

"An 'Asp.Net' accident waiting to happen" – Draft article

Source:

<http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.framework.aspnet.security/2003-10/0327.html>

From: Dinis Cruz (*dinis_at_ddplus.net*)

Date: 10/30/03

Date: 29 Oct 2003 16:04:38 -0800

Hello

Please see bellow the final draft of an article soon to be published.

I would appreciate your comments and corrections of anything that I might have got wrong.

Best regards

Dinis Cruz

An 'Asp.Net' accident waiting to happen

I would like to call the attention of the Asp.Net Community and Microsoft to an accident waiting to happen. In a time where Security is finally being taken seriously by Microsoft, their focus is still in adding features to products and not making the existent products secure.

The accident will be the wide spread exploitation of websites hosted in shared hosting environments (such as ISPs). The problems described next will also affect any major Asp.Net application, but their problems will be dealt privately and lessons will not be learnt by the community.

Asp.Net is the latest and most powerful web application development tool produced by Microsoft. Although it is a major technological advance from its predecessors (Asp, ISAPI, Web Classes, Custom COM Objects developed in Visual Studio 6, etc.) it is also very dangerous.

The .Net framework (which is fully implemented in Asp.Net) is very feature rich and powerful. It provides the developer (i.e. development team) with a huge array of tools, objects and methods that make their

web application development a quick, easy and effective process.

Microsoft, knowing that security is becoming more and more a central issue for end clients, included in the .Net framework several technologies that allow the creation and deployment of secure applications. The most important are:

- Code Access Security (CAS)
- Ability to deploy APTCA (AllowPartiallyTrustedCallersAttribute) assemblies in the GAC (Global Assembly Cache)
- Built in Encryption technology (DPAPI)
- Several methods to implement Secured Database Accesses
- Several authentication methods and Build-in Impersonation features
- Extended used of Role Based Security (used in conjunction with Windows server security features)

IIS 6 web server and windows 2003 also provide some tools to deploy secure websites:

- Applications pools (allow the execution of each website under the rights on a unique (low privileged) user)
- Better management of process and threads

These tools should allow the developer to create secure applications that could be deployed in secure servers.

The problem is that today, the development and deployment of secure web applications in secure web servers is almost impossible because:

- Although CAS (Code Access Security) could be used to limit what a web application could 'do' on the host server, in the real world it doesn't work. Any web application that is executed in 'Partially trusted' environments (i.e. not in 'Full Trust') will not have access to fundamental Asp.Net features such as: Database connectivity using OleDb or ODBC, use of COM objects and many other important features.
- If a developer wants to use the powerful Asp.Net development environment in a quick, easy and effective way, the solution is to run its Asp.Net code in 'Full Trust' environments. The problem is that most security tools and technologies previously mentioned only work effectively in 'Partially trusted' environments
- Most ISPs that provide Shared Hosting environments allow their hosted websites to execute with 'Full Trust' rights. This means that even if a developer manages to get their web application to work in 'Partially trusted' environments, he would not be able to find a secure host for it. The only way around this is to purchase a dedicated server, which is also very dangerous, because it would then be the developer's responsibility to securely configure and manage it.

- Most developers will store sensitive information (such as usernames and passwords) in an unencrypted format in configuration files stored in their website's folders (for example in Web.config). This happens because the current version of Asp.Net doesn't provide a quick, effective and scalable solution to store encrypted data in the registry (or other secure location)
- In a shared hosting environment everybody has access to everybody's temporary files (i.e. the 'Asp.Net Temporary files' folder) and in Asp.Net all code is initially compiled into IL (Intermediate Language) which is easily decompiled into VB or C#. This means that every user with access to a valid account in a shared hosting environment can read the source code (.aspx, vb, cs or dll) of every website hosted in that server.
- In 'Full Trust' environments, the developer has access to the entire windows 32 API and (using reflection) all .Net functions (private or public). This means that it is easy to write code that: Executes commands on the server (i.e. creates processes), lists usernames, lists running process, list installed services, open TCP connections, etc...

Fundamentally, the problem is that the current version of the .Net Framework (version 1.14) doesn't allow the creation of secure hosting environments.

Unless (of course), one is prepared to developed 'Partially Trusted' web applications.

This could be done by using the reduced (and very limited) 'Partially Trusted' Asp.Net development environment, or by spending an extra 50% to 100% development time in creating secure strong name assemblies that can be published in the server's GAC (although this is not very practical for ISPs).

Today (Oct/2003):

- 1) There are hundreds of ISPs providing 'Full Trust' Asp.Net shared hosting services
- 2) Microsoft is not publicly acknowledging (as a problem or security vulnerability) the fact that these ISPs have no alternative but to provide 'Full Trust' shared hosting environments. And consequently is not focused on providing or developing a solution.
- 3) Microsoft is focused in developing the next version of Asp.Net (version 1.2) which will have even more features and will be even more powerful (although Microsoft says that the new version will be more secure)

4) Malicious users have already realized that the current windows based ISP's hosting environments are insecure and easy to exploit. Although most incidents are still private and the ISPs will not publicly acknowledge their existence, recently one case was so serious that it got some press attention: See the "Interland Security Incident" story where a malicious user injected malicious code in several websites hosted by Interland's shared hosting servers. This code used known bugs in the website's visitors Internet Explorer Web Browser which (description from the Wired article) "... Web surfers who have visited the compromised sites might get a surprise on their next phone bill when they discover expensive calls made to 900 numbers. Others could also find themselves party to a denial-of-service attack launched against another computer...". For more details read this article: "Wired.com : 'Security holes Vex Web Hosting Firm' : <http://www.wired.com/news/business/0,1367,60303,00.html>" and this "news.com : 'Web Hosting Company confirms hack attack' : <http://news.com.com/2100-1002-5076050.html>" and this "ComputerWorld.com : 'Security breach at Web host leaves sites at risk' : <http://www.computerworld.com/securitytopics/security/story/0,10801,84675,00.html>".

5) The ISPs are telling their clients that their shared hosting servers are secure and the clients believe them (reinforced by the fact that Microsoft doesn't acknowledge the problem). This is quite important because if the clients were aware of these problems, they would demand (and possibly pay more for) secure environments to host their websites.

The massive exploitation of these vulnerabilities by malicious users is just a matter of time. In my view the most dangerous problem is the fact that (today) there is no real alternative (i.e. solution) to securely host Asp.Net websites in a shared hosting environment.

If in a near future a major security incident occurs (with widespread media coverage) and, the ISPs had to (or where forced to) change the current 'Full Trust' level to a 'Partially trusted' level, the side effect would be that most (if not all) hosted Asp.Net websites would stop working or would seriously malfunction.

In conclusion, it is an 'Asp.Net' accident waiting to happen!

Dinis Cruz is a Security Consultant currently working for several UK governmental Departments, International Corporations and ISPs. He is specialist in .Net Security and is the creator and main developer of the Open Source web application ANSA (Asp.Net Security Consultant). He is also the managing director of DDPlus; a UK based IT Security Company (www.ddplus.net). Dinis can be contacted on dinis@ddplus.net