WorkshopPLUS Windows PowerShell Desired State Configuration

Module 4: Security

Student Lab Manual

Version 1.2

Conditions and Terms of Use

Contents

Lab 4: Security

Introduction

PowerShell Desired State Configuration files contain sensitive data. It is important to encrypt this information at rest and in transit. In this lab you will work with certificates for both scenarios.

Objectives

After completing this lab, you will be able to:

- Assess the certificate requirements for securing DSC data.
- Encrypt configuration credentials at rest.
- Encrypt configuration data in transit.
- Compare security techniques against compliance requirements.

Prerequisites

If you have not completed the **Module 1** lab, then you will need to install WMF 5.0 by running the following two scripts in the order listed:

- C:\PShell\Labs\Lab00\Lab_00_01_01_Stage_Resources.ps1
- C:\PShell\Labs\Lab00\Lab_00_01_02_Install_WMF_5.ps1

These scripts will reboot the MS1, MS2, and PULL servers once complete. You will need to reconnect to the PULL virtual machine.

Logon to the **PULL** server as **Administrator** with the password **PowerShell5!** .

Estimated time to complete this lab

120 minutes

For more information

See the slide deck for the course or use the following internet links:

Securing the MOF file – https://msdn.microsoft.com/en-us/PowerShell/DSC/secureMOF

Want to secure credentials in Windows PowerShell Desired State Configuration? - http://blogs.msdn.com/b/powershell/archive/2014/01/31/want-to-secure-credentials-in-windows-powershell-desired-state-configuration.aspx

PsDscAllowPlainTextPassword example - https://gallery.technet.microsoft.com/xComputerManagement-Module-3ad911cc

PsDscRunAsCredential – See WMF 5.0 Release Notes in the C:\PShell folder

Scenario

You are creating a proof-of-concept lab for PowerShell Desired State Configuration. Your company's Information Security group wants a demonstration to assure them that no password data is stored or transmitted in plain text.

NOTE: Use the Windows PowerShell Integrated Scripting Environment (ISE) for typing and editing the PowerShell scripts in this course. The automatic Intellisense will be very helpful in the learning process. If you need to reactivate Intellisense, use the **CTRL SPACE** keyboard shortcut.

Each lab has its own files. For example, the lab exercises for module 2 have a folder at **C:\PShell\Labs\Lab02** on the **PULL** virtual machine. Use these provided files to avoid typing long sections of code.

We recommend that you connect to the virtual machines for these labs through Remote Desktop rather than connecting through the Hyper-V console. When you connect with Remote Desktop, you can copy and paste from one virtual machine to another.

Exercise 4.1: Certificate Services

Objectives

In this exercise, you will:

- Review the Active Directory Certificate Server configuration
- Grant permissions to enroll the WebServer certificate template
- Review the certificate Group Policy for autoenrollment

Scenario

Active Directory Certificate Services has already been installed and configured in your lab on the **DC** server. You need to validate the configuration for use by Desired State Configuration. Also, you need to make sure that later the PULL server can request a web server SSL certificate.

Task 4.1.1: Discovering Certificate Services

1. On the **PULL** server open the PowerShell console and run the following command:

```
Get-Service -ComputerName dc -Name certsvc
```

Question A: What is the **Status** of the service?

2. Launch the **CertificateMMC** from the desktop.
3. Click on **Certificate Templates (DC.contoso.com)**.
4. Scroll to the bottom of the template list on the right and locate the security properties of the two **Workstation Authentication** templates. PKI best practices suggest that you copy any template before deploying it. Compare the **Allow** permission differences for the **Domain Computers** group and record them in the table below:

	Workstation Authentication	Workstation Authentication 2
Full Control	☐	☐
Read	☐	☐
Write	☐	☐
Enroll	☐	☐
Autoenroll	☐	☐

Question B: Later we will need to request a **Web Server** certificate for the PULL server. Does this template currently allow enrollment for the **Domain Computers** group?

