Securing DotNetNuke

Hardening DotNetNuke Installations

Cathal Connolly

Revision 1.0.0
Last Updated: May 24, 2006
Applies to: DotNetNuke ver. 3.2+

Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this document remains with the user.

The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, places, or events is intended or should be inferred.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Perpetual Motion Interactive Systems, Inc. Perpetual Motion may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Perpetual Motion, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

DotNetNuke® and the DotNetNuke logo are either registered trademarks or trademarks of Perpetual Motion Interactive Systems, Inc. in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Abstract

The purpose of this guide to explain some of the areas where users can harden the security model of DotNetNuke installs. It contains advice for both planned installs (“pre-installation hardening”) and existing installations (“post-installation hardening”).

Contents

Chapter 1: Getting Started

Introduction

The purpose of this guide to assist users in assessing how best to secure installations of DotNetNuke. Due to the large amount of different environments in which DotNetNuke can be deployed, it is not a prescriptive guide, but rather one that should help you evaluate the settings that best fit for your deployments.

DotNetNuke has existing for a number of years now, and as with any evolving project, some of this history has mandated particular decisions and scenarios. Some of these areas can, with care, be hardened to provide a greater degree of protection against potential attacks. In this guide we will cover both guidelines that can be followed for new installations and guidelines that can be used to tighten up existing deployments.

Before you beginning planning DotNetNuke installs, you should ensure that you have secured your servers, both IIS, and if used, SQL. Please review these Microsoft guides to help you harden default installations.

Steps to help secure SQL 2000

Secure IIS5 checklist

Managing a secure IIS6 solution

Do I really need to apply these steps?

That decision is of course up to you. DotNetNuke has had a number of security penetration tests, by a number of different entities, with generally favorable results. Any identified areas of potential weakness have been addressed, and now in many cases multiple layers of blended security exist to counter common web application attacks.

However, diversity is always an admirable concept in security. Taking the time to harden your installations, and make them different from vanilla installs is yet another layer of protection.

Chapter 2: Authentication choices

Authentication Schemes

When running DotNetNuke, you have 2 layers of authentication – the database itself, and users within the portal. Both of these have 2 different schemes that can be chosen from. You may not have options in this area, for instance if you're using shared web hosting, with a shared SQL server, your setup is dictated by your ISP, but if you're using DotNetNuke in an intranet/extranet scenario, or control your own servers, you should consider the following areas before installing DotNetNuke.

Database

DotNetNuke ships with a SQL database provider. This allows you to connect to SQL 2000, MSDE 2000, SQL 2005 and SQL express. All of these applications can support 2 different authentication schemes known as Windows authentication and SQL authentication. When you install any of these products, Windows authentication is enabled by default, but commonly SQL authentication will be added also (leaving the database in what is known as "mixed mode")

Windows authentication requires the use of a user, often a domain user, though a local machine user can be used also. This scheme is regarded as being more secure than SQL authentication, as it does not require you to specify the username or password in the connection string in your web.config file. It manages this as your domain/local machine will authenticate the user context by passing a hash of the user's password, rather than the password itself.

In addition, by using a windows based user, that user can then leverage windows policies such as time limited passwords, password recycling etc. It also allows your network administrators greater control over the database itself, which is a common requirement in enterprises.

The other scheme, SQL authentication, requires that you create a user in the server program, and then map it to a login in the database (your hosting provider will normally have done this for you). To then authenticate, you add the users name and password in your connection string. Whilst it's unlikely that anyone will gain access to your web.config file, it is a possibility. This reason, and Windows authentications other advantages, mean that it's regarded as being safer, and should be considered if your setup will support it.

Please see the "DotNetNuke Installation Guide.pdf" document provided in the documentation download for further details on how to correctly configure any applicable connection strings.

User authentication

Since 3.12, DotNetNuke has had integrated support for Active Directory authentication, along with its traditional forms authentication database based code. Using an Active directory authentication scheme again allows you to utilize the domain policies, and similarly to Windows authentication with SQL Server, is regarded as being a safer option.

Database authentication is the default setting for DotNetNuke. To learn more about configuring DotNetNuke for Active Directory authentication, please see the documentation provided on the Active Directory Project page.

Chapter 3: Reducing Surface area

DotNetNuke ships with a number of different modules and features, not all of which you may need or want in your installation(s). Reducing the potential attack surface area of a system by running only those services and features needed is a good security best practice. Ideally you should refrain from installing un-required components, but it's also possible to remove modules after installation.

