Microsoft® Virtual Labs

Windows 7: UAC Data Redirection: .Net Framework

Microsoft

Table of Contents

Windows 7: UAC Data Redirection: .Net Framework 1
Exercise 1 Exploring User Account Control Virtualization 2
Exercise 2 Exploring User Account Control Virtualization 10

Windows 7: UAC Data Redirection: .Net Framework

Objectives

After completing this lab, you will be better able to:

- Troubleshoot a file redirection issue
- Use Process Monitor to find the root cause of the issue

Scenario

Many applications are still designed to write files to the Program Files, Windows® directories, or system root (typically the C drive) folders. Some applications are designed to update Microsoft® Windows registry values, specifically values in HKLM/Software. But there is one problem: the files or registry values are not created or updated. In this lab, you will experience first hand the effects of UAC virtualization and will walk through the steps to solve the problem.

Estimated Time to Complete This Lab

90 Minutes

Computer used in this Lab

Win7Devs

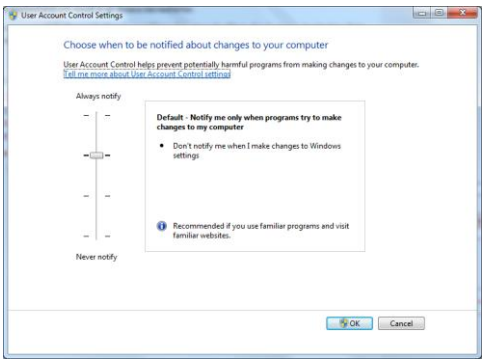
The username for the Administrator account on this computer is **Win7User** and the password is: **pass@word1**

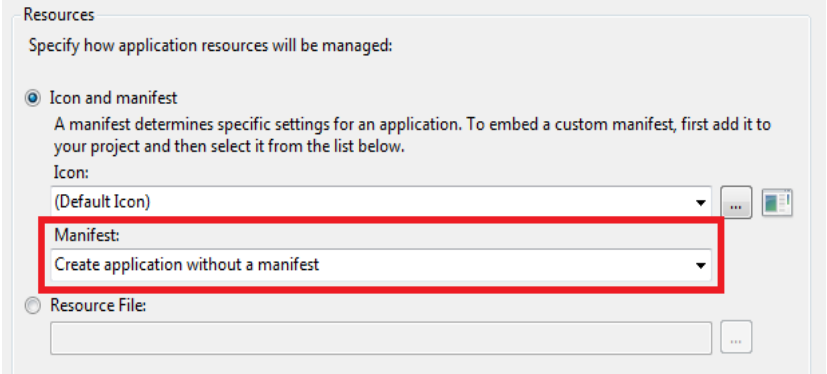
Exercise 1
Exploring User Account Control Virtualization

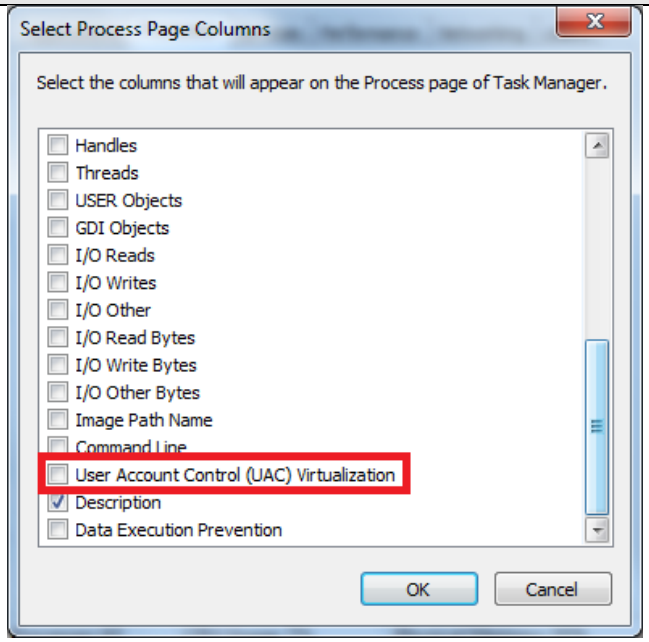
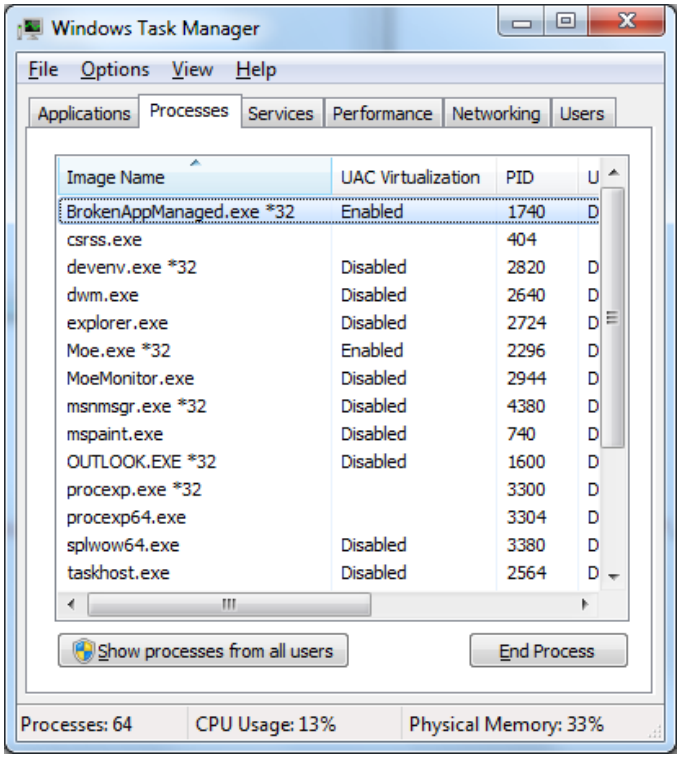
Scenario

