WinRM Configuration

Windows Remote Management (WinRM) is used by the Windows connection manager to connect to nodes agentlessly.

Enabling WinRM via Group Policy

Using Windows Group Policy to enable WinRM provides users with an interface to centralize the management and configuration of WinRM for new and existing Active Directory computers. This article explains the steps required to create and apply an “Enable WinRM” Group Policy Object.

Creating the Group Policy Object (Windows Server 2012 R2)

1. Click Start, Run, and type “gpedit.msc” to open the Windows Group Policy Object Editor window
2. Find the Windows Remote Management (WinRM) GPO under Computer Configuration\Administrative Templates\Windows Components\Windows Remote Management (WinRM)\WinRM Service
3. Select the Allow remote server management through WinRM setting and click edit policy setting from the left information pane to open the Allow remote server management through WinRM configuration window
4. Click on the Enable radio button and type in * for both IPv4 and IPv6 filter boxes as shown below and click Apply and OK to save the settings.
Enabling HTTPS WinRM for Systems Not Connected to a Domain

When used between two systems on a domain, WinRM uses Kerberos to authenticate that the target server is trusted. Agentless scanning of non-domain-joined systems requires additional setup.

WinRM on the Connection Manager host must be able to authenticate the target system. This is achieved with certificates.

To perform agentless scanning on non-domain systems, the following is required:

- An HTTPS WinRM listener on the scan target, with a certificate that is considered valid on the Connection Manager host
- Basic authentication configured on the scan target WinRM listener, to allow local account user authentication
- Firewall rules on the scan target to permit traffic to the HTTPS WinRM listener port (default 5986)
- A TrustedHosts entry for the target in the WinRM configuration on the Connection Manager host

The following documentation describes the creation of a self-signed certificate on a target host and the configuration of an HTTPS WinRM listener service. It exports the self-signed certificate to a predictable location for importation into the trusted certificate store on the Connection Manager host.
Importing the certificate on the Connection Manager host and modifying the WinRM TrustedHosts list are manual steps, described further on.

Note that WinRM over HTTPS uses port 5986 by default, which will have to be configured on the scan target's Node Edit page in UpGuard.

Remote Endpoint Configuration

Starting with the remote endpoint that we are wanting to initiate a connection to, the following PowerShell script will configure everything we need for HTTPS communication to be setup and will also (optionally) remove access for HTTP. You may need to first enable the ability to run scripts on your system using the command

```
```

Before you start, you may optionally set "Continue" to view the status of the script during runtime. Run the following script on the client node that you wish to connect to:

```powershell
# Configure a Windows host for remote management with WinRM/HTTPS
# ---------------------------------------------------------------
# # This script checks the current WinRM/PSRemoting configuration and
# # makes the necessary changes to allow Ansible or UpGuard to connect,
# # authenticate and execute PowerShell commands.
# # It also exports keys so that they can be loaded on to UpGuard's Windows
# # connection managers.
# # Cert validity is set to 10 years for this use case.
# # Set $VerbosePreference = "Continue" before running the script in order to
# # see the output messages.
# Set $SkipNetworkProfileCheck to skip the network profile check. Without
# specifying this the script will only run if the device's interfaces are in
# DOMAIN or PRIVATE zones. Provide this switch if you want to enable winrm on
# a device with an interface in PUBLIC zone.
# # Written by Trond Hindenes <trond@hindenes.com>
# Updated by Chris Church <cchurch@ansible.com>
# Updated by Michael Crilly <mike@autologic.cm>
# Updated by Zachary Acreman <zakk.acreman@upguard.com>
# # Version 1.0 - July 6th, 2014
# Version 1.1 - November 11th, 2014
# Version 1.2 - May 15th, 2015
# Version 1.3 - May 13th, 2016

Param (  
    [string]$SubjectName = $env:COMPUTERNAME,  
    [string]$IpAddress = (get-netadapter | ? status -eq 'up' | get-netipaddress -AddressFamily ipv4).ipaddress,  
    [string]$FilePath = $env:USERPROFILE + "\winrm.crt",  
    [int]$CertValidityDays = 3650,  
    [switch]$SkipNetworkProfileCheck,  
    [switch]$CreateSelfSignedCert = $true  
)
```
# Setup error handling.
Trap
{
  $_
  Exit 1
}
$ErrorActionPreference = "Stop"