Pre-installation procedure

During installation, DotNetNuke checks the Install Folder, and iterates through it's sub folders, installing resources. Before you install DotNetNuke, please explore these folders, and remove any components, particularly modules, that you do not plan to use in this particular install or modules that have been superseded by more advanced versions e.g. the discussions module if often removed as the forums module is much more advanced. To delete it go to install/module and remove Discussions*.zip (note: the name will differ depending on the version of DotNetNuke and whether you use the source or install downloads)

It's also a good idea to check the DotNetNuke projects page , and download the latest version of components you intend to use, as they offer updated versions that contain fixes for security issues.

Post-installation procedure

If you have existing DotNetNuke installations and want to remove unwanted modules, you can do this via the existing GUI.

- Log in as a Superuser (e.g. the "Host" user)
- Go to Host->Module Definitions
- Click on the pencil icon to the left of the module you want to remove.

- Press the “Delete” link towards the bottom of the page, and press Ok to the confirmation dialog that pop’s up.
- Repeat the process for any other modules you wish to remove.

Web.config settings

There are a number of settings in your web.config that can be altered to provide stronger security. In some cases, you will have to edit installation templates, to ensure that the changes you make to the web.config do not cause an error during installation/upgrade.

Forms node

In asp.net 1.1, persistent cookies have a lifetime of 50 years. However, if you’re running DotNetNuke under asp.net 2.0, then you can control the lifetime of persistent cookies by setting the forms timeout. If the timeout period is not specified in the web.config, the asp.net framework defaults to 30 minutes.

DotNetNuke ships with a file called release.config, which you rename to web.config during the installation procedure. This file specifies a timeout of 60 minutes, which is reasonably secure. However, some organizations may choose to reduce this to a smaller value, so if a user is away from their PC, the potential window of opportunity is reduced e.g. to reduce the timeout from 60 minutes to 15 minutes alter the web.config as follows:

Default setup

```
<authentication mode="Forms">
   <forms name=".DOTNETNUKE" protection="All" timeout="60" />
</authentication>
```

More secure version

```
<authentication mode="Forms">
   <forms name=".DOTNETNUKE" protection="All" timeout="15" />
</authentication>
```

DNNSQLMembershipProvider node

minRequiredPasswordLength & minRequiredNonalphanumericCharacters.

These two attributes are used in conjunction to determine the required complexity of the passwords DotNetNuke requires. Their defaults are 4 and 0 respectively. These values are used, as historically they correspond to the default length and complexity of the "host" account. You can alter these to more secure defaults i.e. 8 characters , one of which must be non-alphanumeric

Default Setup

```
<add name="DNNSQLMembershipProvider"
type="DotNetNuke.Security.Membership.DNNSQLMembershipProvider, DNNSQLMembershipProvider"
connectionStringName="SiteSqlServer" enablePasswordRetrieval="true"
enablePasswordReset="true" requiresQuestionAndAnswer="false"
minRequiredPasswordLength="4" minRequiredNonalphanumericCharacters="0"
requiresUniqueEmail="false" passwordFormat="Encrypted" applicationName="/"
description="Stores and retrieves membership data from the local Microsoft SQL Server
database" />
```

More Secure Setup – increased password length to eight characters, and required at least one non-alphanumeric characters.

```
<add name="DNNSQLMembershipProvider"
type="DotNetNuke.Security.Membership.DNNSQLMembershipProvider, DNNSQLMembershipProvider"
connectionStringName="SiteSqlServer" enablePasswordRetrieval="true"
enablePasswordReset="true" requiresQuestionAndAnswer="false"
minRequiredPasswordLength="8" minRequiredNonalphanumericCharacters="1"
requiresUniqueEmail="false" passwordFormat="Encrypted" applicationName="/"
description="Stores and retrieves membership data from the local Microsoft SQL Server
database" />
```

Note: If you choose to increase the web.config defaults, you must also alter the two default users ("admin" and "host") so their passwords match this rule, otherwise installation will fail.

To do this, open the file at install/DotNetNuke.install.resources. Locate the <superuser> and <administrator> nodes, and change the respective <password> subnodes to reflect the new rules e.g.