---

5. On the **Security** tab of the **Web Server** template add **Domain Computers** with **Allow Enroll**.

NOTE: This lab configuration does not follow best practices. In a production environment your certificate administrator would grant permissions for a specific server in a manner compliant with company policies.

6. Leave the certificate management console open.

Task 4.1.2: Certificate Template for Credential Encryption

1. In the left pane of the **CertificateMMC**, click on **Certificate Templates (DC.contoso.com)**. These certificate templates dictate certificate properties when computers or users perform an enrollment request for a certificate. These certificate templates are stored in Active Directory.
2. Click on **Certificate Authority (DC.contoso.com)**, expand **ContosoRootCA** and click on **Certificate Templates**. The certificate templates shown here have been configured for issue by request. Notice that not all certificate templates from Active Directory have been added to this list.
3. Go back to **Certificate Templates (DC.contoso.com)**. Create a new certificate template for use with Desired State Configuration. Right click the **Workstation Authentication** certificate template, and then click **Duplicate Template.**
4. Go to the **General** tab and type **PowerShell DSC** in the **Template display name** field.
5. Go to the **Subject Name** tab and set **Subject name format** to **Full distinguished name**.
6. Go to the **Extensions** tab, click **Application Policies**, and click **Edit**.
7. Remove the **Client Authentication** application policy by selecting it and clicking **Remove**.
8. Click **Add**, select **Document Encryption**, and click **OK** twice.
9. Go to the **Security** tab and select **Domain Computers**. In the bottom pane select **Autoenroll** and click **OK**.

10. Click on **Certificate Authority (DC.contoso.com)**, expand **ContosoRootCA** and click on **Certificate Templates**.
11. Right click **Certificate Templates**, click **New**, and then click **Certificate Template to Issue**. Select **PowerShell DSC** and click **OK**.

Task 4.1.3: Exploring Certificate Policies

1. On the **PULL** server click **Start**, then **Administrative Tools**, then **Group Policy Management**.
2. Expand the **contoso.com** domain.
3. Click on the **Certificate Policy** at the root of the domain. If prompted, click **OK**.
4. View the settings in the policy. On the top right side click **Settings**. Then on the far-right side click **show all**.
5. Review the settings and answer the question below.

   Question C: Think about the certificate template permissions we discovered earlier. Will every computer in the domain receive a Workstation Authentication certificate? Will every computer in the domain receive a Web Server certificate? Why or why not. Explain your answer.

   ____

   ____

   ____

   ____

6. Close the **Group Policy Management Console** and the **Administrative Tools** window.

Exercise 4.2: Credential Encryption

Objectives

In this exercise, you will:

- Create a configuration with a clear text password.
- Prepare a certificate for credential encryption.
- Create and push a configuration with an encrypted password.

Scenario

In the previous exercise we saw that all computers are receiving workstation certificates to use for encryption. Now we will locate that certificate on the server and use it to encrypt a password.

Task 4.2.1: Unencrypted Credentials

1. Open the **PowerShell ISE** on the **PULL** server and run the following command:

```
Get-DscResource | ForEach-Object { $_.Name ; $_.Properties | Format-Table }
```

2. Scroll through the output and notice that many DSC resources have properties of type **PSCredential**.

   Question A: What one **PSCredential** type property appears on all of the resources?

   ____

3. Change to the **Lab04** directory.

```
cd C:\PShell\Labs\Lab04
```

4. Open the lab script **Lab_04_02_01a_Unencrypted_Credentials.ps1**.
5. This configuration uses the **User** resource to create a local account on the server. The account **Password** property comes from the Password attribute of a PSCredential object. Note that the user name of the credential object is ignored. Run the script to generate the configuration MOF file. Supply any password you like when prompted.

   Question B: Did it succeed? Why or why not?

   ____

   ____

   ____

6. In order to use a plain text password in a configuration you must specify a value for **PsDscAllowPlainTextPassword** in a configuration data block. This is a safety measure ensuring that the user understands they are violating security practices. Open the script file **Lab_04_02_01b_Unencrypted_Credentials.ps1**.
7. Notice the changes in this version of the script. See lines 8, 20-27, 30.
8. DSC configurations can take an automatic parameter of **ConfigurationData**. Run the following in the command pane of the PowerShell ISE:

```
Get-Command ClearTextPassword -Syntax
```

Question C: What is the data type expected by the **ConfigurationData** parameter?

