

Re: AzMan Still the way to go?

Source:

<http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.framework.aspnet.security/2006-08/msg00249.html>

- *From:* "Joe Kaplan" <joseph.e.kaplan@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 25 Aug 2006 14:06:47 -0500
-

Agreed with D. His book should be very good too. I'm looking for it. I like mine too, but it is really only appropriate if you need to do LDAP programming, and I'd say that isn't needed here.

The thing to know with Windows authentication (IWA in IIS) in ASP.NET is that the user will be authenticated by the OS and your code will be given a `WindowsPrincipal` object representing the authenticated user. The `WindowsPrincipal` contains the user's name and their groups. You can check the groups directly for authorization by doing `IsInRole`.

If you want to create a mapping between the principals in AD (users and groups) and your application-specific roles, that is one thing that AzMan shines at. However, it is also big and complex and may be overkill. A very simple approach is to create a simple mapping between AD groups and your application roles and then create a new `GenericPrincipal` object in the `Authenticate` event handler of `global.asax` that simply reads the users groups, figures out which mapped application roles they get, and then creates the appropriate array of roles to feed into the `GenericPrincipal` constructor. After that, the rest of your code can authorize based on the application-specific roles. This type of approach is basically like a poor man's AzMan, where you have to implement the storage of the role mappings and the implementation of the role mappings yourself and don't get the advantage of having the powerful role/task/operation model that AzMan supports. The benefit is that there is no black box and you understand exactly how everything works. The downside is that you may have to write more code, won't get a pretty UI for maintaining the mapping policy for free, and have less abstraction in your model, which may mean that your code ends up being harder to maintain. It is all trade-offs. :)

It may also be possible to use the SQL role provider to get some of this functionality; I'm not sure how it integrates with Windows auth. The thing you want to try to avoid is storing all of your users in SQL, as that will create a maintenance nightmare with keeping the data in sync with AD. Ick!

Joe K.
