Hey, Scripting Guy! How Can I Sign Windows PowerShell Scripts with an Enterprise Windows PKI? (Part 2 of 2)

ScriptingGuy1 June 17, 2010

16

Q Hey Scripting Guy! Will you give us the final steps in the step-by-step guide about how to use an existing Windows PKI installation to sign Windows PowerShell scripts, or were you just kidding yesterday?

--HR

A Hello HR,

Microsoft Scripting Guy Ed Wilson here. Just as we promised yesterday, we continue today with the final steps involved in Windows PKI to sign scripts. We go back to Ragnar Harper.

Step 3: Sign my Windows PowerShell script and run it

In this step we will be inside Windows PowerShell, and we will sign our script. For this purpose I have a simple script named demoscript.ps1.

```
$yourName=Read-Host "What is your name?"

Write-Host "Hello $yourName"
```

Just a quick reminder that your requirements for signed scripts are set using the **Set-ExecutionPolicy** cmdlet (or by Group Policy).

Setting	Description
Unrestricted	No requirements; all scripts allowed
RemoteSigned	All local scripts allowed; only signed remote scripts
AllSigned	All scripts need to be signed
Restricted	No scripts allowed

For this demonstration, my **executionpolicy** is set to **AllSigned**. If I just try to run my script, it will fail, as shown in the following image.

```
PS C:\> .\demoscript.ps1
File C:\demoscript.ps1 cannot be loaded. The file C:\demoscript.ps1 is not digitally signed. The script will not execut
e on the system. Please see "get-help about_signing" for more details..
At line:1 char:17
+ .\demoscript.ps1 <<<<
    + CategoryInfo          : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException
```

We will use the cmdlet **Set-AuthenticodeSignature** to sign the script. I will start storing the code signing certificate in a variable named **$cert**.

```
$cert=(dir cert:currentuser\my\ -CodeSigningCert)
```

Then I am ready to sign my script with the **Set-AuthenticodeSignature** cmdlet. This is shown in the following image.

```
PS C:\> $cert=(dir cert:currentuser\my\ -CodeSigningCert)
PS C:\> Set-AuthenticodeSignature .\demoscript.ps1 $cert -TimestampServer http://timestamp.comodoca.com/authenticode

    Directory: C:\

SignerCertificate                         Status                                 Path
-----------------                         ------                                 ----
AD69CCE84F7A44AAA61A2E69F8C5A7D8538F31C2  Valid                                  demoscript.ps1

PS C:\> _
```

As you see, the status is valid, so the signing was successfully done. Please note that I recommend that you supply the **TimeStampServer** parameter. This will make sure the script works even though the certificate that signed it is expired. It will tell the system that the code signing certificate was valid at the time of signing. (Okay, I can imagine there are some situations where this might not be correct, but I also guess it will be good enough for most of us.) If you do not use the **TimeStampServer** parameter, the script will stop to work when the certificate used for signing expires. There are multiple sources for timestamping out there. Use one that suits you.

Let us try to run the scripts again, and see what happens. The results are shown in the following image.

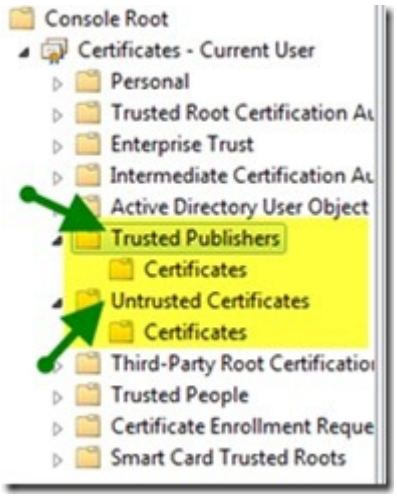
```
PS C:\Scripts> .\demoscript.ps1

Do you want to run software from this untrusted publisher?
File C:\Scripts\demoscript.ps1 is published by CN=Administrator, CN=Users, DC=harper, DC=labs and is not trusted on
your system. Only run scripts from trusted publishers.
[V] Never run  [D] Do not run  [R] Run once  [A] Always run  [?] Help (default is "D"): _
```

We get a question if we want to run the script or not. The question says that this is a script from an untrusted publisher. In Step 4, I will show you how to make the publisher (code signing certificate) trusted for your domain.

