

Re: Security – Permissions Configuration

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2004-10/2846.html>

From: UAError (null_at_null.null)

Date: 10/27/04

Date: Wed, 27 Oct 2004 10:53:24 -0400

"Angelos Karantzalis" <akarantzalis@agiltech.gr> wrote:

>On second read of your post, I can see better where you're coming from...
>
>Well, the definition you found was a bit "constrained" ... you CAN assign
>permissions to principals and not only codebases with Java Policy Files.
>
>I'd think that CAS sound a bit like a small part of Java Authentication and
>Authorisation (JAAS) :D
>
>Consider the following:
>
>You're building an application, which supports services in the form of
>dynamically downloaded plugins. You can expect that more than one users will
>be using the same installation. Some of those plugins though, need the user
>has special authorization (role) to run, others don't. To complicate the
>matter a bit further, you can execute the plugin under a "login domain" –
>you have multiple login domains in the app, and each time you try to do
>anything, you do it under a login domain, a "context" if you will ... , and
>the plugins are shared between domains. But you need different roles to use
>them under each domain ...
>
>Now, I need a way to assign Code Access (or Data Access, or whatever
>Access) permissions – at runtime – based on the roles of my user, under the
>specific login domain she's currently using (mind you .. login domains are
>just a click on a drop-down list of domains for the user !). So ? There
>simply MUST be a way to couple CAS and RBS, because it makes sence !!! I
>can't believe that MS would leave something like that out of the framework,
>when it's trying desperately to match – and possibly exceed – the Java
>functionality.
>
>.. but most importantly ... sooner or later with the advent of C#, you can
>expect to find much more Java people invading your "turf", bringing with
>them a whole new mentality where specialization is actually ... BAD !!!
>Volvos & SAABs are different cars in behavior, but you can drive them both,
>and expect more-or-less the same basic "services" from both, don't you ?
>

>*Open your mind a bit ... ;]*

>

>*Angel*

>*O:]*

Don't ask of others what you aren't willing to do yourself
... ;]

I suspect that both SUN and Microsoft simply applied different philosophies to the implementation of security. Maybe SUN determined that the Java-VM had to be responsible for ANY aspect of security, especially as they ultimately have NO control over OS platform security.

Microsoft has FULL control over OS platform security and they chose not to duplicate security features that were already present in the OS (ignoring flavours of Win9x and below).

Windows Security can protect most platform resources with Access Control Lists (ACLs), so (at least for the time being) .NET does not duplicate Windows Security.

CAS and RBS cannot grant the user privileges and access rights he or she doesn't already have. If a .NET application tries to do something the user isn't entitled to do by virtue of the access granted to the user account and memberships in Windows Groups, the OS lets .NET know and .NET will propagate an exception to the applications Application Domain.

If you want to prevent an exception, you have the option of using `WindowsPrincipal.IsInRole` to check whether the current user belongs to a Windows group that you assume or know to have access to the resource.

While privileges are associated with an account or windows group, access rights are not – access rights are associated with the resource being protected. Currently you need to use unmanaged code to read and manipulate ACLs (that will probably change in the future).

RBS is present to enable you to protect resources that your application creates and cannot be protected by the OS platform.

CAS is there to further refine the level of trust that you have in code of a specific identity or origin.

The only way you can have an application do things the user isn't allowed to do, is by actually running the application

in the security context of a more privileged account. That practice however exposes you to a potential "Elevation of Privilege" attack – this practice has been actively discouraged since Microsoft's security pushes starting in late 2001 as it violates the "Principle of Least Privilege".

In fact the internet has grown so "hostile" that the current best practice is to run an application in a "least privilege" context which impersonates the user only in sections of code where the application needs gain privileges that USER has.

Now if you are interested in Windows Security as it affects .NET (rather than .NET Security) have a look at Keith Brown's Book on-line.

"The .NET Developer's Guide to Windows Security"
<http://pluralsight.com/wiki/default.aspx/Keith.GuideBook.HomePage>

*>when it's trying desperately to match – and possibly exceed – the Java
>functionality.*

And again, you are making assumptions. They aren't.

There is no denying they snoozed when they underestimated the impact that Java would have. Probably because most people thought of Java as a language, not a platform. So when the door slammed on a SUN/Microsoft alliance, they had to do something.

"Never engage an enemy on their own terms"

SUN et al had made significant progress so if they pitched .NET against the Java-VM they could at best hope for a tie, and that would be a faint hope.

So instead, they are using .NET as a catalyst to transform their server-line to something that may be able to leave Intel-based hardware platforms in the future, so in fact .NET is a key component of Windows struggle against the *nixes.

For the short term .NET focuses on distributed architectures simply because the market for servers that "scale up" is dominated by the *nixes. So Windows/.NET is going after the "scale out" server market – which suits corporate mentality; buy a few cheap servers this financial year, a few more the next (the fact that this requires a team of highly skilled (and expensive) administrators and developers to pull off is material for yet another debate).

Expect to see some timebased server–licensing/maintenance fee options that eliminate upfront cost of the Server OS and are somewhat competitive to the then better understood cost–of–ownership of Open Source OSs; this also fits the corporate "financial year" mentality even though it will increase the total cost of ownership over the products lifetime (car–leases took off; why not this?).

Java and .NET will both stick around and developers will have to live with it.

Windows and *nixes will also stick around and administrators will have to live it.

Meanwhile the battle for marketshare rages on ...