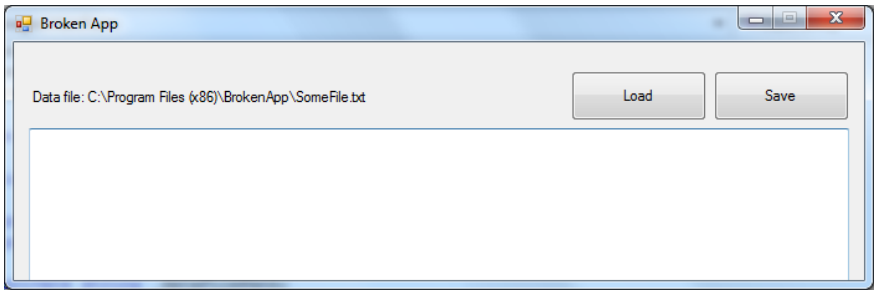
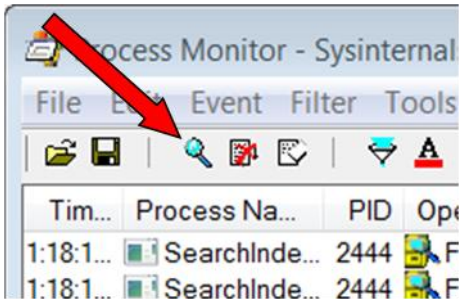
In this exercise, you will diagnose a managed (.NET Framework) application that is incompatible with Windows 7and exhibits User Account Control (UAC) virtualization, specifically, file redirection, to the VirtualStore folder.

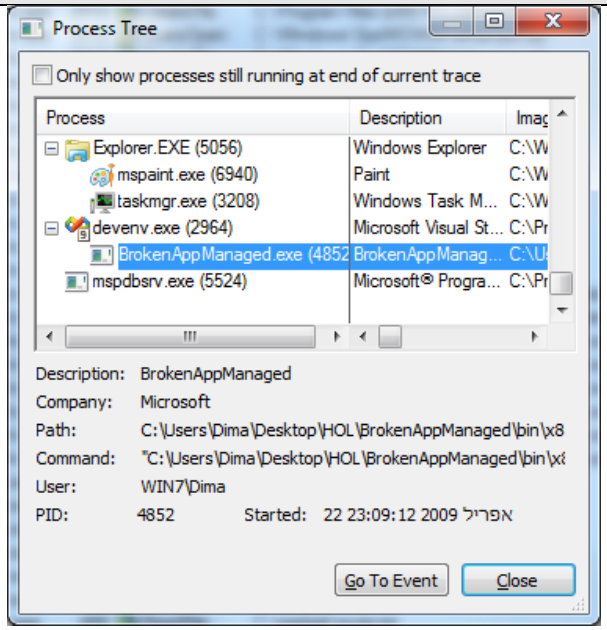
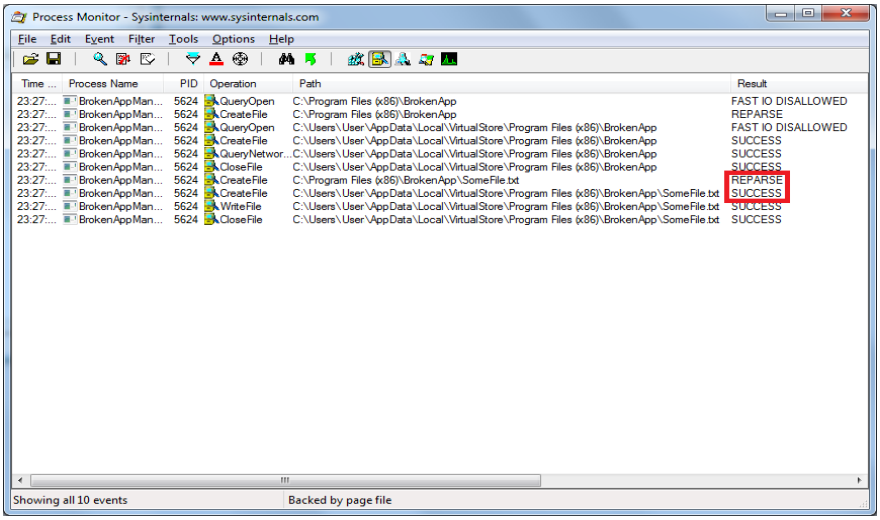
You will start by identifying the problem through a series of tests. Next, you will add a UAC manifest section and compile the application, thereby marking it as UAC-aware. As you run the application, you will notice that UAC does not virtualize your files when writing to any of the protected folders, such as Program Files; rather, the operation is blocked and results in an “access denied” error. Finally, you will fix the application by modifying it to store the file to the Application Data (ProgramData) folder.

