Elevating process privilege programmatically?

Asked 16 years, 3 months ago Modified 10 months ago Viewed 249k times

179

I'm trying to install a service using InstallUtil.exe but invoked through `Process.Start`. Here's the code:

```
ProcessStartInfo startInfo = new ProcessStartInfo (m_strInstallUtil,
strExePath);
System.Diagnostics.Process.Start (startInfo);
```

where `m_strInstallUtil` is the fully qualified path and exe to "InstallUtil.exe" and `strExePath` is the fully qualified path/name to my service.

Running the command line syntax from an elevated command prompt works; running from my app (using the above code) does not. I assume I'm dealing with some process elevation issue, so how would I run my process in an elevated state? Do I need to look at `ShellExecute` for this?

This is all on Windows Vista. I am running the process in the VS2008 debugger elevated to admin privilege.

I also tried setting `startInfo.Verb = "runas";` but it didn't seem to solve the problem.

c# .net windows windows-services process-elevation

Share
Improve this question
Follow

edited Mar 23, 2017 at 15:24
Cœur
38.6k ● 26 ● 202 ● 276

asked Sep 25, 2008 at 13:38
Scott Marlowe
8,025 ● 12 ● 46 ● 52

1 After I have added `startInfo.UseShellExecute = true;` in addition to `startInfo.Verb = "runas";` it worked fine for me. – Matt Feb 17, 2020 at 15:16

7 Answers

Sorted by: Highest score (default)

205

You can indicate the new process should be started with elevated permissions by setting the Verb property of your startInfo object to 'runas', as follows:

```
startInfo.UseShellExecute = true;
startInfo.Verb = "runas";
```

This will cause Windows to behave as if the process has been started from Explorer with the "Run as Administrator" menu command.

This does mean the UAC prompt will come up and will need to be acknowledged by the user: if this is undesirable (for example because it would happen in the middle of a lengthy process), you'll need to run your entire host process with elevated permissions by Create and Embed an Application Manifest (UAC) to require the 'highestAvailable' execution level: this will cause the UAC prompt to appear as soon as your app is started, and cause all child processes to run with elevated permissions without additional prompting.

Share
Improve this answer
Follow

edited Feb 5 at 13:31
Hans Passant
940k ● 148 ● 1.7k ● 2.6k

answered Sep 25, 2008 at 13:57
mdb
52.8k ● 11 ● 66 ● 62

11 "runas" didn't work for me either. Might be that it only works with UAC turned off? – Dirk Vollmar Apr 3, 2009 at 12:43

20 @LukePuplett I'd thank them. It's a great way to mitigate root-kit installation without actually segregating the user roles completely. – Colton Mar 11, 2013 at 16:32

6 @Sparksis, but there's also another point of view. The people who know how to defend their OS, they know how to do it without UAC. And dummy users, who don't know what's defence, they are clicking 'YES' everytime when they see UAC window. Also nothing can prevent bad-hackers-virus-writers from using exploits to bypass it;) So I agree with LukePuplett =) – Jet Mar 27, 2013 at 20:14

35 Looks like you have to set `startInfo.ShellExecute=true` as well for this to work ... also I haven't been able to get this method to work with executable files that are on a network share (have to copy them to local temp directory before running) ... – TCC Sep 27, 2013 at 14:53

12 @Jet Sorry but that's the wrong mindset. Every time a UAC dialog is shown during install, that's a failing of the system. Where it's valuable is when you see a UAC box without running something requiring elevation. Doesn't matter how pro you - any of us - think we are on our PCs, you can't magically block zero days/drive-by downloads (eg a recent one on the official Nobel Prize site). If you browsed to that site and got a UAC prompt, you'd know there was something wrong. With UAC off, you'd never know you just joined a botnet. The cost of the advance warning is having to click Yes occasionally – Basic Mar 25, 2015 at 9:49

60

This code puts the above all together and restarts the current wpf app with admin privs:

```
if (IsAdministrator() == false)
{
    // Restart program and run as admin
    var exeName =
System.Diagnostics.Process.GetCurrentProcess().MainModule.FileName;
    ProcessStartInfo startInfo = new ProcessStartInfo(exeName);
    startInfo.Verb = "runas";
    System.Diagnostics.Process.Start(startInfo);
    Application.Current.Shutdown();
    return;
}

private static bool IsAdministrator()
{
    WindowsIdentity identity = WindowsIdentity.GetCurrent();
    WindowsPrincipal principal = new WindowsPrincipal(identity);
    return principal.IsInRole(WindowsBuiltInRole.Administrator);
}


// To run as admin, alter exe manifest file after building.
// Or create shortcut with "as admin" checked.
// Or ShellExecute(C# Process.Start) can elevate - use verb "runas".
// Or an elevate vbs script can launch programs as admin.
// (does not work: "runas /user:admin" from cmd-line prompts for admin pass)
```

Update: The app manifest way is preferred:

Right click project in visual studio, add, new application manifest file, change the file so you have requireAdministrator set as shown in the above.

A problem with the original way: If you put the restart code in app.xaml.cs OnStartup, it still may start the main window briefly even though Shutdown was called. My main window blew up if app.xaml.cs init was not run and in certain race conditions it would do this.