Default setup

```
<superuser>
      <firstname>SuperUser</firstname>
      <lastname>Account</lastname>
      <username>host</username>
      <password>host</password>
      <email>host</email>
      <locale>en-US</locale>
      <timezone>0</timezone>
```

```
   </superuser>
```

Secured setup – host password is eight characters long with one alpha-numeric character

```
<superuser>
      <firstname>SuperUser</firstname>
      <lastname>Account</lastname>
      <username>host</username>
      <password>dnnhost!</password>
      <email>host</email>
      <locale>en-US</locale>
      <timezone>0</timezone>
   </superuser>
```

Note: This change is probably the single most important hardening step, as increasing the length and complexity of the two known accounts ("host" and "admin"), significantly reduces the chances of someone cracking one of them using a dictionary based attack.

With the release of DotNetNuke 3.3/4.1, the default password length has been extended to 7, and the relevant files have been updated.

passwordFormat

The DNNSQLMembershipProvider node also defines the storage format of the passwords. By default DotNetNuke uses encryption of user passwords. This provides a good level of protection, and allows you to retrieve your password as encryption is a reversible operation. However, if you do not wise to support password retrieval, or want to ensure maximum protection, you may choose to use Hashing instead. Hashing is a non-reversible operation, so even if your database is accessed or stolen, a hacker cannot reverse engineer your password.

Snippet of default setup

```
…requiresUniqueEmail="false" passwordFormat="Encrypted" applicationName="/"…
```

Alternative setup utilizing Hashed passwords

```
…requiresUniqueEmail="false" passwordFormat="Hashed" applicationName="/"…
```

Objectqualifier and databaseowner attributes

These two attributes control how SQL Server will create and reference objects. Whilst on the surface, altering these does not appear to add any additional security, diversity is always a good idea with regards to security. A number of well known portal products, and web applications have suffered attacks by automated robots, after having exploits discovered.

If such a case were to happen to DotNetNuke, the automated attacks would be likely to assume the default settings for both these, so using alternative values may provide some additional security i.e. delete from tabs will work on a default portal, but if a portal has an objectqualifer (e.g “secdnn”) then the tabs table would be called something different (“secdnn_tabs”) and this sql statement would fail.

Chapter 3: Altering Portal settings

Installation Template Settings

It is a good idea to determine in advance what your ideal hardened settings are, and to create a custom installation template (you can use the DotNetNuke.install.resources one as a blueprint). For a full description of templates, please see "DotNetNuke Host Template.pdf" document provided in the documentation download. We've already seen one example of the templates (altering the admin and host user passwords), but there are a large number of settings that can be controlled via templates. As before, it is preferable to determine these before installation, but the settings can also be altered via the interface later.

The following list details the settings that have security implications, and the recommended values for these. In the interests of brevity, we are listing them in the form

"User interface label"/*Templatevalue* – description

Where the "user interface label" is the text visible via the interface e.g. on the Admin->Site Settings screen, and the italicized *Templatevalue* is the equivalent template value.

DotNetNuke.install.resources template/Host menu

To access the host menu, log in as a superuser, and go to Host->Host Settings

To alter the template, edit the file at install/ DotNetNuke.install.resources

- "Show Copyright Credits"/ *Copyright* – ensure this is checked. This will remove both the version number from the title and the copyright credits from the output, which will help protect against potential hackers profiling your application.

- "Use custom error messages"/ *UseCustomErrorMessages* – ensure this is checked. If not exceptions will be thrown as normal, if checked then exceptions will be passed to the custom logging mechanism, which will log the issue, and if

the module allows error displays then displays the error in the nominated errorplaceholder.

- "Anonymous demo signup"/*DemoSignup* - ensure this is unchecked (set to "N") – this will stop users from creating their own portals. Allowing anonymous users this level of access is not advised.

- "Auto-unlock accounts after (minutes)"/ *AutoAccountUnlockDuration* - After an account is locked out due to unsuccessful login attempts, it can be automatically unlocked with a successful authentication after a certain period of time has elapsed. Enter the number of minutes to wait until the account can be automatically unlocked. Enter "0" to disable the auto-unlock feature. The default is 10, it is recommended that you either set it to a large value or disable the autounlock function by setting it to 0. If you set it to zero, then the only way accounts can be unlocked is if a portal admin or superuser unlocks them via the "User Accounts" menu.

- "File upload permissions"/ *FileExtensions* – best practices suggest removing any unnecessary file extensions. If after installing your portal, you do not plan on installing additional skins/modules, remove "zip".

