

# self-inflicted authentication issue

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2007-07/msg01389.html>

---

- *From:* kpg <[no@xxxxxxxx](mailto:no@xxxxxxxx)>
  - *Date:* Mon, 23 Jul 2007 07:25:24 -0700
- 

ASP.NET 2.0

I have an unusual situation dealing with forms authentication, not doubt brought on by how I have structured the application.

The setup:

Users enter the site from one of several pages, A,B,C, etc. These pages allow all users. In these pages I setup site customizing session variables then redirect users to a common home page.

The home page is protected by deny anonymous users, so users are sent to a login page to be authenticated. Authentication is based on how they entered the site, A,B,C.

I set things up this way so I can use a common login page, but that page is customized based on how the user entered the site: A,B,C. (different logo, text, etc.).

The problem:

Let's say a user entered on page 'A', gets authenticated and is sitting on the home page. Then they edit the browser URL to navigate to page 'B'. (the user should not do this, but users do all sorts of thing they shouldn't do).

Well, since the user is already authenticated the login screen is by-passed. Additionally, since the home page is already loaded in their browser, the home Page\_Load event is not fired. This results in Page A authentication but Page B session variables – a big mess

To solve this problem I want to un-authenticate the user in the page\_load event of the entry pages (A,B,C..), but removing the ASPXAUTH cookie does not seem to unauthenticate the user as the login page is not displayed for some reason, but it does force the home page\_load event, so this is half correct – but

still no good.

I tried setting the web.config to allow anonymous, deny all on the entry pages (A,B,C), and this works upon normal entry, but when the user navigates there after authorization they can't get past the login screen, because once they are authorized they are denied access to the requested entry page and sent back to the login page – and endless loop.

I did come up with something that works: I check for the ..ASPXAUTH cookie on the entry page and if it is present I send the user to an error page.

To improve on this I keep a session variable "LastValidPage", and if the user enters the site while authenticated (the cookie is present) I simply redirect them to this page, so from the users standpoint nothing has happened.

My question is – is there a better way?

I thought of having each entry page actually be a customized login page (but I like the idea of having a single login page).

I'm not sure how this would work, the home page would deny anonymous, so the system would want to redirect to a login.aspx page – I suppose I could have the login.aspx page display an error message instead of presenting a login. If the user tried to access the home page directly they would get the login error page. Then if the user changed to a different entry page mid-site, they would need to login based on that page's criteria and everything could be kept straight.

Thoughts or comments?

Thanks.  
kpg