As for this computer, you can now make this publisher trusted by choosing **A** for **Always run**. If you choose **V** for **Never run**, you will explicitly make this publisher untrusted, and scripts signed by this certificate will not run.

Let's stop and see what exactly is happening here. If you make any choice persistent (such as **Always run** or **Never run**), the code signing certificate is stored as a trusted or untrusted publisher on your computer. You can see this through the GUI if you open mmc.exe and load the Certificates snap-in, as shown in the following image.



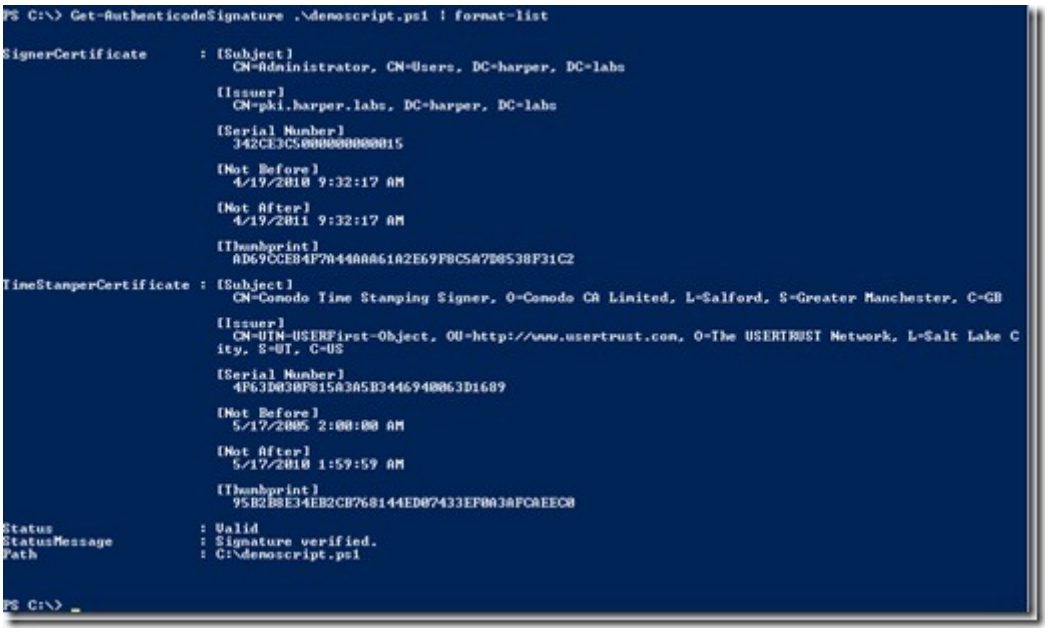
Or, you could also do this from Windows PowerShell:

dir cert:\CurrentUser\TrustedPublisher

dir cert:\CurrentUser\Disallowed

As you will see in Step 4, you can also control this setting through Group Policy. For now, you can just click **Run Once**, and the script is allowed to execute. If you open the script, you will see that the signature is attached at the bottom.

You can also use validate the signature using the **Get-Authenticode** cmdlet.



In this step, I showed you how to sign a Windows PowerShell script, and also how to make it trusted or untrusted on your computer. In the next step, we will make the code signing certificate trusted in our domain using group policy.

Step 4: Make the code signing certificate trusted in my domain

If you were to deploy this in your domain, you would probably use Group Policy to make sure the code signing certificate in use is a trusted publisher. To do this there a two steps:

