# Week 4 PowerShell

Contents

[Week 4 PowerShell 1](#_Toc145226543)

[Get Help on a Command 1](#_Toc145226544)

[Tutorial 1: System Information 1](#_Toc145226545)

[Tutorial 2: Working with the File System 4](#_Toc145226546)

[Assignment Submission 5](#_Toc145226547)

Time required: 60 minutes

**NOTE:** Perform this lab on Server1.

# Get Help on a Command

You are trying to figure out how to automate gathering information from Event Logs. A good place to start is with PowerShell’s help system.

|  |
| --- |
| # How to get help on a command  Get-Help Get-EventLog -ShowWindow |

A help window will popup. Click **Settings** 🡪 Click a Check Mark in all **Help Sections**. Click **OK**.

Look through the information displayed. You will find examples of these commands. Run both.

|  |
| --- |
| # List all event logs  Get-EventLog –list  # List 20 most recent entries from the system log  Get-EventLog -LogName System -Newest 20 |

# Tutorial 1: System Information

Windows Management Interface (WMI) is a comprehensive management framework provided by Microsoft for managing components and services in a Windows environment. It allows you to access and manipulate system settings, configurations, and information programmatically, making it a valuable tool for system administrators, including those involved in Information Technology and Cybersecurity.

In the following tutorial, we get information about our server using PowerShell and WMI.

This script is designed to retrieve and display various system information using Windows Management Instrumentation (WMI) objects. It provides details about the operating system, BIOS, processor, memory, and disk space.

A screen shot of a computer

Description automatically generated

This line uses the **Get-WmiObject** cmdlet to retrieve information about the operating system and stores it in the variable **$osInfo**. It specifically queries the **Win32\_OperatingSystem** class.



This line retrieves information about the **BIOS** (Basic Input/Output System) and stores the BIOS version in the variable **$biosInfo**.



This line collects information about the computer's processor and stores it in the variable **$cpuInfo**. It queries the **Win32\_Processor** class.



This line fetches information about physical memory (RAM) and calculates the total capacity by summing up the values. The result is stored in the variable **$memoryInfo**. It queries the **Win32\_PhysicalMemory** class and uses **Measure-Object** to calculate the sum of the **Capacity** property.



This line retrieves information about logical disks (hard drives) with a drive type of 3 (which represents local disk drives) and stores the results in the variable **$diskInfo**. It queries the **Win32\_LogicalDisk** class with a filter to select only local disk drives.

A computer screen shot of code

Description automatically generated

These lines use **Write-Host** to display specific system information obtained earlier in the script. It includes details about the operating system name and architecture, processor name, BIOS version, and memory capacity (converted from bytes to gigabytes).

A screen shot of a computer code

Description automatically generated

This section iterates through each logical disk stored in **$diskInfo** using a foreach loop. For each disk, it calculates and displays the free space in gigabytes (GB) by dividing the free space in bytes by 1GB. The result is rounded to two decimal places.

Example run:

A blue screen with white text

Description automatically generated

# Tutorial 2: Working with the File System

In this Hands-On Project, you work with PowerShell providers.

1. Right-click the Start menu and choose Windows PowerShell (Admin) to open Windows PowerShell.
2. At the prompt, type **Get-PSProvider** and press enter to view the available PowerShell providers.
3. **Insert a screenshot:**

Click or tap here to enter text.

1. Type **Get-PSDrive** and press enter to view the expanded list of PowerShell providers. Note that the filesystem provider is available for each drive letter on the system, and that registry provider is available for HKEY\_CURRENT\_USER and HKEY\_LOCAL\_MACHINE.
2. **Insert a screenshot:**

Click or tap here to enter text.

Each PowerShell provider treats each item that it works with as an object.

1. At the prompt, type **gci | Get-Member** and press enter to view the properties available for the objects within the current directory of the filesystem provider. Note that there is a PSIscontainer property that indicates that the object is a subdirectory.
2. Type **gci | Where-Object {$\_.psiscontainer}** and press enter to view only directories.
3. **Insert a screenshot:**

Click or tap here to enter text.

1. Following this, type **gci | Where-Object {!$\_.psiscontainer}** and press enter to view only non-directories (i.e., files).
2. **Insert a screenshot:**

Click or tap here to enter text.

Within the filesystem provider, you can create, edit, and remove objects, such as files and directories. Execute the following commands at the command prompt, in turn. For each one, interpret the output (referencing the aliases and cmdlets within this module, as necessary).

A screen shot of a computer code

Description automatically generated

1. **Insert a screenshot of the result:**

Click or tap here to enter text.

A screen shot of a computer screen

Description automatically generated

1. **Insert a screenshot of the result:**

Click or tap here to enter text.

A black background with text

Description automatically generated

1. **Insert a screenshot of the result:**

Click or tap here to enter text.

## Assignment Submission

1. Attach the program files.
2. Attach this completed document.
3. Submit in Blackboard.