Lab 17:

PowerShell 7

Linux Server Security   
 2024-2025

© Daan Pareit

## Introduction

# Lab concept

PowerShell is evolving to a cross-platform shell. We’ll see how the new PowerShell can be used on Windows and Linux and how remote sessions can be created to each other.

# Learning goals

* Installing PowerShell 7 on Windows 11 and Windows Server
* Installing PowerShell 7 on linux distros
* Remote PowerShelling from Linux to Windows over WinRM
* Remote PowerShelling from Windows to Linux over SSH
* Some other options for remote connections by using OpenSSH server on Windows

# Practicalities and prerequisites

You need:

* pfSense VM
* Windows 11 VM
* Windows Server VM (king)
* Debian VM
* RHEL VM

## “Windows PowerShell” vs “PowerShell 7” on Windows 11

First, check what default Windows PowerShell we have running on our different machines

* For the Windows client, core server and full server, check the version and edition of the PowerShell which is installed (check the PSVersiontable variable).

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

This is for Core server and Win 11 machine

* Install the latest stable release of PowerShell 7 now also on your Windows 11 VM via winget. This will install it complementary to your default built-in Windows PowerShell 5.1 . Follow instructions on <https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows>

A screen shot of a computer

Description automatically generated

A computer screen with white text

Description automatically generated

* Close all open terminals and then open a new one. Verify that you can run Windows PowerShell with the ‘powershell’ command and PowerShell 7 with the ‘pwsh’ command. Check the differences in version and edition for both PowerShell flavours (check the PSVersiontable variable).

A screenshot of a computer

Description automatically generated

## PowerShell 7 on Windows Server

We will now install PowerShell 7 on the Windows Server. This will install it complementary to your default built-in Windows PowerShell.

* ‘winget’ will not work (yet) on Windows Server. You need to install the MSI file manually. Look at <https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows> . to download the latest stable release of PowerShell 7 on your Windows Server (e.g. using curl.exe ). You’ll need the “PowerShell-<version>-win-x64.msi” file. Execute the msi file, as explained in de web page or simply via “msiexec.exe /i <msi\_file>" . (Don’t use “.\” for relative paths, msiexec doesn’t understand that.)

Invoke-WebRequest -Uri "https://github.com/PowerShell/PowerShell/releases/download/v7.4.6/PowerShell-7.4.6-win-x64.msi" -OutFile "PowerShell-7.4.6-win-x64.msi" -UseBasicParsing

* Logout and login again to be able to use the ‘pwsh’ command to start PowerShell 7.

A screenshot of a computer program

Description automatically generated

* Now compare $PSVersiontable on the default Windows PowerShell and on the PowerShell 7 on Windows Server.

A screenshot of a computer program

Description automatically generated

It has a special edition, Core.

Also shows the OS, and of course the version.

* Run the following on PowerShell cmdlet on Windows Server, both in Windows PowerShell and in PowerShell 7: Get-Variable is\*

A screen shot of a computer

Description automatically generated

PowerShell 7

A black background with white text

Description automatically generated

PowerShell 5

Useful for scripting indeed.

## PowerShell 7 on linux

Now, let’s install the latest PowerShell 7 on our Linux!

# Install PowerShell 7 on Debian

* Install PowerShell on your Debian VM. Go to <https://aka.ms/powershell> and follow the links for installing on Linux. There’s a specific section for Debian. The method of adding an extra Package Repository to your apt package manager is not (yet) mentioned for Debian 12, but you can use the same method. It works just fine.

# Download the PowerShell package file

wget https://github.com/PowerShell/PowerShell/releases/download/v7.4.6/powershell\_7.4.6-1.deb\_amd64.deb

###################################

# Install the PowerShell package

sudo dpkg -i powershell\_7.4.6-1.deb\_amd64.deb

# Resolve missing dependencies and finish the install (if necessary)

sudo apt-get install -f

https://learn.microsoft.com/en-us/powershell/scripting/install/install-debian?view=powershell-7.4

# Install PowerShell 7 on RHEL

* Install PowerShell 7 on your RHEL VM. Also go to <https://aka.ms/powershell> and follow the procedure to add the Microsoft repository and subsequently install PowerShell via the dnf package manager.

sudo dnf install https://github.com/PowerShell/PowerShell/releases/download/v7.4.6/powershell-7.4.6-1.rh.x86\_64.rpm

# Using PowerShell on Linux

* Run PowerShell on Linux via pwsh. Check if you are indeed running PowerShell 7 by displaying the value of PSVersiontable.

