

Windows PowerShell

Powershell is a powerful tool from Microsoft. Typically seen as the advanced, modern version of CLI (Command prompt), combining CLI, and a scripting language built on a .NET framework. Unlike older models, PowerShell is object orientated which means it can handle complex data types and interact with system components more effectively.

So to understand power shell we need to know what an **object** is.

For example a **car** has **properties**, **color**, **model**, **FuelLevel** and then **methods** such as **Drive()**, **HonkHorn()** and **Refuel()**.

What do we call the advanced approach used to develop PowerShell?

Object-Oriented

Example:

1. Start Virtual Machine
2. Select Applications - Internet - Remmina
3. Click Cancel on the prompt to ignore
4. Left drop down select "SSH"
5. Input IP and then "Enter"
6. Click "ok" and input credentials as per the machines instructions

PowerShell Basics Summary

Launching PowerShell

From Start Menu, Run dialog (Win+R → powershell), File Explorer (type powershell in address bar), Task Manager, or Command Prompt.

In the given example, it's launched from Command Prompt.

Cmdlets & Syntax

PowerShell commands are called **cmdlets** and follow a **Verb-Noun** format (e.g., Get-Content, Set-Location).

Get-Command lists all available cmdlets, functions, aliases, and scripts.

You can filter results with parameters like -CommandType Function.

Getting Help

Get-Help <cmdlet> gives usage, parameters, and examples.

Use options like -examples, -detailed, -full, or -online for more info.

Aliases

Shortcuts for cmdlets (e.g., dir → Get-ChildItem, cd → Set-Location).

View them with Get-Alias.

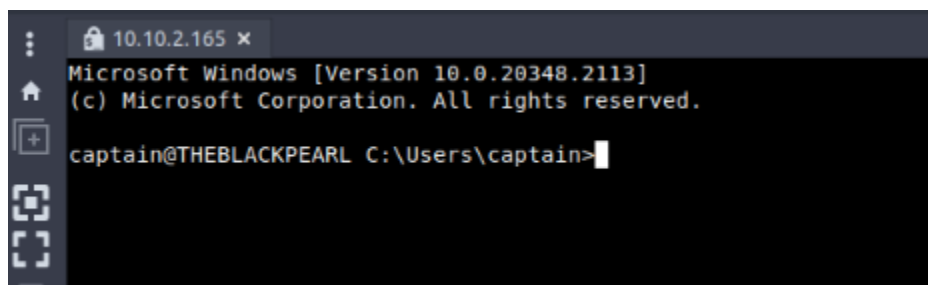
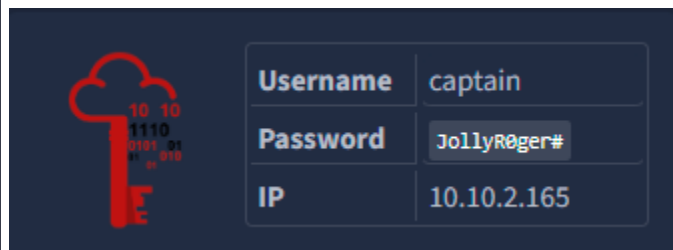
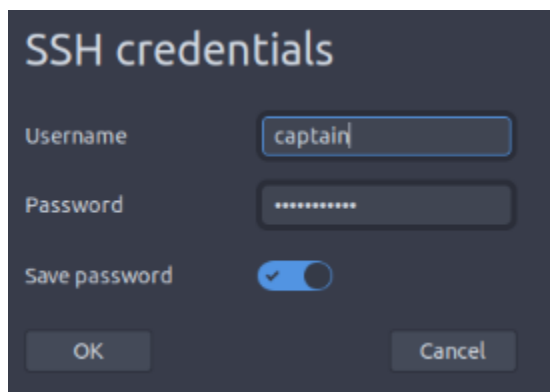
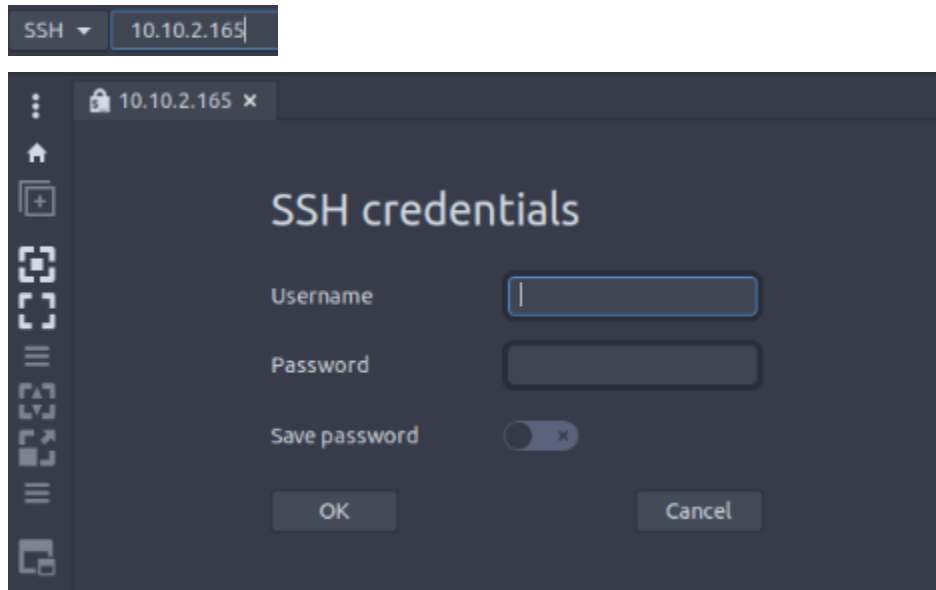
Downloading More Cmdlets