Share

Improve this answer

Follow

edited Aug 14, 2012 at 21:13

answered Jun 5, 2012 at 22:12

Curtis Yallop

7,279 ● 3 ● 51 ● 37

See below for improvements. – JCCyC Sep 10, 2014 at 22:20

requireAdministrator did not work in my case. So I had to do it your way. This solved my problem ! thank you. But I had to set Single Instance Application to false. – chris Nov 18, 2015 at 3:16

This worked for me. I also passed the original `string[] args` to the respawned process. – DotNetPadawan Jan 25, 2019 at 20:02

1 Good example. On .NET 5 WinForms, you must include the line startInfo.UseShellExecute = true; – A. Niese May 21, 2021 at 22:59

And you may want to put Process.Start in a try..catch in case the user clicks 'no' to the UAC dialog. Otherwise you may experience a hang: stackoverflow.com/questions/23350175/… – A. Niese May 22, 2021 at 3:26

24

According to the article [Chris Corio: Teach Your Apps To Play Nicely With Windows Vista User Account Control, MSDN Magazine, Jan. 2007](), only `ShellExecute` checks the embedded manifest and prompts the user for elevation if needed, while `CreateProcess` and other APIs don't. Hope it helps.

See also: [same article as .chm]().

Share

Improve this answer

Follow

edited Oct 10, 2018 at 16:43
Kuba hasn't forgotten Monica
98.2k ● 17 ● 164 ● 327

answered Oct 23, 2008 at 23:56
Yevgeniy Shapiro

8

```
[PrincipalPermission(SecurityAction.Demand, Role = @"BUILTIN\Administrators")]
```

This will do it without UAC - no need to start a new process. If the running user is member of Admin group as for my case.

Share

Improve this answer

Follow

edited Jun 6, 2019 at 13:25
mhu
18k ● 10 ● 65 ● 94

answered Jan 12, 2012 at 8:43
hB0
2,037 ● 1 ● 30 ● 33

5 Using this code I get an permission error: "Request for security entity permission failed." :(– Leonardo Sep 21, 2015 at 18:57

1 Perhaps "If the running user is memeber of Admin"* – hB0 Sep 22, 2015 at 9:30

1 interesting - this can go on any method, not just an action method (I even tried it on method defined in an ASPX page) – Simon_Weaver Apr 17, 2017 at 10:22

1 In Microsoft's documentation, as of 2023, it notes that "Code Access Security (CAS) has been deprecated across all versions of .NET Framework and .NET. Recent versions of .NET do not honor CAS annotations and produce errors if CAS-related APIs are used. Developers should seek alternative means of accomplishing security tasks." – nwsmith Feb 23, 2023 at 18:52

2 @hB0 The way to get the executable to require to run with Administrator credentials is to apply a '.manifest' to the '.exe' as a post-build step. The way I choose to do this was using Microsoft's 'mt.exe' utility, following the instructions [here](). However if your using Visual Studio, I believe you can automate this step. – nwsmith Mar 21, 2023 at 15:58

6

i know this is a very old post, but i just wanted to share my solution:

```
System.Diagnostics.ProcessStartInfo StartInfo = new
System.Diagnostics.ProcessStartInfo
{
    UseShellExecute = true, //<- for elevation
    Verb = "runas",  //<- for elevation
    WorkingDirectory = Environment.CurrentDirectory,
    FileName = "EDHM_UI_Patcher.exe",
    Arguments = @"\D -FF"
};
System.Diagnostics.Process p = System.Diagnostics.Process.Start(StartInfo);
```

NOTE: If VisualStudio is already running Elevated then the UAC dialog won't show up, to test it run the exe from the bin folder.

Share Improve this answer Follow

answered Jan 5, 2022 at 20:56

BlueMystic
2,277 ● 27 ● 25

Exception thrown: `'System.InvalidOperationException'` in System.dll Exception message: The Process object must have the `UseShellExecute` property set to `false` in order to redirect IO streams. – Art Hansen Mar 30 at 13:07

3

You should use Impersonation to elevate the state.

```
WindowsIdentity identity = new WindowsIdentity(accessToken);
WindowsImpersonationContext context = identity.Impersonate();
```

Don't forget to undo the impersonated context when you are done.

Share

Improve this answer

Follow

edited Dec 21, 2010 at 12:33

Leon Bambrick
26.3k ● 9 ● 52 ● 76

answered Sep 25, 2008 at 13:53

Vijesh VP
4,540 ● 6 ● 32 ● 32

34 You haven't said how to get the accessToken. LogonUser will provide a user's restricted security context when UAC is enabled. – cwa Mar 30, 2011 at 18:45

If you're using VSTS, you can get a Personal Access Token from the Settings in the Web Portal. (Look for settings icon next to the user picture.) MSDocs – Su Llewellyn Feb 5, 2020 at 23:42

1

There is an alternative way of starting a process that requires elevation without having to have elevation in the parent process using the windows `Task Scheduler`. Basically you manually create a task and check `Run with highest privileges` in the `General` tab. This task can be run programmatically using a Task Scheduler library.

Note: It is not possible to dynamically create such a task as this would require your process to be elevated.

Using the TaskScheduler library finding and running the task looks like this:

```
TaskService.Instance.RootFolder.AllTasks
        .FirstOrDefault(x => x.Name == "My elevated Task").Run();
```

Or, instead of creating the task manually you can create it programmatically (requires elevation once) and later start it without.

Share

Improve this answer

Follow

edited May 4, 2023 at 13:01

answered May 4, 2023 at 9:26

henon

2,471 ● 23 ● 25