9. Observe the syntax on lines 20-27 of the script. This is a special configuration data structure, and we will cover it more in a later module. For now, add the following text on line 24.

```
PsDscAllowPlainTextPassword = $true
```

10. Save and run the script. When prompted enter any password of your choice. If you get an error, verify that the configuration block looks like this:

```
$ConfigData = @{
    AllNodes = @(
        @{
            NodeName = 'localhost'
            PsDscAllowPlainTextPassword = $true
        }
    )
}
```

11. Now view the plain text password in the MOF file output. Type the following into the command pane at the bottom of the PowerShell ISE:

```
psedit .\ClearTextPassword\localhost.mof
```

12. In the MOF file find the plain text password you typed.
13. We do not want to publish this configuration to any servers, because then you could find the plain text password locally in the path (Though in WMF5 and newer they are encrypted by default): **C:\Windows\System32\Configuration\current.mof**

Task 4.2.2: Working with Certificates for Credential Encryption

1. Earlier we created the **PowerShell DSC** certificate template on the certificate server and explored the Group Policy that was set to autoenroll for all servers. Now we will find this certificate on the MS1 server. Open a remote session to **MS1**:

```
$node = New-PSSession -ComputerName ms1
Enter-PSSession -Session $node
```

2. View the certificate(s) installed on this server:

```
cd cert:
cd .\LocalMachine\My
dir
dir -DocumentEncryptionCert
```

3. If the document encryption ceritificate is not present yet, then perform a group policy update to force **MS1** to attempt auto enrollment:

```
gpupdate /force
```

4. Check for the document encryption certificate again:

```
dir -DocumentEncryptionCert
```

NOTE: It can take a few moments before new certificates appears. Alternatively, repeat step 4 until a new certificate appears.

5. Put the encryption certificate into a variable that we can use. Then inspect the certificate to make sure that it has a private key for decryption and the certificate chain is valid:

```
$cert = dir -DocumentEncryptionCert
$cert | fl *
$cert.PrivateKey
$cert.Verify()
```

6. Now make a folder to hold the public key we must export:

```
cd C:\
md PublicKeys
```

7. Export the certificate public key and view its thumbprint:

```
Export-Certificate -Cert $cert -FilePath C:\PublicKeys\MS1.cer
Get-PfxCertificate -FilePath C:\PublicKeys\MS1.cer
```

8. Now return to the local PowerShell session on the **PULL** server, create a folder to hold public keys, and use the new WMF 5.0 **Copy-Item** technique to copy the file over the session:

```
Exit-PSSession

md C:\PublicKeys
Copy-Item -FromSession $node -Path C:\PublicKeys\MS1.cer -Dest C:\PublicKeys
dir C:\PublicKeys

Remove-PSSession $node
```

9. You now have a copy of the target node's public key to use for encrypting passwords published in configurations to that node (either push or pull).

NOTE: For a fully-automated version of these steps see the blog post:

Want to secure credentials in Windows PowerShell Desired State Configuration?

http://blogs.msdn.com/b/powershell/archive/2014/01/31/want-to-secure-credentials-in-windows-powershell-desired-state-configuration.aspx

Note that this post was written during the WMF 4.0 era of Desired State Configuration and will require minor adjustment to work with document encryption certificates in WMF 5.x.

Task 4.2.3: Encrypting Credentials

1. Now that we have a certificate for encrypting credentials, we need reference it in two places: the configuration script for encryption and the target node LCM for decryption. Begin by opening the lab file **Lab_04_02_03a_Encrypted_Credentials.ps1**.