Use `Find-Module -Name "<pattern*>"` to search PowerShell Gallery (requires internet).

Install with `Install-Module -Name "<ModuleName>"`.

May require confirming trust in the repository.

Practical Example:



The black pearl! Love the pirates of the caribbean reference.

Questions:

How would you retrieve a list of commands that start with the verb “remove”?

Get-Command -Name Remove*

When cmdlet has its traditional counterpart echo as an alias?

Write-Output

What is the command to retrieve some example usage for the cmdlet New-LocalUser?

Get-Help New-LocalUser -examples

Here's the short version of that section:

List files/folders: Get-ChildItem (like dir in CMD or ls in Unix).

Change directory: Set-Location (like cd).

Create files/folders: New-Item -Path ... -ItemType Directory/File.

Delete files/folders: Remove-Item (combines del and rmdir).

Copy files/folders: Copy-Item.

Move files/folders: Move-Item.

Read a file's contents: Get-Content (like type in CMD or cat in Unix).

And for your question — instead of the Windows type command, you'd use:

Get-Content

Piping (|) in PowerShell passes **objects** (not just text) between commands, allowing more powerful processing.

Example: Get-ChildItem | Sort-Object Length → lists files sorted by size.

Filtering:

Where-Object filters based on properties (e.g., -eq, -ne, -gt, -ge, -lt, -le, -like).

Example: Get-ChildItem | Where-Object -Property Extension -eq .txt → only .txt files.

Example: Get-ChildItem | Where-Object -Property Name -like ship* → files starting with "ship".

Selecting properties: Select-Object chooses specific details (e.g., Name, Length) or limits results.

Text search: Select-String searches inside files (supports regex), similar to grep or findstr.

Pipelines can chain multiple cmdlets to sort, filter, and format output in one sequence.

This section basically teaches you: **get stuff** → **filter it** → **pick details** → **search inside** → **all in one pipeline**.

Answer the questions below

How would you retrieve the items in the current directory with size greater than 100? [for the sake of this question, avoid the use of quotes (" or ') in your answer]

Get-ChildItem | Where-Object -Property Length -gt 100

✓ Correct Answer

PowerShell provides powerful cmdlets for system administration and automation, offering more detailed output than traditional commands.