A screenshot of a computer

Description automatically generated

* Now run: ‘Get-Variable is\*’ on Windows and Linux and see how this can be useful in PowerShell 7 scripts to determine the host OS.

A screen shot of a computer

Description automatically generated

* On Linux, you can now also use common cmdlets in PowerShell. Try e.g. ‘Get-Process’ and ‘Get-ChildItem’.

A screenshot of a computer

Description automatically generated

Sweet mother of god this feels so wrong for the Linux community

## PowerShell Remoting between Linux and Windows, using PowerShell 7

# PowerShell remoting from Windows to Windows using PowerShell 7

* Verify that the ‘normal’ way of PS remoting also works in PowerShell 7 (cfr Lab about “Windows Active Directory recap”, section about remote administration). Try to connect from your Windows 11 VM to your Windows Server within PowerShell 7:
  + Enter-PSSession -ComputerName king -Credential donald@COMPANY-<firstname>.serverlabs.be
* What shell do we get on king when we have made the remote PowerShell session?

I get a powershell 5, so a Desktop one.

A computer screen shot of a black screen

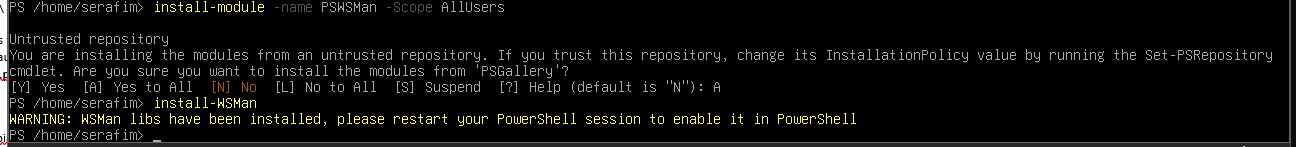
Description automatically generated

# PowerShell remoting from linux to Windows (!), using WinRM

Now, mind blowing, let’s connect from debian linux to Windows, using remote powershelling.

* We need an additional package for Microsoft authentication: gss-ntlmssp. Install the package. To know more about it, you can use:  
  apt show gss-ntlmssp
* To have the proper communication with Windows over WinRM from within Linux, we’ll add a third-party module within your PowerShell 7. On your debian start an elevated powershell: ‘sudo pwsh‘. In there, perform following commands:  
  Install-Module -Name PSWSMan -Scope AllUsers

Install-WSMan  
<https://www.powershellgallery.com/packages/PSWSMan>

  
Exit your elevated powershell session and start a regular pwsh session.

* Try to connect from your debian VM to your Windows Server in PowerShell 7, similar as Windows-to-Windows:
  + Enter-PSSession -ComputerName 192.168.11.50 -Credential donald@COMPANY-<firstname>.serverlabs.be
    - Note: that we use the IP address here, as the name ‘king’ is not resolved on Debian. (And no TrustedHosts setting is needed as in Windows.)

A black screen with white text

Description automatically generated

* What shell do we get on KING when we have made the remote PowerShell session?

A screenshot of a computer

Description automatically generated

Also a Windows Powershell

* Note that we are indeed connecting over the traditional Windows Remote Management port. Run netstat -p TCP on king. You should see a connection from the debian IP to king over that port.

