

Re: Getting logged in user from a service?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-06/msg00867.html>

- *From:* "Larry Smith" <no_spam@xxxxxxxxxxxxx>
 - *Date:* Tue, 5 Jun 2007 20:26:03 -0400
-

... WMI will simply ... add the privilege (to it's own token)when needed on a per call basis.

That would be surprising considering that privileges can't be added to an existing token since tokens are mostly immutable. There is no function that does this. You need to add it to an existing account first and create a new logon session. At least that's the way Windows security has always worked assuming MSFT hasn't made any radical changes lately (which would greatly surprise me in this area). Of course MSFT can always make their own code do anything it wants but I don't see why they would breach their own security.

Some classes and methods need an impersonation token from the base client, if the token holds a needed privilege to execute or access a namespace class, WMI enables this privilege, when the token misses the privilege, the call fails.

That way it's possible to perform some administrative tasks without you, the client, to run as an administrator, you are simply delegating the task to another more privileged process, which on it's turn is rather restricted.

Look back at what Nicholas proposed, get the Handle of another process, if you don't run in the same logon session, you won't be able to access to process object to obtain the Handle (that is, Win32 OpenProcess will return access denied) unless you are running as administrator and even then, you won't probably be able to get all handles, there is nothing you can do without a process handle, game over. This is not an issue when using WMI (or System.Management, this is .NET right!).

An administrator can do anything it wants including accessing all handles. It just has to enable the SeDebugPrivilege privilege in its own token. I don't follow the other details however. If a program has the correct privilege(s) to do something then it doesn't need to delegate it to someone else. It can just do it on its own though it may have to enable a given privilege first (a simple task). If by delegating however you mean allowing WMI to perform the privileged operation using its own credentials (on the client's behalf), then this itself would be a a serious security violation

Re: Getting logged in user from a service?

by allowing an ordinary user to perform a privileged operation. I therefore don't see how WMI eliminates security issues for the programmer. It might simplify the housekeeping which is good and useful but you still need to consider the security environment you're running in (since WMI will fail if you access something you don't have permission for). This is no different than when you roll your own code without WMI.

Just try this as non administrator,

```
Process [] procs= Process.GetProcesses();
foreach(Process pro in procs)
{
try {
IntPtr handle = pro.Handle;
}
catch( Exception ex){Console.WriteLine(ex);}
}
and watch all:
System.ComponentModel.Win32Exception: Access is denied
at System.Diagnostics.ProcessManager.OpenProcess(Int32 processId, Int32
access, Boolean throw
IfExited)
thrown on you ...
```

This is exactly why so many programs run in the "administrators" account these days, worse some have "SeDebugPrivilege" enabled, talking about a surface attack.

Only administrators normally have SeDebugPrivilege by default though it's not enabled by default (which is the reason you can't end another session's process from the task manager for instance, even as an administrator). The administrator can easily enable this privilege on-the-fly however. In fact, if you do this using some utility for instance (which is easily written yourself), you'll find that you can end another session's process from the task manager. In any case, a rogue process that compromises the administrator account can do whatever it wants. so the SeDebugPrivilege is irrelevant (since it can easily be enabled if required)