Get-ComputerInfo – Retrieves comprehensive system details (OS, hardware, BIOS, etc.) in one command.

Get-LocalUser – Lists all local user accounts with their status and description.

Get-NetIPConfiguration – Displays network interface details such as IPs, DNS servers, and gateways.

Get-NetIPAddress – Shows all configured IP addresses, including inactive ones, with detailed properties.


These tools help IT professionals quickly gather and manage critical system and network information for both local and remote machines.

Practical:

So I was tasked with finding out a few questions:

Answer the questions below

In the previous task, you found a marvellous treasure carefully hidden in the target machine. What is the hash of the file that contains it?



Uh-oh! The answer you provided may not be in English. Please review it and try again.

What property retrieved by default by `Get-NetTCPConnection` contains information about the process that has started the connection?

Submit

It's time for another small challenge. Some vital service has been installed on this pirate ship to guarantee that the captain can always navigate safely. But something isn't working as expected, and the captain wonders why. Investigating, they find out the truth, at last: the service has been tampered with! The shady lad from before has modified the service `DisplayName` to reflect his very own motto, the same that he put in his user description.

With this information and the PowerShell knowledge you have built so far, can you find the service name?

Submit

So the first one was regarding the carefully hidden treasure chest, or I believe that is it, now this is located in a separate user file, so after navigating to the user directory and selecting the p1r4t3 user, I went to documents and found the “hidden-treasure-chest” and then “big-treasure.txt” which I believe is the target for this question in particular. Using the command:

Get-FileHash big-treasure.txt

Which the response was an output of:

71FC5EC11C2497A32F8F08E61399687D90ABE6E204D2964DF589543A613F3E08

Which was the correct answer to the question!

Next we must use the command:

Get-NetTCPConnection

Which is getting the Transport Control Protocol of all active connections to our machine.

Now it's important to recognise we must do this back in our original User: captain. So I navigated back to our initial User profile:

C:\Users\captain

Get-NetTCPConnection

This in response gave a multitude of various ports that were listening or established.

To answer the question, what property contains information about the process that has started, it would actually be: **OwningProcess**

This is the process for that independent connection.

Finally we are graced with a small challenge, a vital service has been installed on the pirate ship, to guarantee the captain can always navigate safely. But something isn't working. Someone has modified the "DisplayName" to reflect his very own motto, the same that he put in his user description.

With this knowledge I did go into the captains cabin again to search the 3 txt files, which I found a flag, which was an actual pirate flag, pretty cool, I tried opening the hat.txt but it said "Don't touch my hat!" Sorry captain! Also the boots.txt contained nothing, which the file does say 0 indicating nothing but I checked regardless. So I want to now navigate to users, to see what other users there are on the machine.

We have Administrator, captain, p1r4t3 and public. I am going to investigate further into p1r4t3. But I couldn't see anything, lets try listing processes.

Using the **Get-Service** Command we can see a list of running services, as well as their display names.

Running p1r4t3-s-compass A merry life and a short one.

Pahahaha, P1r4t3-s-compass! What is this imposter doing!

After navigating the different processes running, I finally discovered the perpetrator!

Blue team uses PowerShell scripts for tasks like log analysis, detecting anomalies, extracting IOCs, malware analysis, and automated system scans.

Red team uses it for system enumeration, remote command execution, and obfuscating scripts to bypass defenses.

System administrators use it for integrity checks, configuration management, network security, and automated incident response.

Invoke-Command is a key cmdlet for running commands/scripts on local or remote systems.

Example 1: Run a local script (-FilePath) on a remote machine.

Example 2: Run commands remotely (-ScriptBlock) with optional credentials, without needing to write a script.

PowerShell scripting is a powerful skill in cyber security, useful for both defense and offense.

Conclusion:

I really see how PowerShell can be used in both an offensive and defensive scenario, for me personally it's excellent to be able to revisit these in basic terms to really solidify my understanding of what powershell is, and how I can use it to navigate running processes to identify anything fishy, as well as understand if there is any services being used as a backdoor, such as the example above.