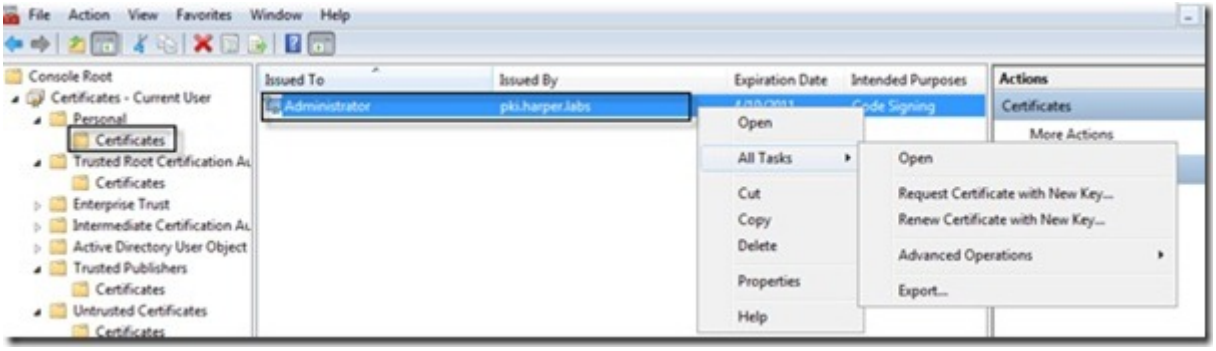
1. Export the code signing certificate.

2. Create a policy and import the code signing certificate into trusted publishers.

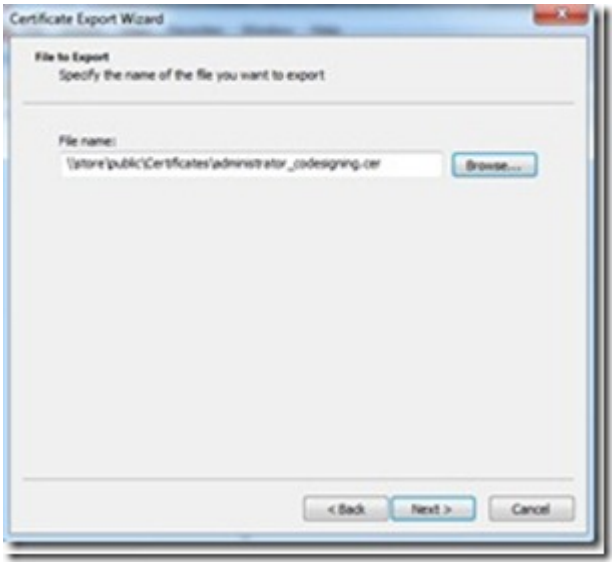
Export the code signing certificate

Let's start with exporting the code signing certificate from the client computer where we requested the certificate.

Start the Certificates snap-in as shown in Step 2 yesterday. Open the **Personal** node, and then **Certificates**. In the content pane, you will now see your certificate. (The one with **Intended Purpose**set to **Code Signing**). Right-click the certificate, click **All Tasks**, and then click **Export**. You can see this in the following image.



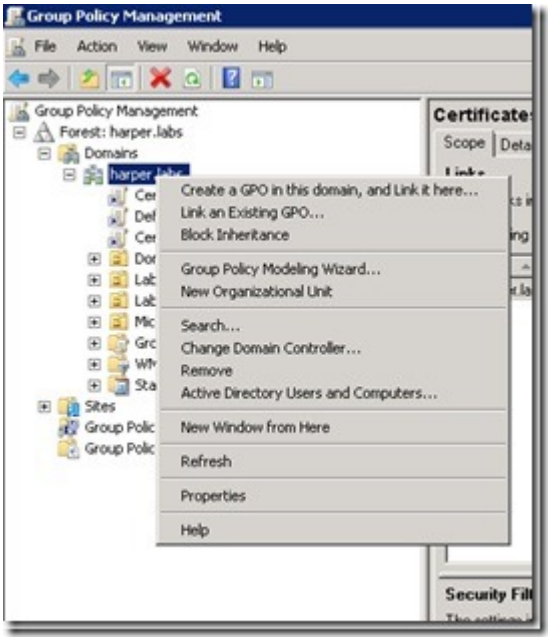
Click **Next** in each of the three dialog boxes you see. Make sure that you save the certificate somewhere you can access it from the computer on which you are going to run Group Policy Management. There is no security risk making the public part of this certificate available, so you can store it wherever you want.



This finishes the export part from the client. Now we need to open up the Group Policy Management Console. This is a part of the Server Administration tools and is usually found if you have installed RSAT (Remote Server Administration Tools) on your client or on your domain controller. For this demonstration, I will run this from one of my domain controllers.