# Detect PowerShell version.
If ($PSVersionTable.PSVersion.Major -lt 3)
{
  Throw "PowerShell version 3 or higher is required."
}

# Find and start the WinRM service.
Write-Verbose "Verifying WinRM service."
If (!(Get-Service "WinRM"))
{
  Throw "Unable to find the WinRM service."
}
ElseIf (((Get-Service "WinRM").Status -ne "Running")
{
  Write-Verbose "Starting WinRM service."
  Start-Service -Name "WinRM" -ErrorAction Stop
}

# WinRM should be running; check that we have a PS session config.
If (!((Get-PSSessionConfiguration -Verbose:$false) -or (!(Get-ChildItem WSMan:\localhost\Listener))))
{
  if ($SkipNetworkProfileCheck)
  {
    Write-Verbose "Enabling PS Remoting without checking Network profile."
    Enable-PSRemoting -SkipNetworkProfileCheck -Force -ErrorAction Stop
  }
  else
  {
    Write-Verbose "Enabling PS Remoting"
    Enable-PSRemoting -Force -ErrorAction Stop
  }
}
Else
{
  Write-Verbose "PS Remoting is already enabled."
}

# Make sure there is a SSL listener.
$listeners = Get-ChildItem WSMan:\localhost\Listener
If (!$listeners | Where {$_-Keys -like "TRANSPORT=HTTPS"})
{
  # HTTPS-based endpoint does not exist.
  $cert = New-SelfSignedCertificate -DnsName $IpAddress,$SubjectName
  -CertStoreLocation "Certi:\LocalMachine\My"
  $thumbprint = $cert.Thumbprint
  Write-Host "Self-signed SSL certificate generated; thumbprint: $thumbprint"

  Export-Certificate -Cert $cert -FilePath $FilePath
Write-Host "Self-signed SSL certificate exported to $FilePath"

# Create the hashtables of settings to be used.
$valueset = @{}
$valueset.Add('Hostname', $IpAddress)
$valueset.Add('CertificateThumbprint', $thumbprint)

$selectorset = @{}
$selectorset.Add('Transport', 'HTTPS')
$selectorset.Add('Address', '*')

Write-Verbose "Enabling SSL listener."
New-WSManInstance -ResourceURI 'winrm/config/Listener' -SelectorSet $selectorset -ValueSet $valueset

Else

Write-Verbose "SSL listener is already active."

# Check for basic authentication.
$basicAuthSetting = Get-ChildItem WSMan:\localhost\Service\Auth | Where {$_.Name -eq "Basic"}
If (($basicAuthSetting.Value) -eq $false)

Write-Verbose "Enabling basic auth support."
Set-Item -Path "WSMan:\localhost\Service\Auth\Basic" -Value $true

Else

Write-Verbose "Basic auth is already enabled."

# Configure firewall to allow WinRM HTTPS connections.
$fwtest1 = netsh advfirewall firewall show rule name="Allow WinRM HTTPS"
$fwtest2 = netsh advfirewall firewall show rule name="Allow WinRM HTTPS" profile=any
If ($fwtest1.count -lt 5)

Write-Verbose "Adding firewall rule to allow WinRM HTTPS."
netsh advfirewall firewall add rule profile=any name="Allow WinRM HTTPS" dir=in localport=5986 protocol=TCP action=allow

ElseIf (($fwtest1.count -ge 5) -and ($fwtest2.count -lt 5))

Write-Verbose "Updating firewall rule to allow WinRM HTTPS for any profile."
netsh advfirewall firewall set rule name="Allow WinRM HTTPS" new profile=any

Else

Write-Verbose "Firewall rule already exists to allow WinRM HTTPS."

