Azure PowerShell Authentication

Before you access Azure using PowerShell, you need to authenticate the account you want to access Azure using. Generally to authenticate you execute Azure CmdLet, Add-AzureAccount. This will prompt you for your credentials which are cached and used for subsequent interactions with Azure during the PS session.

Within a script you might authenticate as follows:

$userName = "<your organizational account user name>"

$securePassword = **ConvertTo-SecureString** `

-String "<your organizational account password>" -AsPlainText -Force

$cred = **New-Object** System.Management.Automation.PSCredential($userName, $securePassword)

**Add-AzureAccount** -Credential $cred

The problem with method is that your password is contained in the code in plain text

An alternative method id to encrypt the password using PKI which provides a high degree of security.

# Certificate based authentication using PowerShell

## Create a Certificate

Create a server certificate, you can do this via IIS Manager (this example uses v10 on Windows 10)

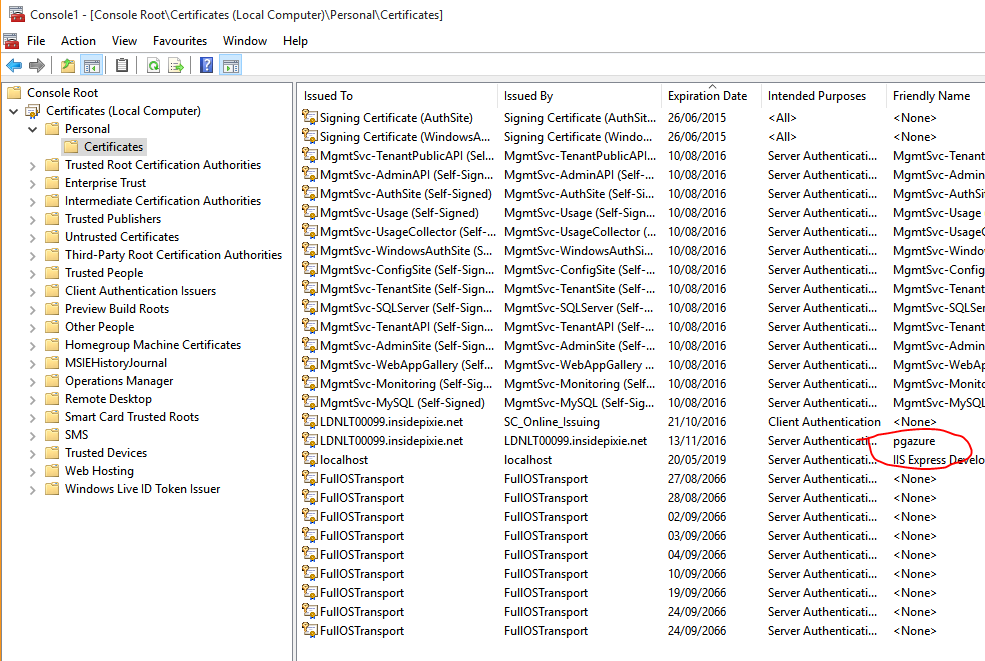
1. Launch IIS manager
2. Select the “Server Certificates” icon
3. Select “Create Self-Signed Certificate” from the actions pane on the right.
4. Select a friendly name, I chose “pgazure”. The certificate will be created
5. Double click the certificate and select “Details” tab. Scroll down and make a note of the “Thumbprint” and the friendly name.
6. Export the certificate:
   1. Enter the file name
   2. Enter a password. The password is entered when you come to import the certificate into the vault on a machine you want to use it on. That is the only time it is used. It provides an extra level of security to ensure the certificate can only be used by the owner.
7. Now you will have a certificate in a .pfx file which can be deployed into the certificate vault on any machine that wishes to execute PS Azure scripts.

## Import the Certificate

I order to use the certificate for authentication, it needs to be imported into the machine where the scripts are run.