2. Notice the two new properties in the configuration data block: **CertificateFile** and **Thumbprint**. We must populate these with the certificate information we collected in the previous task. Edit the **CertificateFile** value to point to the **.cer** file.

```
CertificateFile = 'C:\PublicKeys\MS1.cer'
```

3. Now retrieve the thumbprint from the certificate file.

```
Get-PfxCertificate -FilePath C:\PublicKeys\MS1.cer
```

4. Copy and paste the **Thumbprint** value into the configuration data.

```
#Don't use the following thumbprint, it's just an example. Use the thumbprint
#from the previous step's output.
Thumbprint = '1D1FB28E22FB2410291291D47ED45DD2BA725A33'
```

5. This same thumbprint also needs to go into the **LocalConfigurationManager CertificateID** in the configuration script above. However, we want to use the built-in **$Node** variable to pull it out of the **ConfigurationData** instead of hard-coding it. Edit the line to look like this:

```
LocalConfigurationManager
{
    CertificateID = $Node.Thumbprint
}
```

6. Make sure you are in the **Lab04** file path.

```
cd C:\PShell\Labs\Lab04
```

7. Now save and run the configuration script. When prompted enter a complex password such as **P@ssw0rd**.

Question D: What output did you receive?

8. View the configuration MOF file.

```
psedit .\EncryptedPassword\ms1.mof
```

Question E: Is the password encrypted now?

9. Before we push the configuration we must first push the LCM update with the **CertificateID**. This tells the target node which certificate to use for decryption:

```
Set-DscLocalConfigurationManager -Path .\EncryptedPassword -ComputerName ms1 -Verbose
```

10. At last we are ready to push the encrypted password configuration to the target node:

```
Start-DscConfiguration -Path .\EncryptedPassword -Wait -Verbose
```

11. Validate the configuration with these two separate commands:

```
Get-DscConfiguration -CimSession ms1

Invoke-Command -ComputerName ms1 -ScriptBlock {net user}
```

Question F: Write down the local user accounts returned by the NET USER command:

NOTE: These steps are drawn out for teaching purposes in the lab. In your environment you can automate much of this for fast configuration deployments.

Exercise 4.3: Secure Pull Server

Objectives

In this exercise, you will:

- Build a secure DSC pull server
- Publish a configuration
- Configure the LCM on the pull node
- Pull a configuration using HTTPS

Scenario

You want to ensure that configuration MOF files are transmitted over an encrypted channel to the target nodes.

Prerequisites

You should have successfully completed lab task 4.1.1 step 5, granting **Enroll** permission for **Domain Computers** on the **Web Server** certificate template.

You should already have the **xPSDesiredStateConfiguration** module installed. Use only the latest version from the PSGallery with this command:
Install-Module xPSDesiredStateConfiguration –Force

If either of these steps have not been completed, then run the script **Lab_04_03_00_Prerequisites.ps1**.

Task 4.3.1: Build a Secure Pull Server

1. Before we build a secure pull server we must remove the HTTP pull server configuration from the previous module lab. Run the following commands in the **PowerShell ISE**:

```
Install-WindowsFeature Web-Scripting-Tools
Get-WebSite -Name PSDSC* | Remove-Website
```

2. A secure pull server only requires two additional steps beyond the HTTP pull server built in the previous module lab. We must request a web server SSL certificate, and then add its thumbprint to the DSC Service configuration script template. Many web administrators use the IIS Manager console graphical interface to request a certificate. Instead, we want to automate the certificate request with PowerShell. We will use a PowerShell technique called *splatting* with the **Get-Certificate** cmdlet. Begin by opening the lab file **Lab_04_03_01a_CertRequest.ps1**. Complete the server fully qualified domain name (FQDN) for the **DnsName** and **SubjectName** properties to match the script below:

```
$Req = @{
    Template         = 'WebServer'
    DnsName          = 'pull.contoso.com'
    SubjectName      = 'CN=pull.contoso.com'
    Url              = 'ldap:'
    CertStoreLocation = 'Cert:\LocalMachine\My'
}
```

NOTE: It is extremely important that the **DnsName** and **SubjectName** are formatted correctly in the certificate. A mismatch of these to the actual server name will cause an invalid certificate error when accessing the HTTPS address of the server.