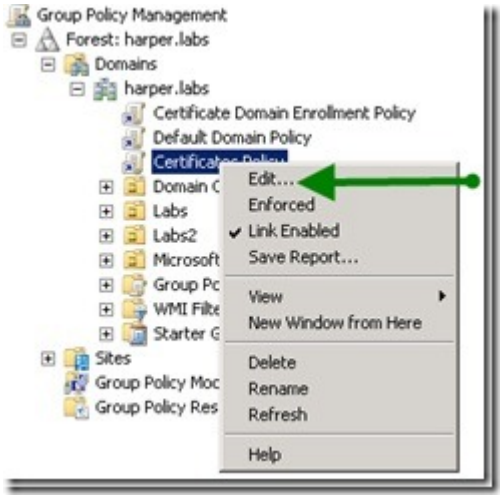
Create a policy and import the code signing certificate into trusted publishers

When I open the Group Policy Management Console, I start by creating a new policy. I open my domain (**harper.labs**), right-click it, and click **Choose Create a GPO in this domain, and link it here**.

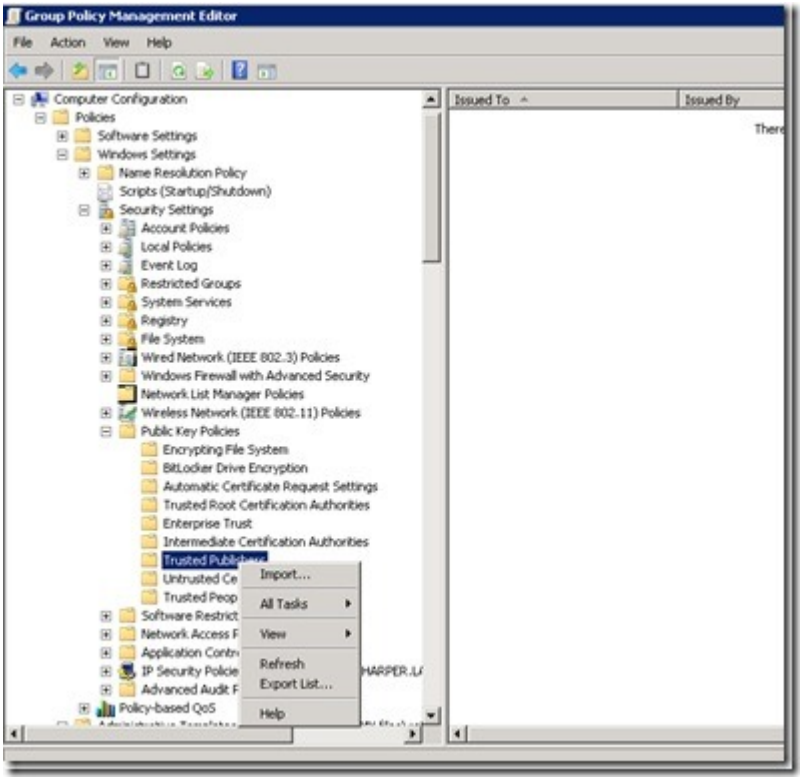


Make sure that you create this Group Policy object (GPO) where you want it in your own domain. For this demonstration, I create it at the domain level. I give the policy the name **Certificates Policy**, and I click **OK**.

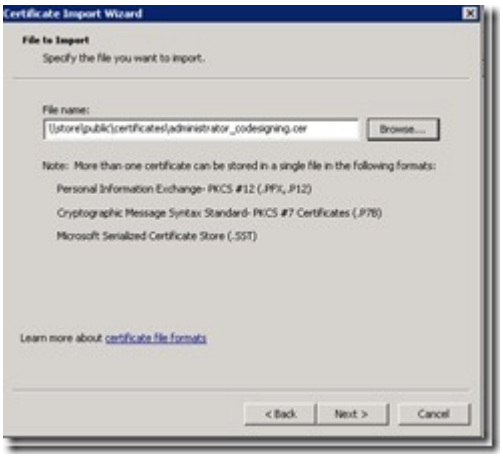
Select the policy (**Certificates Policy**) in the navigation pane, right-click it, and click **Edit**, as shown in the following image.



Wait for the Group Policy Editor to start, and then click **Computer Configuration**, click **Policies**, click **Windows Settings**, and then click **Public Key Policies**. You are now ready to start the import. Right-click **Trusted Publishers**, and then click **Import**.

