

## Re: Code Access Security, Evidence Based Security, Code Access Permission, Role Based Permission, etc

*Source:* <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2004-02/0341.html>

---

*From:* Ollie (*why*)

*Date:* 02/24/04

Date: Tue, 24 Feb 2004 11:14:32 -0000

Your're not the only one who is confused by this.....

"Novice" <6tc1ATqlinkDOTqueensuDOTca> wrote in message news:6359C4EA-6E1D-4AC1-BEF1-2AC1E069CE5C@microsoft.com...

> *Hi, I've done some reading on Microsoft .NET security and am a little confused on the relationship between Code Access Security, Evidence Based Security, Code Access Permission, Role Based Permission, Declarative and Imperative Security Statements, etc.*

>

> *Here is my current understanding:*

>

> *Code Access Security is access based on the identity of the code not the user running it (if this is true, then only the Identity Permission Code Access Permission should fall under CAS – i.e. role-based security shouldn't be a part of CAS). Protected operations in CAS are file I/O, access to the registry, etc. CAS is being used when an assembly's evidence is examined and a set of configurable rules (security policy) are applied to determine a code's permissions.*

>

> *There are three types of "permission classes" (I'm quoting "permission classes" because of confusion of the word permission in both Identity Permission and Role-Based Permission):*

> *Identity Permission, Code Access Permission and Role Based Permission*

>

> *Permission objects of the above types are assigned to an assembly based on its evidence and the security policy when the assembly is loaded into the CLR. All three "permission classes" fall within the domain of CAS or do only Code-Access Permissions and Identity Permissions fall within the domain of CAS?????? It seems likely to me that since you can enforce CAS using all of the "permission classes" then all three "permission classes" fall under the domain of CAS. However, the statement:*

> *"Code Access Security is access based on the identity of the code not the user running it "*

> *doesn't seem in agreement with that.*

>  
> -----  
> --*Identity Permission* --  
> *amounts to the values of host evidence associated with an assembly. To me it makes little sense to call this Identity Permission, since they are just values of the evidence of an assembly and regardless of any of the values of the evidence, the permissions granted to an assembly are based on the evidence AND the security policy – I could set up my security policy not to give any permissions to evidence of any type.*  
>  
> --*Code–Access Permission*--  
> *amounts to all the different types of permissions that can be granted to an assembly – file access, network access, etc. This makes sense, since these are actual permissions – rights to do certain things.*  
>  
> --*Role–Based Permission*--  
> *are the values associated with the user id and their role(s) (i.e. the Principal)????* Again this is confusing, these aren't permissions, these are merely values that represent the user whose account is being used to execute the assembly (yes, users can be impersonated and thus be executed as if the active user were the one being impersonated, but I'm trying to keep this as simple as possible).  
> -----  
> *Evidence–Based security is the security you get from using Identity Permissions????* This is a term that was used in a microsoft document – so I didn't just invent it.  
> -----  
> *Imperative and Declarative Permission Requests or*  
> *Imperative and Declarative Permission Styles or*  
> *Imperative and Declarative Security Statements are ways of requesting that the current assembly (its evidence) and/or user and/or granted permissions (by CLR – using evidence and security policy) have certain "permissions". Can Declarative Security Statements make use of all of the permission classes? I.E. can I specify that the access to the C drive is restricted to:*  
> *– a particular user or a user belonging to a particular role (i.e. role–based permission)*  
> *– an assembly that has access to the entire file system (as granted by the CLR using evidence and security policy) (i.e. code–access permission)*  
> *– an assembly that has an assembly evidence object of type Developer (this would be a customized evidence object) with the name "John Smith" (yes I'm aware this is easily falsified) (i.e. identity–based permission).*  
> -----  
>  
> *Can someone try to sift through the information I've stated above and confirm/deny the questions and change any inaccuracies.*  
>  
> *Thanks,*  
> *Novice*