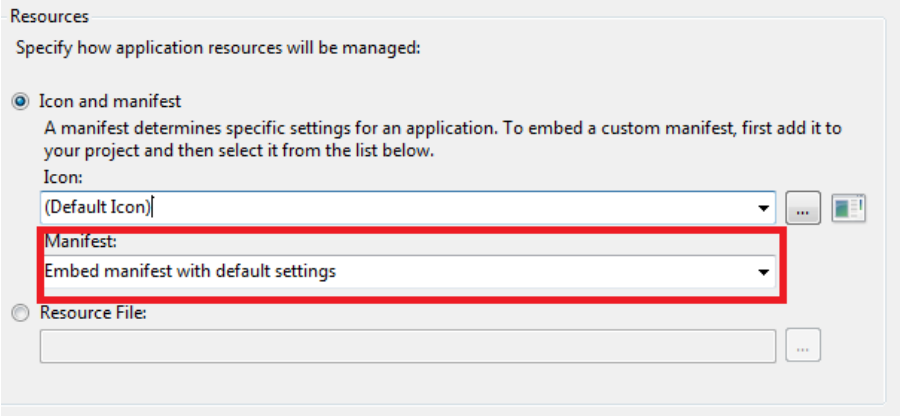
Tasks	Detailed Steps
Complete the following task on: **Win7Devs** **1. Run the Application without a Manifest**	***Note:*** *In this task, you will run the application without a manifest, which simulates an older application and triggers the UAC virtualization mechanism. New applications created with Visual Studio 2008 automatically embed a manifest containing a UAC section by default.* **a.** Make sure UAC is enabled. **b.** From the **Start** menu: **c.** Open Search **d.** Type “UAC” **e.** Click "Change User Account Control settings" in the search results list. **f.** The **User Account Control Settings** dialog box appears. To ensure UAC is NOT disabled: **g.** Set the UAC slider at the default level (as pictured) **h.** Click **OK**.  **i.** Open **Windows Explorer** and navigate to the folder containing the solution (**C:\Labs\Managed\UAC Redirection\Begin**)

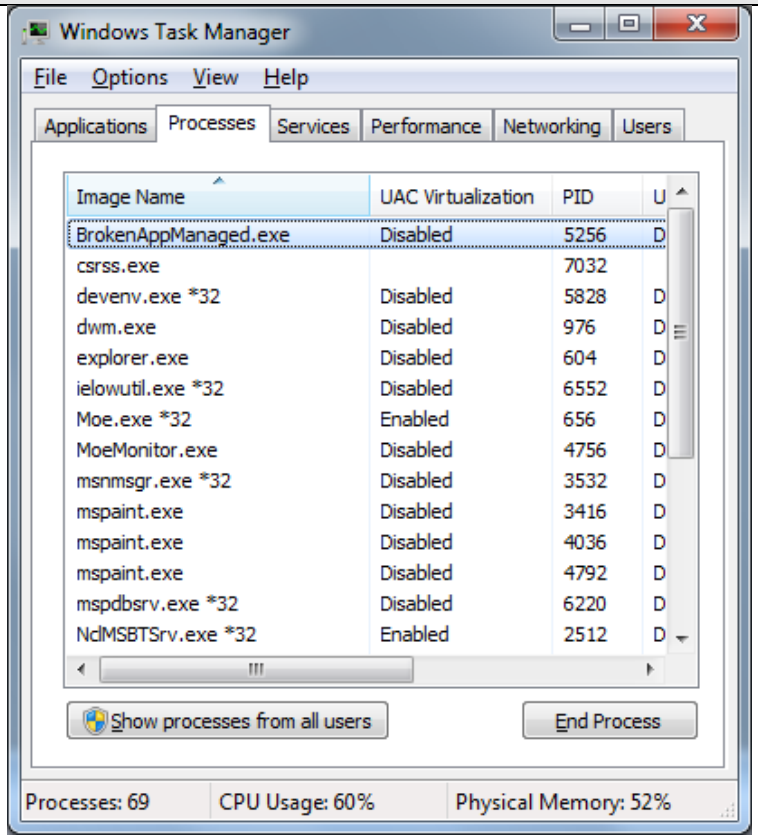
Tasks	Detailed Steps
	j. Double-click the "**DataRedirection**" solution file to open the solution. ***Note:*** *Make sure you* ***don't*** *start Visual Studio with* ***Administrator*** *privileges. If Visual Studio is started with elevated privileges, then "Visual Studio (Administrator)" will display in the title bar* **k.** In the toolbar, set the target architecture to **x86**:  **l.** Right-click the **BrokenAppManaged** project and select **Properties**. ***Note:*** *Configure the project to build the application without a manifest (the default manifest contains a UAC section)* **m.** In the **Application** tab, under **Manifest**, verify "**Create application without a manifest**" is selected.  **n.** Close the **Properties** window. **o.** In **the Solution Explorer** pane, double-click on **FileIO.cs** to open it. **p.** Navigate to the **FileIO** class. **q.** Inspect the **Save** and **Load** functions and the static constructor and observe how the data file path is constructed as a file under Program Files\BrokenApp. **r.** Build the project and run it by pressing the **F5** key. **s.** Open Task Manager and click the **Processes** tab. **t.** From the **View** menu, choose **Select Columns** **u.** The **Select Process Page Columns** dialog box appears **v.** Check the **User Account Control (UAC) Virtualization** check box **w.** Click **OK**

Tasks	Detailed Steps
	 **x.** Notice that the **UAC Virtualization** column is enabled for your process.  **y.** Type some text into the edit box and then click **Save**. The operation should succeed (that is, you won't receive any error).

Tasks	Detailed Steps
	 **z.** Try to navigate to the path indicated (C:\Program Files\BrokenApp). ***Note:*** *You won't find BrokenApp under Program Files because the write file operation was redirected to the VirtualStore folder*
2. Find the Problem	***Note:*** *In this task, you will walk through the different steps that will confirm that your application is experiencing UAC virtualization.* ***Note:*** *In this task, we will use the Process Monitor application. This utility is available for download from Microsoft TechNet (http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx.* **a.** From the **Start** menu, launch **Process Monitor**. Click **Yes** for the **User Account Control** dialog box. **b.** Click **Cancel** in the **Process Monitor Filter** dialog box. **c.** In the menu bar, click **Filter** -> **Reset Filter**. **d.** Ensure **Process Monitor** is capturing events by verifying the third toolbar button is not crossed out. You can also toggle capturing on/off by pressing **CTRL-E**.  **e.** In the **BrokenAppManaged** application, click **Save** again. **f.** Stop capturing in **Process Monitor** by clicking the third toolbar button (or by pressing **CTRL-E**). **g.** In **Process Monitor**, from the **Tools** menu, click **Process Tree**. The **Process Tree** dialog box appears.