A screen shot of a computer

Description automatically generated

* Now, we can also use PSSessions to copy a file from debian to your Windows. Find or create a file you’d like to copy. Then you can copy that file by putting the PSSession in a variable and using it in a copy command. This goes as follows:
  + $session=New-PSSession -ComputerName 192.168.11.50 -Credential donald@company-daan.serverlabs.be
  + copy-item *<file-to-copy>* -Destination 'c:\' -ToSession $session
  + $session | Remove-PSSession

A black screen with white text

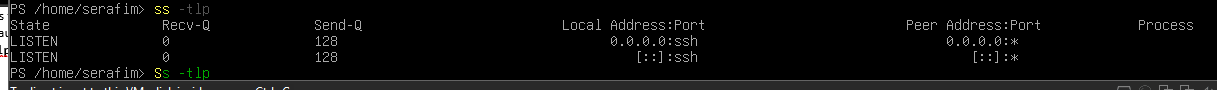
Description automatically generated

* Verify in your Windows Server that you indeed have received this file.

A screenshot of a computer

Description automatically generated

* Note that a connection in the opposite direction won’t work because no WinRM service is running on linux. You can see that no listener is on that port with ss -tlp on debian (in a bash shell).



# PowerShell remoting from Windows to linux, using SSH

Now we’ll set up PowerShell remoting from our Win11 VM to our debian VM over SSH. However, remember that we’ve limited SSH access to our debian server via TCP wrappers. Only your laptop host was given exclusive SSH access to debian.

* Change the TCP wrapper setup to allow the full 192.168.11.0/24 network to have SSH access to the debian server. (Have a look at the relevant lab about what config files to change if you forgot.)

Oh for god sake..

Change /etc/hosts.allow

We don’t run WinRM on Linux, but we do have an SSH server. And PowerShell Remoting has SSH support in PowerShell 7 (not in Windows PowerShell 5.1)!

* First, we need to configure the SSH daemon in debian to allow incoming connections via PowerShell. To this end, add the following subsystem in the config file of the ssh daemon (/etc/ssh/sshd\_config), somewhere near the end of the file:
  + Subsystem powershell /usr/bin/pwsh -sshs -NoLogo
    - Note: the -NoLogo still appears to be necessary although MS documentation claims it not to be

A screen shot of a computer screen

Description automatically generated

* Restart your ssh service using systemctl

sudo systemctl restart ssh

* Now connect from your Win11 VM to Debian using:
  + Enter-PSSession -HostName 192.168.11.10 -UserName *<username>*
    - Note: the SSH fingerprint verification message upon first connect will appear where you must confirm the identity by typing ‘yes’. Currently (a bug?) you will not see yourself typing ‘yes’, but it does get through.
    - Note: -HostName instead of -ComputerName ; -UserName instead of -Credential. What’s the difference? Look at the documentation: <https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/enter-pssession>
    - In Windows Powershell 5.1, are -HostName and -UserName available for Enter-PSSession there?

It only works (the parameters) for PowerShell 7

A screenshot of a computer

Description automatically generated

## Some other possibilities for a remote shell

# PowerShell remoting from linux to linux, using SSH

* Now, we could also do PowerShell remoting from RHEL to Debian over SSH, using Enter-PSSession in a similar way as the previous exercise. Start PowerShell on your RHEL and start a remote PowerShell session to Debian.

A screenshot of a computer

Description automatically generated

# SSH’ing from linux to Windows (via OpenSSH server on Windows)

Since Windows Server 2025, an OpenSSH server is installed by default, although not being active. (Earlier versions had to install it as optional Windows Capability.)

* Start the sshd service on KING .

Start-Service sshd

* Verify that:
  + The firewall allows incoming SSH:

Get-NetFirewallRule -Name \*ssh\* |select name,enabled,action,profile

A screenshot of a computer screen

Description automatically generated

* + There is a process which listens on the default SSH port:

Get-NetTCPConnection -State Listen



* Note the profile the firewall rule applies to. Change it to also allow SSH connection on the current profile of your server if necessary.

Set-NetFirewallRule -Name "OpenSSH-Server-In-TCP" -Profile Any

* Now test if you can ssh from your debian Linux VM to your Windows server KING (192.168.11.50).

ssh Donald@192.168.11.50

* What shell do you get at KING when you make an SSH connection from debian to KING?

it is a cmd shell

A screenshot of a computer screen

Description automatically generated

* Note that SSH server configuration settings can be made in the %programdata%\ssh\sshd\_config file. See: <https://learn.microsoft.com/en-us/windows-server/administration/OpenSSH/openssh-server-configuration> for more information.