# Test a remoting connection to localhost, which should work.
$httpResult = Invoke-Command -ComputerName "localhost" -ScriptBlock {get-computername} -ErrorVariable httpError -ErrorAction SilentlyContinue
$httpsOptions = New-PSSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck
```powershell
$httpsResult = New-PSSession -UseSSL -ComputerName "localhost" -SessionOption
$httpsOptions -ErrorVariable httpsError -ErrorAction SilentlyContinue

If ($httpResult -and $httpsResult)
{
    Write-Verbose "HTTP: Enabled | HTTPS: Enabled"
}
ElseIf ($httpsResult -and !$httpResult)
{
    Write-Verbose "HTTP: Disabled | HTTPS: Enabled"
}
ElseIf ($httpResult -and !$httpsResult)
{
    Write-Verbose "HTTP: Enabled | HTTPS: Disabled"
}
Else
{
    Throw "Unable to establish an HTTP or HTTPS remoting session."
}
Write-Verbose "PS Remoting has been successfully configured."
```

### Connection Manager Endpoint Configuration

The next step that is required is to transfer the `winrm.crt` file to the machine that will initiate HTTPS WinRM connections to remote endpoints.

First, copy the `winrm.crt` file from the current user’s home directory to the Windows Connection Manager host server. To install the self-signed certificate that we created on the host, we can use the following PowerShell command:

```powershell
Import-Certificate -Filepath "C:\Path\To\Certificate" -CertStoreLocation "Cert:\LocalMachine\Root"
```

### Verification of Certificate Installation

Optionally, you can verify that the certificate has been stored correctly on both the Windows CM host server and the remote endpoint servers using the certificate add-in of the Microsoft Management Console (MMC).

1. Type ‘mmc’ on the Start screen and launch the mmc console.
2. Navigate to File > Add/Remove Snap-in…
3. Double click on Certificates from the Available snap-ins.
5. Navigate to the Personal\Certificates folder from the sidebar.
6. The certificate corresponding to the hostname or IP address should be in that folder.

### Adding Remote Endpoints to the List of Trusted Hosts
To quickly add the remote endpoint server node to the list of trusted hosts on your Windows CM machine, you may run the following command, replacing "machineA" with the hostname or IP Address of your machine:

```
winrm set winrm/config/client '{TrustedHosts="machineA"}’
```

You may also add more to the list of Trusted Hosts, as the string passed into the TrustedHosts key is a comma separated list of values which it will recognize. A quicker way to add remote endpoints if you choose to write a script for it is by working with the PSDrive (WSMan:\):

You can retrieve a list of your TrustedHosts by running the following command on your Windows CM host:

```
Get-Item WSMan:\localhost\Client\TrustedHosts
```

To set TrustedHosts (replace machineA etc. where appropriate):

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "machineA,machineB,machineC"
```

Wildcards are also accepted:

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value "192.168.140.*,machine*,*"
```

**Note:**

The advantage of working with PSDrive methods is that you can use `Get-Item` to retrieve the current value of the TrustedHosts, save it to a variable (e.g. $OldVal), and `Set-Item` for the new ones with $OldVal,NewHost.

### Verification of WinRM HTTPS Connectivity

To verify that HTTPS WinRM is setup, you can run the command:

```
Enter-PSSession -ComputerName COMPUTERNAME.DOMAIN.COM -UseSSL -Credential (Get-Credential)
```

where you will be prompted to enter the credentials of the user you would like to connect as (connection manager service account) to the remote endpoint. You should expect to successfully connect to the remote endpoint server from your PowerShell console.

**Note:**

Additionally, the same command can be run but without the `-UseSSL` flag to ensure that non-SSL connections can no longer be made.

Finally, your WinRM node in UpGuard can now be edited to use the **port 5986**.

**Note:**

WinRM typically communicates through 5985 (HTTP). Since we are now using HTTPS through WinRM, you are
required to set the port manually to 5986 for the nodes that are configured in this guide.

Additional Help

- Adding a WinRM Node