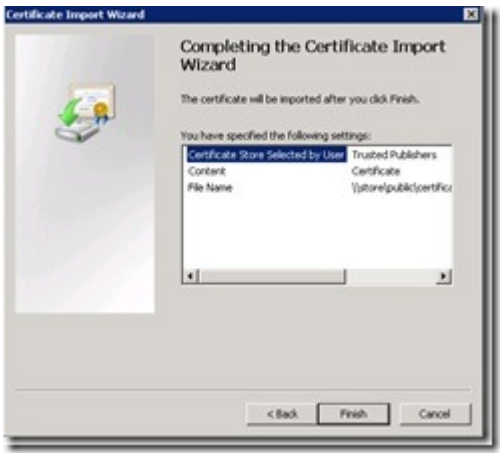


In the dialog box that asks you for the certificate to import, select the certificate you exported earlier. Then click **Next**.

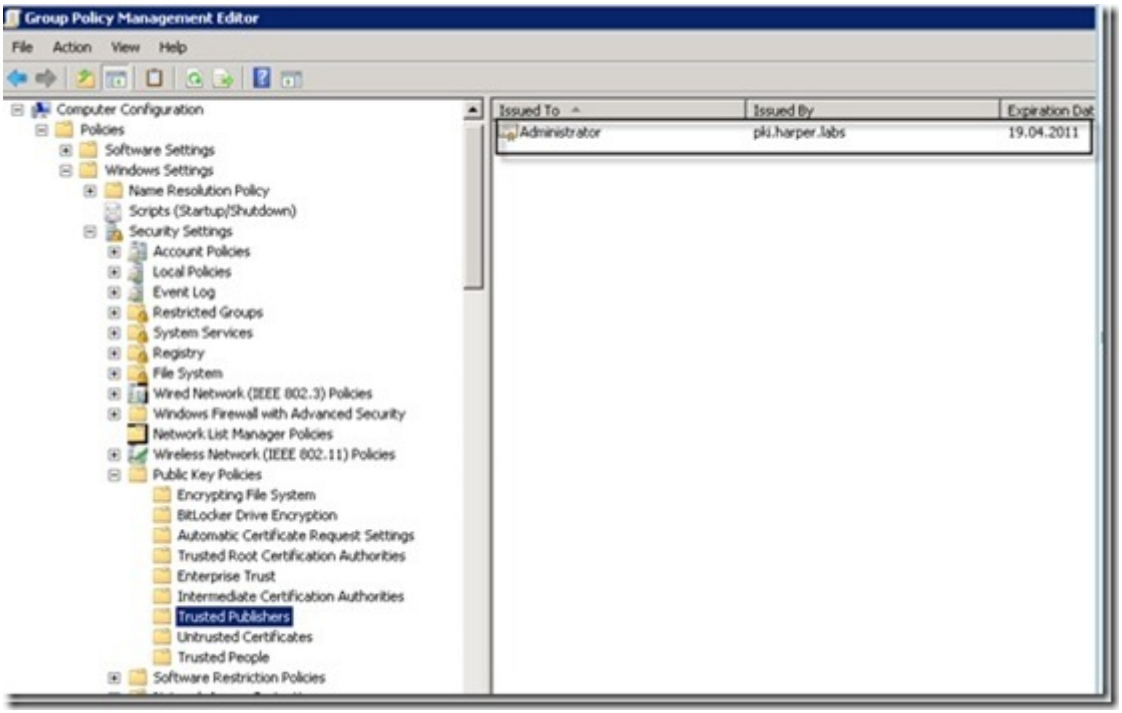


As shown in the following image, make sure the certificate is placed in the Trusted Publishers store, and click **Next**.

Now finish the wizard by clicking **Finish**. You have imported the certificate as a trusted publisher, which is shown in the following image.