- "Skin upload permissions"/ *SkinUpload* – ensure this is set to host, as skins can contain dynamic content, and their use should be constrained.

Additional Notes

It's recommended that if possible, you configure your email server to not support anonymous authentication. Log into host-host settings, and select basic or NTLM authentication, and provide any necessary username and password (or use the *SMTPAuthentication* , *SMTPUsername* and *SMTPPassword* template values). If using anonymous, it is recommended that you configure your mail server so it only accepts emails from non-routable IP ranges (i.e 127.0.0.1/10.0.0.*/192.168.1.*) and the IP address you've configured your web server as.

Export templates

Whilst it is possible to use the existing templates and the "DotNetNuke Host Template.pdf" document to alter them accordingly, DotNetNuke also supports exporting site templates. To create a custom template:

- Log in as Host and create a DotNetNuke portal by going to Host->Portals.
- Click on “Add new portal”. Specify the new portal details, taking care to use a portal admin with a password that matches your desired complexity rules.
- Navigate to your new portal, and log in with your portal admin.
- Go to Admin->Site Settings, and alter the settings you require.
- Log out of the portal, and log back in as a host user.
- Go to Host->Portals, and select the portal name in the “Export Template” dialog. You can now use the exported template file as a good starting point for future portal installations.

Removing portal functions

Host functions are automatically installed, and aren’t controlled by templates. In some cases these may be not required, in which case it is a good idea to remove them. The “SQL” menu is a good example of this – unless you utilize this function, it is safer to remove it, as if an attacker gains access to your portal, it makes running arbitrary SQL much easier.

To remove the function, you will have to delete if from the correct table e.g.

```
delete from tabs where tabname='sql'
```

Note: you will have to amend this query to reflect your choice of objectqualifer/databaseowner

Removing portal administrator functions can be done the same way, but as they are controlled by templates, it is better to amend the templates in advance. To do this, edit the admin.template file in \portals_default, and remove any unused/unnecessary functions by deleting the correct <tab></tab> blocks.

Chapter 4: Additional checklist

The following are a few areas you should check before releasing your portal(s) to production.

Web.config

- <add key="AutoUpgrade" value="false" /> – The default value is true, consider setting this to “False” as shown. This way you can control when and if upgrades are installed
- <compilation debug="false" /> - ensure this is set to false so you do not run the risk of disclosing too much information to potential attackers
- <customErrors mode="RemoteOnly" /> - ensure this is set to “RemoteOnly" so you do not run the risk of disclosing too much information to potential attackers

It is highly recommended that you run your DotNetNuke installations in medium trust code access security. Please refer to “DotNetNuke Code Access Security.pdf” document in the documentation download for further details, and help with coping with common issues with medium trust.

Scheduled Tasks

DotNetNuke comes with a number of scheduled tasks, that perform background operations and can help optimize the portal. In some cases, these provide additional services, such as purging the schedule history, which you may wish to disable, to ensure that any portal changes are always initiated by you, and not by any background task. In particular DotNetNuke.Modules.Admin.ResourceInstaller.InstallResources and DotNetNuke.Services.FileSystem.SynchronizeFileSystem should be considered. The first handles automatic installation of portal resources (i.e. skins/modules etc.), and the second scans the portal folders for new/altered files and writes the updated details to the database.

To alter scheduled task settings:

- Log in as a Superuser such as “host”
- Go to Host->Schedule
- Click on the pencil icon to the left of the scheduled task(s) you wish to alter
- To disable a task, uncheck the “Schedule enabled” checkbox.
- To delete it altogether, click the “delete” link towards the bottom of the page.

Additional Information

The DotNetNuke Portal Application Framework is constantly being revised and improved. To ensure that you have the most recent version of the software and this document, please visit the DotNetNuke website at:

http://www.dotnetnuke.com

The following additional websites provide helpful information about technologies and concepts related to DotNetNuke:

DotNetNuke Community Forums
http://www.dotnetnuke.com/tabid/795/Default.aspx

Microsoft® ASP.Net
http://www.asp.net

Open Source
http://www.opensource.org/

W3C Cascading Style Sheets, level 1
http://www.w3.org/TR/CSS1

Errors and Omissions

If you discover any errors or omissions in this document, please email marketing@dotnetnuke.com. Please provide the title of the document, the page number of the error and the corrected content along with any additional information that will help us in correcting the error.

Appendix A: Document History

Version	Last Update	Author(s)	Changes