|  |  |
| --- | --- |
| **Topic:** | Use of Network script |

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev** | **Date** | **Changes Made** | **By (User)** |
| 0.0 |  | Document Creation | Mathew Gallehawk |

# Utilisation

Here are the detailed steps:

Step 1: Copy Script to Temp File

Save the provided script to a temporary location on your computer, such as C:\temp\network\_scan.ps1

Step 2: Open PowerShell as Admin

Right-click on the Start menu and select "Windows PowerShell (Admin)" or type "powershell" in the search bar and right-click on the result to select "Run as administrator", and PowerShell window with elevated privileges should now be open

Step 3: CD to Temp File Location

In the PowerShell window, navigate to the temporary location where you saved the script using the cd command:

* Type cd C:\temp.
* Press Enter to change directories.

Step 4: Run Script

In the PowerShell window, run the script by typing its name followed by a dot (.):

* Type . .\network\_scan.ps1
* Press Enter to execute the script.

The script should now start running and gathering network device information.

Step 5: Attach Output to Ticket (Optional)

Once the script has completed, you will see output in the PowerShell window.

If required, copy and paste the output into a ticket or document for tracking purposes.

* Press CtrlA to select all text in the console window.
* Right-click on the selected text and choose "Copy" (or press CtrlC)
* Open your ticketing system or document and paste the copied output.

Step 6: Send Output to Network Engineer (Optional)

If necessary, save the script output to a file for later reference or sharing with others:

* + Press CtrlA to select all text in the console window.
  + Right-click on the selected text and choose "Save As" (or press CtrlS)
  + Choose a location and filename, then click Save to write the output to a file.

Alternatively, you can email or share the script output with the network engineer directly.

# Script analysis

Overview

The script uses the PSDiscoveryProtocol module to discover and gather information about network devices on your local network. The script performs the following steps:

1. Checks if the PSDiscoveryProtocol module is installed and installs it if necessary.

2. Captures network packets using the Invoke-DiscoveryProtocolCapture cmdlet.

3. Extracts device information from the captured packets using the Get-DiscoveryProtocolData cmdlet.

4. Outputs the extracted data to both console and file.

Step-by-Step Explanation

Module Management

The script first checks if the PSDiscoveryProtocol module is installed using the Get-Module cmdlet:

powershell

$module = Get-Module -ListAvailable -Name PSDiscoveryProtocol

If the module is not installed, the script prompts you to install it using the Install-Module cmdlet:

powershell

if ($module -eq $null) {

$input = read-host "Module PSDiscoveryProtocol is not installed. Do you want to install it? (Y/N)"

if ($input -eq "Y") {

write-host "Installing module PSDiscoveryProtocol"

Install-Module -Name PSDiscoveryProtocol

} else {

bail

}

}

Packet Capture

The script then captures network packets using the Invoke-DiscoveryProtocolCapture cmdlet:

powershell

$Packet = Invoke-DiscoveryProtocolCapture -Force

This cmdlet sends a discovery protocol packet to all devices on the local network and captures their responses.

Device Information Extraction

The script extracts device information from the captured packets using the Get-DiscoveryProtocolData cmdlet:

powershell

$dataFrame = Get-DiscoveryProtocolData -Packet $Packet

This cmdlet parses the captured packets and extracts relevant information about each device, such as its IP address, MAC address, and VLAN assignment.

Output

Finally, the script outputs the extracted data to both console and file using the output-console and output-file functions:

powershell

output-console $dataFrame

output-file $dataFrame

The output includes information about each device, such as its IP address, MAC address, VLAN assignment, and other relevant details.

Functions

The script defines two custom functions:

1. output-console: Outputs the extracted data to the console window.

2. output-file: Saves the extracted data to a file for later reference or sharing with others.

These functions are used to display and store the output in a human-readable format.