3. Now save and run the script.
4. Using these commands, inspect the certificate that was generated:

```
$cert
$cert | fl *
$cert.Certificate
$cert.Certificate | fl *
```

Question A: When does the certificate expire?

____

HINT: See the certificate property **NotAfter**.

Question B: View the **Subject** and **DnsNameList** properties. Do they match the server FQDN you put into the request?

____

5. You should have the Thumbprint in the variable by the same name. If you need to retrieve it manually, then try this command:

```
dir Cert:\LocalMachine\My -SSLServerAuthentication
```

NOTE: If you see multiple certificates listed, use the one with the name matching your request.

6. We are now ready to build the HTTPS pull server. Open the template script here: **Lab_04_03_01b_Configure_HTTPS_Pull_Server.ps1**

   Question C: Study this template code. Where should you put the web SSL certificate thumbprint?

____

7. Edit line 18. Paste the certificate thumbprint between the quotes.

8. Save and run the script. While the script is running review the code in the script pane above.

   Question D: Why do you think this configuration includes the features for the web management console and the scripting tools? Are they required?

   ____

   ____

   ____

9. Finally, verify the web service is responding correctly. If prompted to add **about:blank** to trusted sites, then do so.

```
Start-Process "https://pull.contoso.com:8080/PSDSCPullServer.svc"
```

> **NOTE:** If you receive an error message in the browser, you may have used an out-dated copy of the **xPSDesiredStateConfiguration** resource module. Rerun the prerequisites script at the beginning of this exercise. If you are prompted to install NuGet, then accept the prompt. Repeat the exercise to install a fresh pull server using the updated module.

10. In the browser window double click the **padlock icon** in the right side of address bar. Click **View Certificates**. Click **Details**. Scroll to the bottom of the fields list and click on **Thumbprint**.

    Question E: Does this thumbprint match the one in your configuration script for the pull server?

    ____

Task 4.3.2: Publish a Configuration for Pull

1. Now that the secure pull server is ready, we need to publish a configuration for a node to pull. Begin by opening the lab script **Lab_04_03_02a_Configuration_For_Pull.ps1**.

2. Review the script. We have added a local group resource in addition to the previous user account. Make sure to update the Thumbprint value on or near line 34 to match the output from one of the following commands:

```
    Get-PfxCertificate -FilePath C:\PublicKeys\MS1.cer
    #or
    (Get-DscLocalConfigurationManager -CimSession ms1).CertificateID
```

Question F: Which line in this configuration will guarantee that the user is present before attempting to add it to the new group? Write the line below:

____

3. Run the script. When prompted, provide the password **P@ssw0rd**. The password must meet complexity requirements. Notice that this configuration will have an encrypted credential, and it will be delivered over an encrypted session to the node.
4. In the previous module we learned how to publish a configuration for pull. We will follow those steps here again. First, view the MOF file generated by the configuration:

```
cd C:\PShell\Labs\Lab04

dir .\EncryptedPull
```

5. Generate a new GUID and capture it into a variable:

```
$guid = (New-Guid).Guid
```

6. We will use this variable in this task and the next one, so we want to keep the PowerShell ISE open, leaving the variable in memory for later. Next we need to construct the target path for the destination GUID configuration name on the pull server. Use **TAB** complete to make sure the file path is accurate as you type:

```
$ConfigPath = "C:\Program Files\WindowsPowerShell\DscService\Configuration\$guid.mof"
```

7. Copy the **ms1.mof** file to the pull server configuration directory, and rename it with the GUID:

```
Copy-Item -Path .\EncryptedPull\ms1.mof -Destination $ConfigPath -Verbose
```

8. Generate the required checksum. It is a good idea to always use the **-Force** parameter when creating checksums. By default, the cmdlet will not override previous files if they exist.

```
New-DscChecksum -Path $ConfigPath -Force
```

9. Finally, verify that the configuration and checksum files were created successfully:

```
dir "C:\Program Files\WindowsPowerShell\DscService\Configuration\"
```

10. Do not close the PowerShell ISE. Leave it open for the next task, keeping variables in memory to reuse later.

Task 4.3.3: Configure the Pull Node

1. We now have a secure pull server with a configuration ready to pull. We will use the WMF 5.0 syntax to configure the Local Configuration Manager (LCM) on the target

node. Using **Windows Explorer** browser to **C:\PShell\**. Find the folder containing the latest WMF 5.0 install and release notes. Open the release notes document.