Tasks	Detailed Steps
	 **h.** Look for **BrokenAppManaged.exe** in the tree and double-click it. **i.** Click **Close** to close the **Process Tree** dialog box. **j.** Right-click the process name **BrokenAppManaged** in Process Monitor. **k.** Select **Include -> Process Name**. This will filter out all other events.  ***Note:*** *You can see that BrokenAppManaged is trying to create the file C:\ProgramFiles \BrokenApp\SomeFile.txt. This file is redirected to the VirtualStore folder, where the actual data file ends up.* *Notice the Result column. The line where the result is "REPARSE" is the original operation. The next line with the result "SUCCESS" is the redirected operation.* **l.** Close the running **BrokenAppManaged** application.

Tasks	Detailed Steps
3. Add a UAC Manifest	***Note:*** *In this task, you will add a manifest to the application to mark the application as UAC-aware. By marking your application as UAC aware, you declare that the application does not require write access to protected locations. UAC virtualization will not apply to your application.* **a.** In **Visual Studio**, right-click the project in **Project Explorer** and select **Properties**. **b.** Right-click the **BrokenAppManaged** project and select **Properties**. **c.** Configure the project to build the application with a manifest. **d.** Under **Manifest**, select "**Embed manifest with default settings**" ***Note:*** *The default manifest will contain a UAC section*  **e.** **Save** the change and close the **Properties** window. **f.** Press the **F5** key to run the application. **g.** Look at **Task Manager** again, and you will notice that virtualization is now disabled for the process:

Tasks	Detailed Steps
	 **h.** Type some text into the edit box and click Save in the BrokenAppManaged application. You should receive an error dialog box. ***Note:*** *Because UAC virtualization is turned off, writing to protected locations results in an error.* **i.** Close the running **BrokenAppManaged** application.
4. Correct the Access Denied Error	***Note:*** *By embedding the manifest containing a UAC section, you declare to Windows 7 that your application is UAC-aware; therefore the application will be unable to write to any protected storage area. In this task, you will change the location to which the text file will be saved and fix the access denied error.* **a.** Return to **Visual Studio**. **b.** In the static constructor of the **FileIO** class in FileIO.cs: **c.** Comment out the line to save to Environment.SpecialFolders.ProgramFiles (Line 49) **d.** Uncomment the line to save to Environment.SpecialFolders.CommonApplicationData (Line 50) **e.** This will save the file to the common program data folder (C:\ProgramData, by default), which is accessible by all users. **f.** Press the **F5** key to rebuild and run the application. **g.** Type some text into the edit box and then click **Save**. The operation should

Tasks	Detailed Steps
	succeed. **h.** Make sure the data file is now saved to the correct folder: **C:\ProgramData\BrokenApp**. ***Note:*** *You now have fixed the redirection issue and saved your data file to the correct location.* *In order for redirection to work in Visual Studio 2008, you must turn off UAC in the manifest generation. To do so:* *For C# projects In Visual Studio:* • *Click the Project menu.* • *Click the Properties for that project.* • *On the Application tab, in the Resources area, select the Icon and manifest button.* • *Select Create application without a manifest.* • *Click OK.* *UAC is turned off here only for demonstration purposes. All executables should contain a UAC section in the manifest. If a UAC section is present in the manifest, Windows will not consider the application a legacy application and does not enable redirection. Writing to Program Files would return an access denied error.*

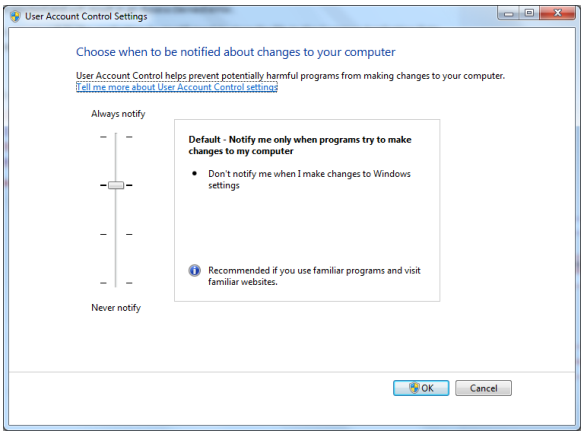
Exercise 2
Exploring User Account Control Virtualization

