

# Re: aspnet\_isapi.dll security limit access to all but 1 file

---

*Source:*

<http://www.derkeiler.com/Newsgroups/microsoft.public.inetserver.iis.security/2006-11/msg00040.html>

---

- *From:* "David Wang" <[w3.4you@xxxxxxxxxx](mailto:w3.4you@xxxxxxxxxx)>
  - *Date:* 11 Nov 2006 15:36:49 -0800
- 

What you want to do is technically impossible given your requirements. What is not clear is an understanding of how the IIS 6.0 and ASP.Net 2.0 request pipelines intermingle, so you will want to read and understand the following blog entries. I still have an unwritten blog entry to explain what is actually failing with your Attempt #1.

[http://blogs.msdn.com/david.wang/archive/2005/10/14/HOWTO\\_IIS\\_6\\_Request\\_Processing\\_Basics\\_Part\\_1.aspx](http://blogs.msdn.com/david.wang/archive/2005/10/14/HOWTO_IIS_6_Request_Processing_Basics_Part_1.aspx)  
<http://blogs.msdn.com/david.wang/archive/2005/10/15/Why-Wildcard-application-mapping-can-disable-Default-D>  
<http://blogs.msdn.com/david.wang/archive/2005/10/16/Why-Wildcard-application-mapping-is-not-catching-404s.a>  
[http://blogs.msdn.com/david.wang/archive/2005/06/29/IIS\\_User\\_Identity\\_to\\_Run\\_Code\\_Part\\_2.aspx](http://blogs.msdn.com/david.wang/archive/2005/06/29/IIS_User_Identity_to_Run_Code_Part_2.aspx)  
<http://blogs.msdn.com/david.wang/archive/2006/04/28/HOWTO-Run-Console-Applications-from-IIS6-on-Window>

The closest hack to get what you want is to configure aspnet\_Isapi.dll as a Wildcard application mapping.

The underlying issue is this – your custom authentication/authorization protocol only applies wherever aspnet\_isapi.dll applies, and aspnet\_isapi.dll only applies at the IIS level, not File/Directory level. Thus, you must make sure that all resource access go through IIS (and aspnet\_isapi.dll) and not through NTFS File/Directory or anything else on IIS.

The insecurity of the custom AuthN/AuthZ protocol is permanent because its trusted computing base (TCB) is the process identity, which is shared between tom and bob. Thus, if tom has access to that process identity (such as by calling `RevertToSelf()`), he can bypass your AuthN/AuthZ protocol to access bob's resources. And this bypass is by-design since the TCB is supposed to be able to access both tom and bob'