Import the certificate into the certificate vault as follows:

1. Run mmc.exe and load the certificates snap-in
2. Select “Computer Account” when asked what the snap-in should manage
3. On “Actions”
4. Import the certificate previously created by selecting Import on the “Personal Store”
5. Select c:\temp\pgazure.pfx
6. Enter the password => “Password1”
7. DO NOT enable the “Enable strong private key” option
8. Select “Next” and opt to place the certificate in the Personal Store
9. The certificate is listed with a “Friendly Name” the same as the filename, in this example it is named pgazure



1. Double click on the certificate and get the value of the Thumbprint, it will look something like this:

‎f0 5e fd f3 e2 b5 63 ed 1b b5 1b 19 64 7e e7 d0 61 63 a2 3c

Record the thumbprint as it is used in code to identify the certificate, think of it as its ID or primary key

## Encrypt the Credentials

The service account credentials need to be encrypted using the certificate. The encrypted credential file is the analogous to a file containing a password. The encrypted credential file is later used by code that requires to authenticate to Azure.

The following code prompts for the service account password and then encrypts it using the certificate’s public key.

# A certificate needs to be generated and stored in the local machine vault under personal

# certificates

# You can use IIS manager to issue an example cert of use one issued by a CA

# Convert the service Account Password to a Secure String.

# A secure String is just a variable type that cannot be read programmtically

$securePassword = Read-Host -AsSecureString -Prompt "Enter Password"

$objectFile = ".\account-creds.xml"

# Generate a key that will be used to encrypt our password, store it in a 32 byte array

$key = New-Object byte[](32)

[System.Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($key)

# Now encrypt the password using the randomly generated key

$encryptedPassword = ConvertFrom-SecureString -SecureString $securePassword -Key $key

# Retrieve the certificate using the certificates thumbprint,

# this can be opbtained by looking in the vault at the certificates properties

$cert = Get-Item -Path Cert:\LocalMachine\My\4EC0E3E7F31F26E27F3ED9F444ED9F91DF412309

-ErrorAction Stop

# Use the public key of the certificate to encrypt our key that we used to

# encrypt the password When we come to decrypt the password we use the certificate's

# private key to decrypt $encryptedKey we can then use key to decrypt the password

$encryptedKey = $cert.PublicKey.Key.Encrypt($key,$true)

# Create a PS dynamic object that contains the encrypted key and the encrypted password.

# So you need this file along with the certificate to obtain the service account cerdentials

$object = New-Object psobject -Property @{

Key = $encryptedKey

Payload = $encryptedPassword

}

Write-Host "Encrypted Credentials stored in $objectFile"

$object | Export-Clixml $objectFile

## Use the encrypted password file in Powershell scripts

The following code is an example of how to use the encrypted password file to authenticate the service account with Azure:

$serviceAccount = Read-Host -Prompt "Enter Service Account Name: "

$objectFile = ".\account-creds.xml"

# Read the Encrypted Key and credentials file into a PS object.

# The Object was previously serialised using whaver format PS uses for object serialisation

# The original object was a anonymous object with two properties: key and payload

$object = Import-Clixml $objectFile

# Read the associated cert from the localmachine vault

$cert = Get-Item -Path Cert:\LocalMachine\My\4EC0E3E7F31F26E27F3ED9F444ED9F91DF412309

-ErrorAction Stop

# Using the cert private key, decrypt the key stored in the credentials object

$key = $cert.PrivateKey.Decrypt($object.Key, $true)

# Decrypt the password using the decrypted key and save in SecureString object

$securePassword = $object.PayLoad | ConvertTo-SecureString -Key $key

$credential = New-Object System.Management.Automation.PSCredential ($serviceAccount, $securePassword)

Add-AzureAccount -Credential $credential

## Summary

1. Create or obtain a certificate. The certificate should contain a public and private key and be password protected.
2. Install the certificate on the machine which is used to encrypted the service account password and any machine that you want to run scripts as the authenticated user.
3. Encrypt the service account password into a password file
4. Use the password file to obtain and decrypt the password within your Azure PS scripts