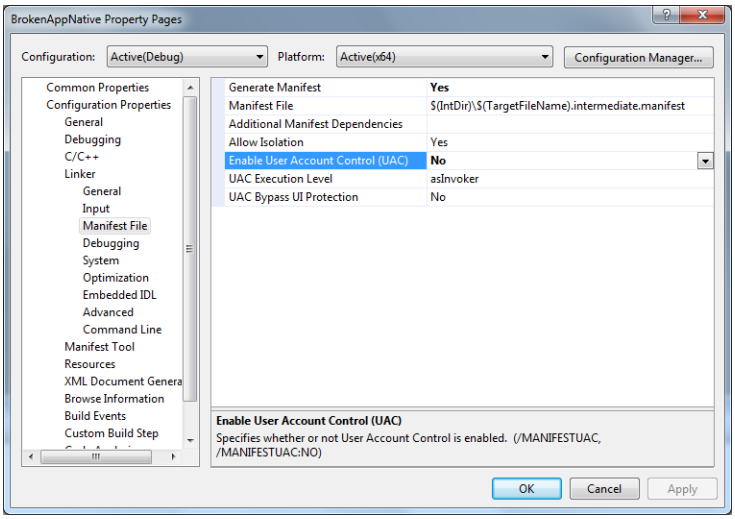
Scenario

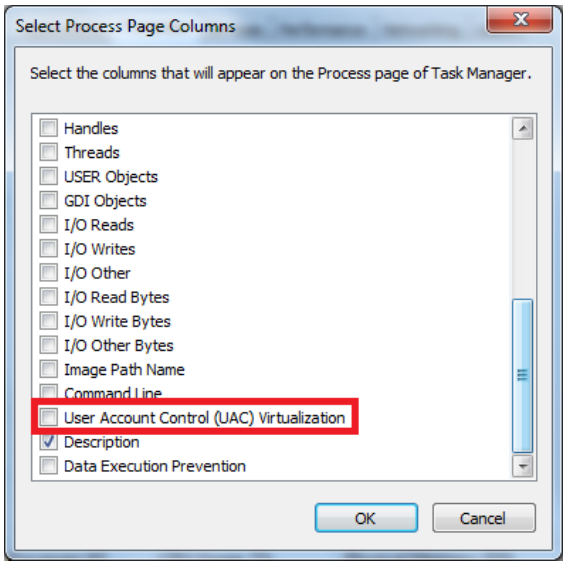
In this exercise, you will diagnose a broken native (Win32) C++ applicationthat exhibits file redirection to the VirtualStore folder.

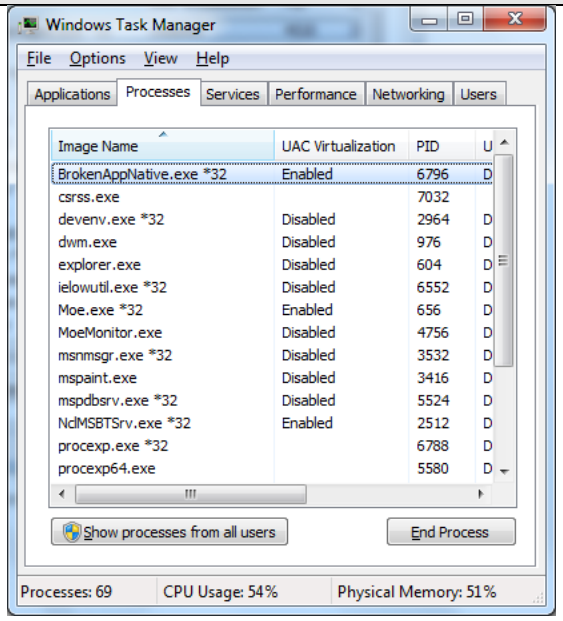
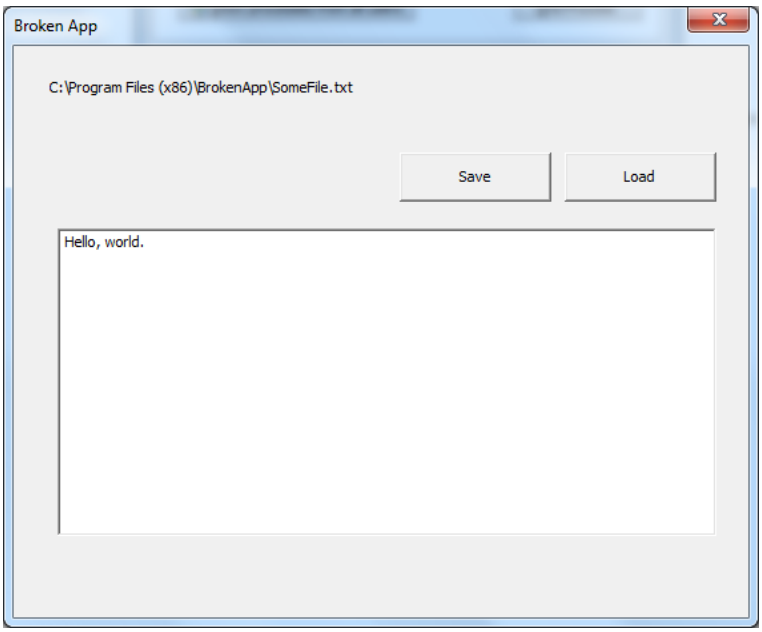
You will then add a UAC manifest section and compile the application , thereby marking it as UAC-aware. You will observe that instead of being redirected, write operations to the Program Files folder will be blocked resulting in an “access denied” error.

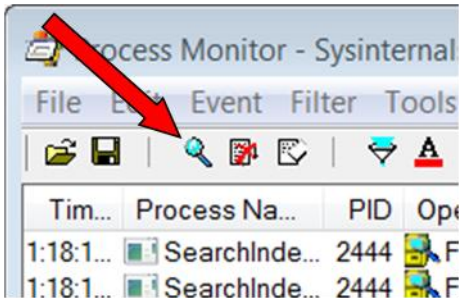
Finally, you will fix the application by modifying it to store the file to the Application Data (ProgramData) folder.