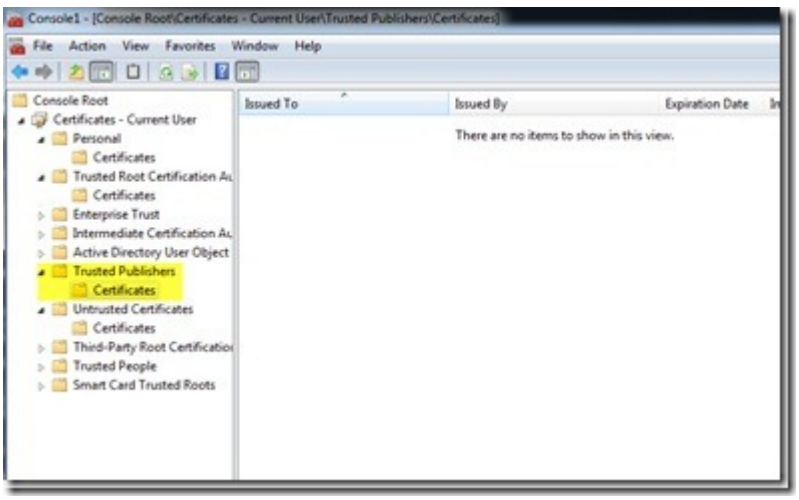


You can confirm this by looking inside the Trusted Publishers node in the Group Policy Editor as shown in the following image.

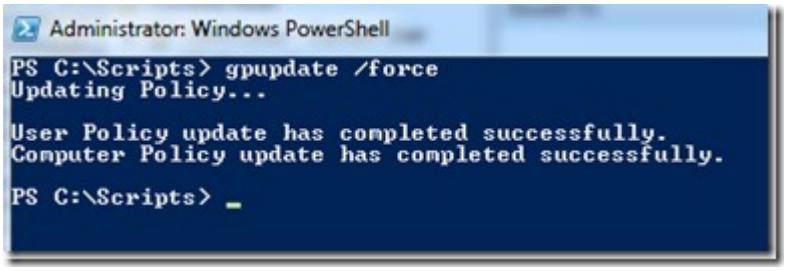


So, the next time the policy is updated on computers in your domain, they will add this certificate as a trusted publisher. You can now run scripts signed by this certificate without being asked if the certificate is trusted or not. You can also do the same with untrusted certificates if you want.

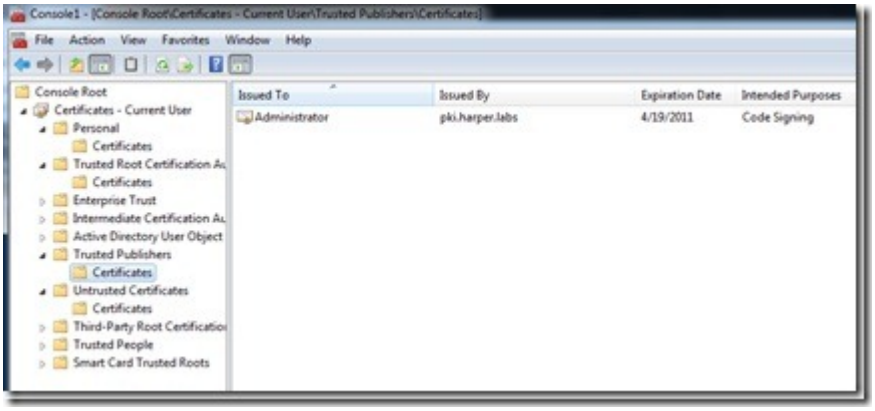
I will test this from my client computer. I will first make sure that the certificate is not in my trusted publishers list. This should be done through the Certificates snap-in on my client.



Then I run **gpupdate /force** from my Windows PowerShell window. The results are shown in the following image.



When the update is finished successfully, I refresh the Trusted Publishers list in my Certificates snap-in. My certificate should now be listed as trusted, as shown in the following image.



I hope this helps, and have fun with Windows PowerShell!

HR, that is all there is to using Windows PKI to sign Windows PowerShell scripts. This brings Guest Blogger Week to a close. Check with us tomorrow as we delve into the virtual mail bag for Quick-Hits Friday.

If you want to know exactly what we will be looking at tomorrow, follow us on Twitter or Facebook. If you have any questions, send e-mail to us at scripter@microsoft.com, or post them on the Official Scripting Guys Forum. See you tomorrow. Until then, peace.

Ed Wilson and Craig Liebendorfer, Scripting Guys