2. The document will open in WordPad with limited functionality. This is OK. Click the **Find** button on the right side of the toolbar.
3. Search for the phrase **meta-config**. The first occurrence is in the table of contents. Find the occurrence in the body of the document titled **Configure DSC's Local Configuration Manager with the meta-configuration attribute**.
4. Scroll down and read this section.

Question G: What is the resource name used for an HTTP or HTTPS configuration pull server?

____

> **NOTE:** You can download your own copy of the latest WMF 5.0 release notes by searching at **http://download.microsoft.com**.

5. Open the lab file **Lab_04_03_03a_LCM_For_Pull.ps1**.
6. The LCM and configuration data syntax have been provided for you. However, the GUID for the configuration needs to be updated for this node. On line 25 supply the GUID either as a string value or the variable name created in the previous task. It is important that this GUID match the MOF file names that we published earlier. In the command pane type:

```
$guid
```

7. Copy and paste the value between the quote marks on line 25.

```
ConfigurationID = 'paste GUID here'
```

8. Here is an example of what your line may look like. The GUID will be different.

```
ConfigurationID = 'b9b7b70f-9f7a-4ee7-b5f0-ffc146274c5a'
```

> **TIP:** If you have lost the GUID captured in the variable earlier, find it again by checking the MOF file name published for pull:
>
> dir "C:\Program Files\WindowsPowerShell\DscService\Configuration\"

9. Notice that the **ConfigurationID** is pulled from the **ConfigurationData** into the LCM configuration above on line 8.
10. Right above ConfigurationID update the **Thumbprint** value with the output from:

```
(Get-DscLocalConfigurationManager -CimSession ms1).CertificateID
```

11. To finish the LCM we must provide the correct server URL on line 14:

```
ServerUrl = 'https://pull.contoso.com:8080/PSDSCPullServer.svc'
```

12. Save and run the script. This will generate the **.meta.mof** file required to configure the target node.
13. Now we need to send the LCM configuration to the target node:

```
Set-DscLocalConfigurationManager -Path .\HTTPS_Pull -Verbose
```

Question H: At this point has the user and group configuration which we created earlier been applied to the target node? Why or why not?

14. We have created a secure pull server, published a configuration, and set the LCM on the target node to know where to find the configuration. At this point we can wait for the LCM refresh and configuration intervals to pass, or we can force the configuration to apply immediately. Run the following command:

```
Update-DscConfiguration -Wait -Verbose -ComputerName ms1
```

15. You should see the blue verbose scroll on server **MS1** as it applies the user and group configuration we created earlier.

NOTE: You may see an error that the password does not meet complexity requirements. If this happens you will need to repeat task 4.3.2 and supply a password that meets the requirements. Then repeat task 4.3.3.

Question I: In this exercise we delivered an encrypted password over an encrypted session (HTTPS). Now we need to satisfy the Information Security team by demonstrating that the password is not stored in plain text on the target node. What is the name and path to the configuration file applied on MS1?

16. Run the following commands to view the remote configuration applied to **MS1**. Find the encrypted password in the file output.

```
Enter-PSSession -ComputerName ms1

Get-Content C:\Windows\System32\Configuration\Current.mof
```

NOTE: In WMF 5 the whole MOF is encrypted on the target node so you will not likely be able to find the encrypted password. You can view the MOF that is on the pull server and see the encrypted password there.

17. Use the following commands to see that the configuration put the **AppServiceAccount** user into the **AppServiceGroup** (use the remote session to ms1):

```
net localgroup AppServiceGroup

Get-DscConfiguration
```

18. Close the remote session:

```
Exit-PSSession
```