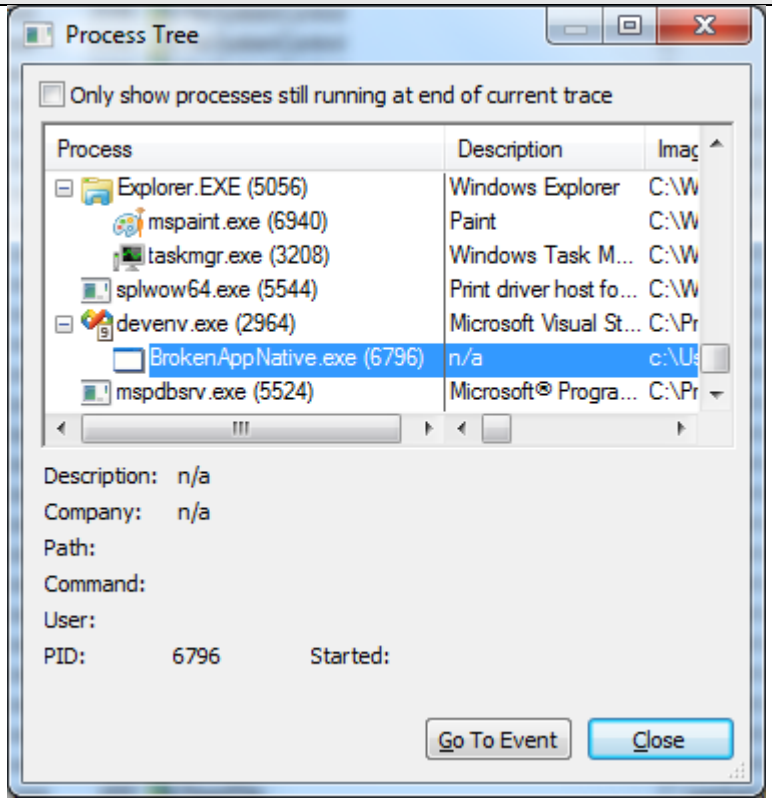
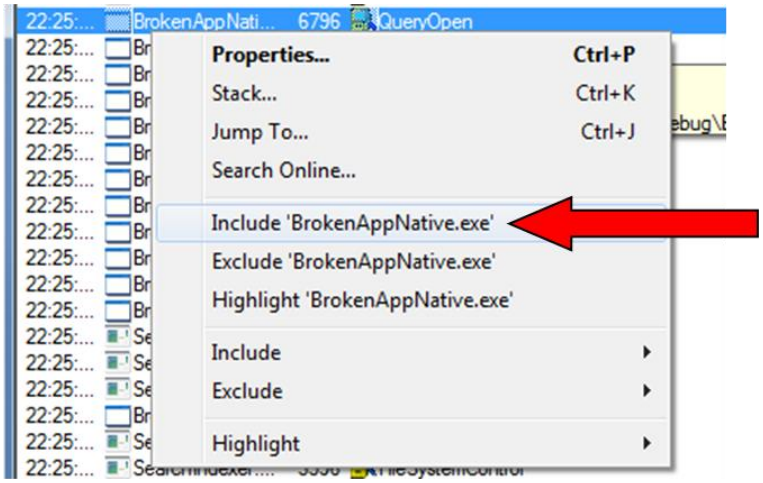
Tasks	Detailed Steps
Complete the following task on: **Win7Devs** **1. Run the Application without a Manifest**	***Note:*** *In this task, you will run the application without a manifest, which simulates an older application and triggers the UAC virtualization mechanism. New applications created with Visual Studio 2008 automatically embed a manifest containing a UAC section by default.* **a.** Make sure UAC is enabled **b.** From the **Start** menu • Open Search • Type **UAC** • Click “Change User Account Control settings” in the search results list **c.** The **User Account Control Settings** dialog box appears. To ensure UAC is NOT disabled: • Set the UAC slider at the default level (as pictured below) • Click **OK** 

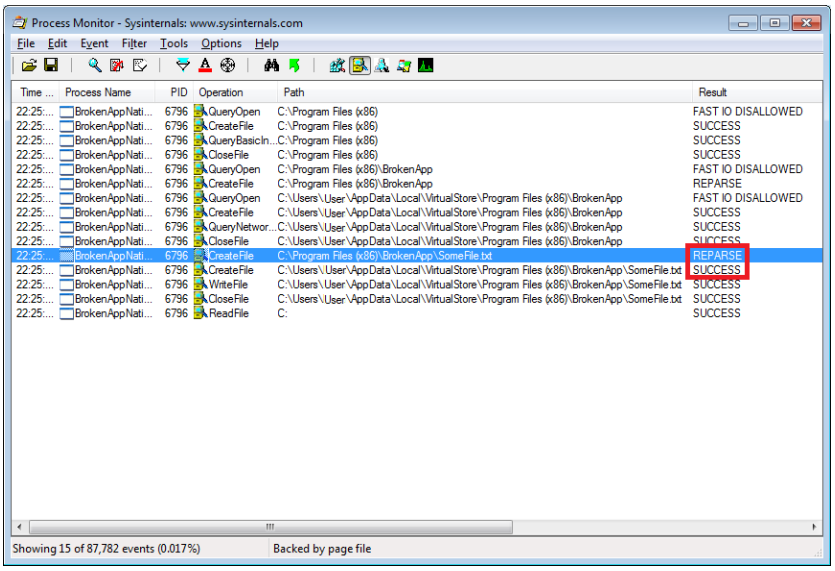
Tasks	Detailed Steps
	d. Open **Windows Explorer** and navigate to the folder containing the solution (**C:\Labs\Native\UAC Redirection\Begin**) **e.** Double-click the "**DataRedirection**" solution file to open the solution. ***Note:*** *Make sure you* ***don't*** *start Visual Studio with* ***Administrator*** *privileges. If Visual Studio is started with elevated privileges, then "Visual Studio (Administrator)" will display in the title bar* **f.** In the toolbar, set the build target to x86:  ***Note:*** *The reason for this is that for x64 applications, virtualization is turned off regardless of a manifest.* **g.** Right-click the **BrokenAppNative** project in Solution Explorer and select **Properties**: **h.** By default, Visual Studio 2008 configures projects to include a UAC section in the manifest; we will turn off this setting to illustrate how a legacy application would behave **i.** In **Properties**: **j.** Under **Configuration Properties**, expand the **Linker** node **k.** Select **Manifest File** **l.** Ensure the "**Enable User Account Control (UAC)**" selection is set to **No** **m.** Click **OK**  **n.** Close the **Properties** window. **o.** In **Solution Explorer**, double-click on the **BrokenAppNative.cpp** file to open it. **p.** Inspect the **SaveFile** and **LoadFile** functions **q.** Observe how the path is constructed in the **MakeDataFilePath** function:

