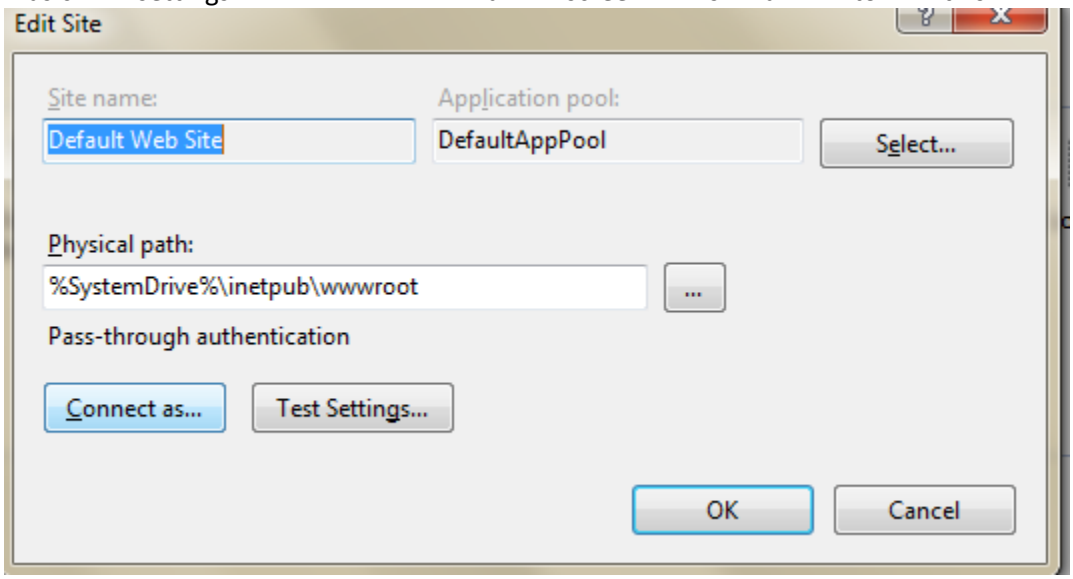


Getting the Framework Version from IIS

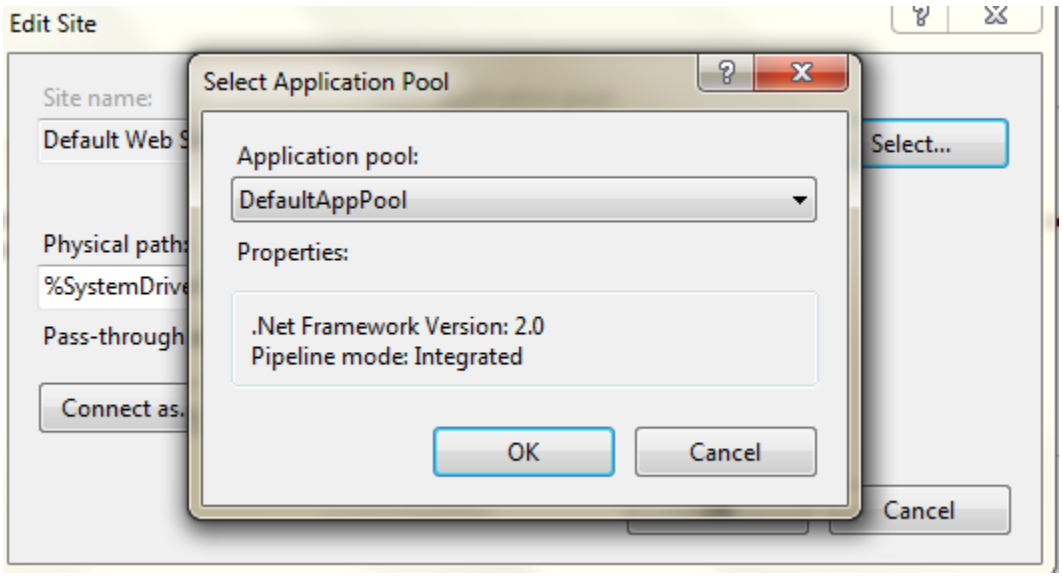
If you have access to Internet Information Server (IIS), open it and expand out the "Application pools" node. This will display a list of available application pools and their .NET framework version.



If you're unsure what application pool your website is using, select the website in IIS, and then click the "Basic settings" link - a screen similar to this will appear:



Note: Clicking the select button will open the application pool dialog which also contains the framework version as shown in the screen shot below:



Getting the Framework Version from the web.config

The version of the framework in use is also shown in the web.config file. The value is in a number of locations, but the most obvious will be the <assemblies> node. If it is using .NET 3.5 or higher, it will contain a number of entries that reference the version number as shown below:

<add assembly="System.Core, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />

Applying the Workaround

Note: The following is extracted from the blog at http://weblogs.asp.net/scottgu/archive/2010/09/18/important-asp-net-security-vulnerability.aspx:

> If you are using ASP.NET 1.0, ASP.NET 1.1, ASP.NET 2.0, or ASP.NET 3.5 then you should follow the below steps to enable <customErrors> and map all errors to a single error page:
>
> 1) Edit your ASP.NET Application's root Web.Config file. If the file doesn't exist, then create one in the root directory of the application.
>
> 2) Create or modify the <customErrors> section of the web.config file to have the below settings:

<configuration> <system.web> <customErrors mode="On" defaultRedirect="~/error.html" /> </system.web> </configuration>

3) You can then add an error.html file to your application that contains an appropriate error page of your choosing (containing whatever content you like). This file will be displayed anytime an error occurs within the web application.

Notes: The important things to note above are that customErrors is set to "on" and all errors are handled by the defaultRedirect error page. There are no per-status code error pages defined – which means there are no <error> sub-elements within the <customErrors> section. This avoids an attacker being able to differentiate why an error occurred on the server, and prevents information disclosure.

Enabling the Workaround on ASP.NET V3.5 SP1 and ASP.NET 4.0

If you are using ASP.NET 3.5 SP1 or ASP.NET 4.0 then you should follow the below steps to enable <customErrors> and map all errors to a single error page:

1) Edit your ASP.NET Application's root Web.Config file. If the file doesn't exist, then create one in the root directory of the application.

2) Create or modify the <customErrors> section of the web.config file to have the below settings. Note the use of redirectMode="ResponseRewrite" with .NET 3.5 SP1 and .NET 4.0:

<configuration> <system.web> <customErrors mode="On" redirectMode="ResponseRewrite" defaultRedirect="~/error.aspx" /> </system.web> </configuration>

3) You can then add an Error.aspx to your application that contains an appropriate error page of your choosing (containing whatever content you like). This file will be displayed anytime an error occurs within the web application.

Note: We highly recommend you read both the Microsoft advisory http://www.microsoft.com/technet/security/advisory/2416728.mspx and Scott Guthries blog for further details http://weblogs.asp.net/scottgu/archive/2010/09/18/important-asp-net-security-vulnerability.aspx and for an example of the error.aspx page.

What Happens Now?

We have already taken steps to protect our DotNetNuke properties to ensure they are safe and to ensure no user data is accessible. We have created an action plan to push this message out as widely as possible. Our development team is working on a fix we can include in the upcoming 5.5.1 release. We are also developing a solution for people who cannot upgrade to 5.5.1 quickly and are not confident about applying the manual workaround. We will release more details once we have them.