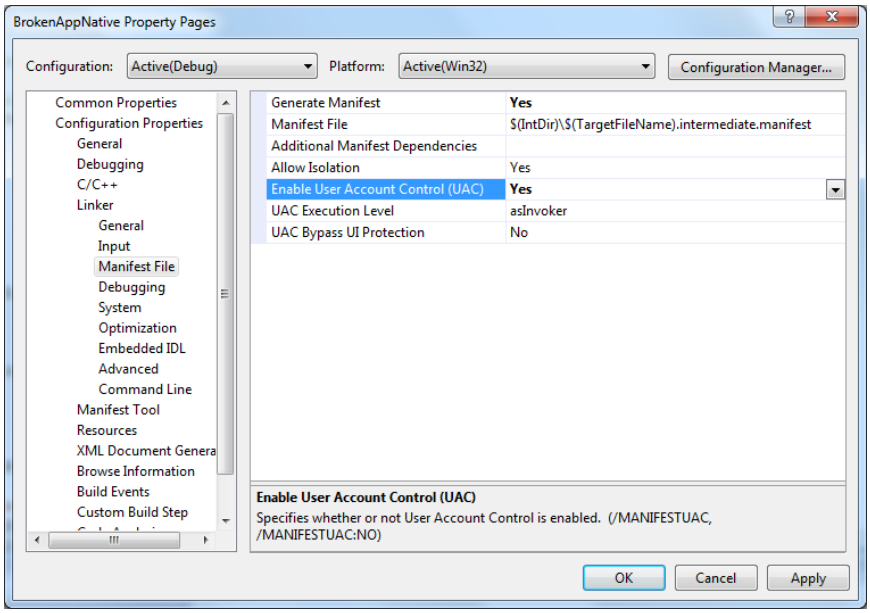
Tasks	Detailed Steps
	r. The data file path saves to a folder under Program Files • **SHGetKnownFolderPath** with the **FOLDERID_ProgramFiles** parameter retrieves the path of the Program Files folder • A subdirectory under Program Files is created first (BrokenApp), if it doesn't exist • Then a filename (SomeFile.txt) is created under that subdirectory **s.** Build the project. **t.** Press the **F5** key to run it. **u.** Open **Task Manager** and click the **Processes** tab. • From the **View** menu, choose **Select Columns** • The **Select Process Page Columns** dialog box appears • Check the **User Account Control (UAC) Virtualization** check box • Click **OK**  **v.** Notice that the **UAC Virtualization** column is enabled for your process:

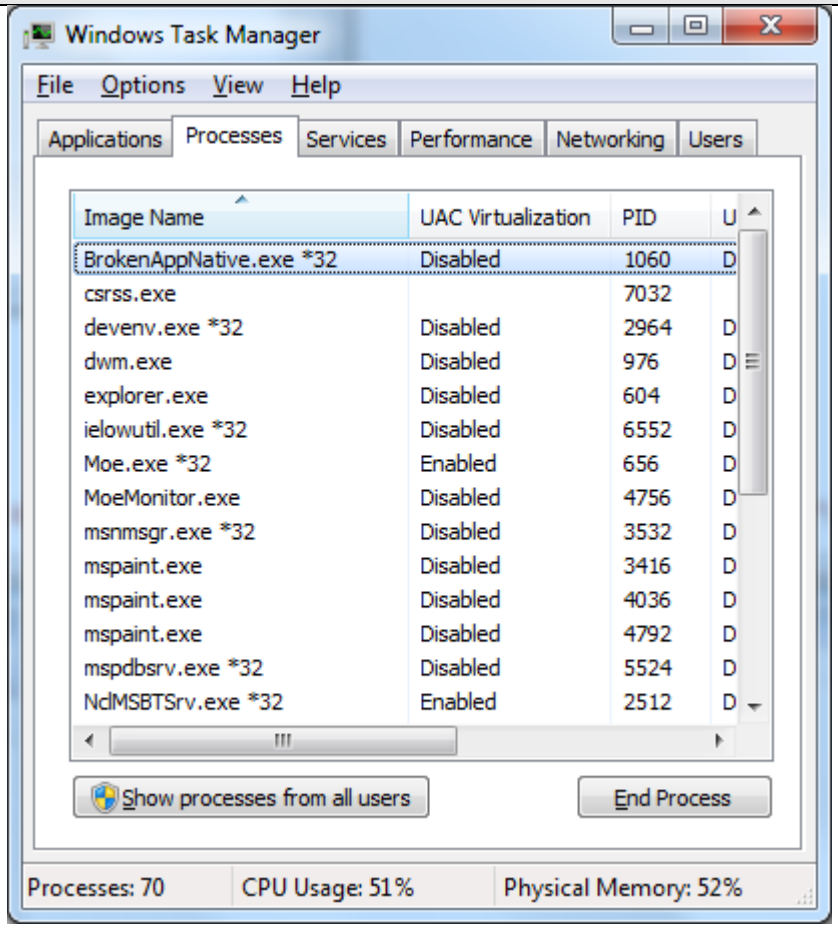
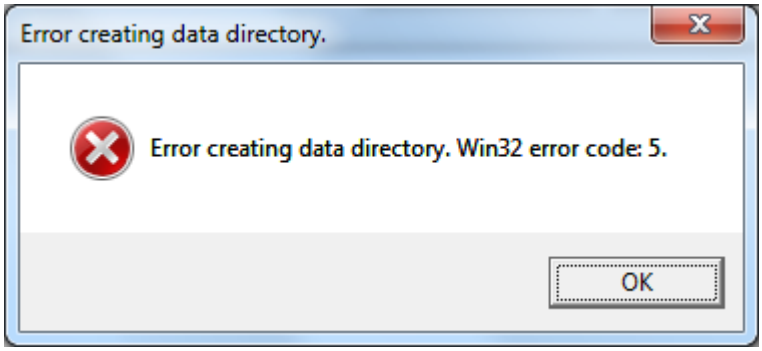
Tasks	Detailed Steps
	 **w.** Type some text into the edit box and then click **Save**. The operation should succeed; that is, you won't receive an error.  **x.** Try to navigate to the path indicated (**C:\Program Files\BrokenApp**). **y.** Close the running **BrokenAppNative** application. ***Note**: You won't find BrokenApp under Program Files because the write file operation was redirected to the VirtualStore folder*
2. Find the Problem	***Note:** In this task, you will walk through the different steps to confirm that your application is experiencing UAC virtualization.*

Tasks	Detailed Steps
	In this task, we will use the Process Monitor application. This utility is available for download from Microsoft TechNet (http://technet.microsoft.com/en-us/sysinternals/bb896645.aspx. **a.** From the **Start** menu, launch **Process Monitor**. Click **Yes** for the **User Account Control** dialog box. **b.** Click **Cancel** in the **Process Monitor Filter** dialog box. **c.** In the menu bar, click **Filter** -> **Reset Filter**. **d.** Ensure **Process Monitor** is capturing events by verifying the third toolbar button is not crossed out. You can also toggle capturing on/off by pressing **CTRL-E**.  **e.** In the **BrokenAppNative** application, click **Save** again. **f.** Stop capturing in **Process Monitor** by clicking the third toolbar button (or by pressing **CTRL-E**). **g.** In **Process Monitor**, from the **Tools** menu, click **Process Tree**. The **Process Tree** dialog box appears:

Tasks	Detailed Steps
	 **h.** Look for **BrokenAppNative.exe** in the tree and double-click it. **i.** Click **Close** to close the **Process Tree** dialog box. **j.** Right-click the process name **BrokenAppNative** in **Process Monitor**. **k.** Click **Include BrokenAppNative**. This will filter out all other events: 

Tasks	Detailed Steps
	 ***Note:*** *You can see that BrokenAppNative is trying to create the file C:\ProgramFiles \BrokenApp\SomeFile.txt. This file is redirected to the VirtualStore folder, where the actual data file ends up.* *Notice the Result column. The line where the result is "REPARSE" is the original operation. The next line with the result "SUCCESS" is the redirected operation.* **l.** Close the running **BrokenAppNative** application.
3. Add a UAC Manifest	***Note:*** *In this task, you will add a manifest to the application to mark the application as UAC-aware. By marking your application as UAC-aware, you declare that the application does not require write access to protected locations. UAC virtualization will not apply to your application.* **a.** In **Visual Studio**, right-click the **BrokenAppNative** project and select **Properties**. **b.** Under **Configuration Properties,** expand **Linker.** **c.** Select **Manifest File.** **d.** Change the "**Enable User Account Control (UAC)"** selection to **Yes.** **e.** Click **OK**.

Tasks	Detailed Steps
	 **f.** Re-build the application. **g.** Press the **F5** key to run the application. **h.** Look at Task Manager again, and you will notice that virtualization is now disabled:

Tasks	Detailed Steps
	 ***Note:*** *This is because the presence of the UAC section in the manifest marks the application as UAC-aware.* **i.** Type some text into the edit box and click **Save** in the **BrokenAppNative** application. You should receive the following error:  ***Note****: Because UAC virtualization is turned off, writing to protected locations results in an error.* **j.** Close the running **BrokenAppNative** application.

Tasks	Detailed Steps
4. Correct the Access Denied Error	***Note:*** *By embedding the manifest containing a UAC section, you declare to Windows 7 that your application is UAC-aware, and therefore will refrain from writing to any protected storage area. In this task, you will change the location to which the text file will be saved and fix the access denied error.* **a.** Return to **Visual Studio**. **b.** Navigate to the **MakeDataFilePath** function in **BrokenAppNative.cpp**. **c.** Comment out the line at the top of the function that includes FOLDERID_ProgramFiles (Line 63). **d.** Uncomment the line that includes FOLDERID_ProgramData (Line 62). **e.** Rebuild the application. **f.** Press the **F5** key to run the application. **g.** Type some text into the edit box and then click **Save**. The operation should succeed. **h.** Make sure the data file is now saved to the correct folder: **C:\ProgramData\BrokenApp**. ***Note:*** *You now have fixed the redirection issue and saved your data file to the correct location.* ***Note:*** *In order for redirection to work in Visual Studio 2008, you must turn off UAC in the manifest generation. To do so:* *For C++ projects In Visual Studio:* *1. Click the Project menu.* *2. Click the Properties for that project.* *3. Expand Configuration Properties.* *4. Expand Linker.* *5. Select Manifest File.* *5. Change the Enable User Account Control (UAC) selection to No.* *6. Click OK.* *UAC is turned off here only for demonstration purposes. All executables should contain a UAC section in the manifest. If a UAC section is present in the manifest, Windows will not consider the application a legacy application and does not enable redirection. Writing to Program Files would return an access denied error.*