Windows PowerShell

Learned:

- `Get-ChildItem`, `New-Item`, `Remove-Item`, and `Get-Content` allow for management of files, which is important when analyzing suspicious files or directories.

- `Get-Process` and `Get-Service` show running programs and services, which can help find suspicious activity.

- `Get-FileHash` can check if a file has been changed or tampered with.

- Piping with commands like `Where-Object` and `Sort-Object` lets you filter and organize information quickly.

Basic Cmdlets

To list all available cmdlets, functions, aliases, and scripts that can be used in PowerShell, we can use `Get-Command`.

Terminal

```
PS C:\Users\captain> Get-Command

CommandType     Name                                               Version
Source
-----------     ----                                               -------    -----
-

Alias           Add-AppPackage                                     2.0.1.0    Appx
Alias           Add-AppPackageVolume                               2.0.1.0    Appx
Alias           Add-AppProvisionedPackage                          3.0        Dism
[...]
Function        A:
Function        Add-BCDataCacheExtension                           1.0.0.0
BranchCache
Function        Add-DnsClientDohServerAddress                      1.0.0.0
DnsClient
[...]
Cmdlet          Add-AppxPackage                                    2.0.1.0    Appx
Cmdlet          Add-AppxProvisionedPackage                         3.0        Dism
Cmdlet          Add-AppxVolume                                     2.0.1.0    Appx
[...]
```

It's possible to filter the list of commands based on displayed property values. For example, if we want to display only the available commands of type "function", we can use `-CommandType "Function"`

Terminal

```
PS C:\Users\captain> Get-Command -CommandType "Function"

CommandType     Name                                               Version
Source
-----------     ----                                               -------    -----
-
Function        A:
Function        Add-BCDataCacheExtension                           1.0.0.0
BranchCache
Function        Add-DnsClientDohServerAddress                      1.0.0.0
DnsClient
Function        Add-DnsClientNrptRule                              1.0.0.0
DnsClient
[...]
```

`Get-Help` provides information about cmdlets, including usage, parameters, and examples. It's the go-to cmdlet for learning how to use PowerShell commands.

Terminal

```
PS C:\Users\captain> Get-Help Get-Date

NAME
    Get-Date

SYNOPSIS
    Gets the current date and time.

SYNTAX
    Get-Date [[-Date] <System.DateTime>] [-Day <System.Int32>] [-DisplayHint {Date
| Time | DateTime}] [-Format <System.String>] [-Hour <System.Int32>] [-Millisecond
<System.Int32>] [-Minute <System.Int32>] [-Month <System.Int32>] [-Second
<System.Int32>] [-Year <System.Int32>] [<CommonParameters>]

    Get-Date [[-Date] <System.DateTime>] [-Day <System.Int32>] [-DisplayHint {Date
| Time | DateTime}] [-Hour <System.Int32>] [-Millisecond <System.Int32>] [-Minute
<System.Int32>] [-Month <System.Int32>] [-Second <System.Int32>] [-UFormat
<System.String>] [-Year <System.Int32>] [<CommonParameters>]
```

```
DESCRIPTION
        The `Get-Date` cmdlet gets a DateTime object that represents the current
date or a date that you specify. `Get-Date` can format the date and time in several
.NET and UNIX formats. You can use `Get-Date` to generate a date or time character
string, and then send the string to other cmdlets or programs.

        `Get-Date` uses the current culture settings of the operating system to
determine how the output is formatted. To view your computer's settings, use `(Get-
Culture).DateTimeFormat`.

RELATED LINKS
    Online Version:
https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/get-
date?view=powershell-5.1&WT.mc_id=ps-gethelp
    ForEach-Object
    Get-Culture
    Get-Member
    New-Item
    New-TimeSpan
    Set-Date
    Set-Culture xref:International.Set-Culture

REMARKS
    To see the examples, type: "get-help Get-Date -examples".
    For more information, type: "get-help Get-Date -detailed".
    For technical information, type: "get-help Get-Date -full".
    For online help, type: "get-help Get-Date -online".
```

Navigating the File system

`Get-ChildItem` lists the files and directories in a location specified with the `-Path` parameter. It can be used to explore directories and view their contents. If no `Path` is specified, the cmdlet will display the content of the current working directory.

Terminal

```
PS C:\Users\captain> Get-ChildItem


    Directory: C:\Users\captain


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
```

```
d-r---          5/8/2021   9:15 AM                Desktop
d-r---          9/4/2024  10:58 AM                Documents
d-r---          5/8/2021   9:15 AM                Downloads
d-r---          5/8/2021   9:15 AM                Favorites
d-r---          5/8/2021   9:15 AM                Links
d-r---          5/8/2021   9:15 AM                Music
d-r---          5/8/2021   9:15 AM                Pictures
d-----          5/8/2021   9:15 AM                Saved Games
d-r---          5/8/2021   9:15 AM                Videos
```

To create an item in PowerShell, we can use `New-Item`. We will need to specify the path of the item and its type (whether it is a file or a directory).

Terminal

```
PS C:\Users\captain\Documents> New-Item -Path ".\captain-cabin\captain-wardrobe" -
ItemType "Directory"

    Directory: C:\Users\captain\Documents\captain-cabin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----          9/4/2024  12:20 PM                captain-wardrobe

PS C:\Users\captain\Documents> New-Item -Path ".\captain-cabin\captain-
wardrobe\captain-boots.txt" -ItemType "File"

    Directory: C:\Users\captain\Documents\captain-cabin\captain-wardrobe

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          9/4/2024  11:46 AM              0 captain-boots.txt
```

`Remove-Item` cmdlet removes both directories and files, whereas in Windows CLI we have separate commands `rmdir` and `del`.

Terminal

```
PS C:\Users\captain\Documents> Remove-Item -Path ".\captain-cabin\captain-
wardrobe\captain-boots.txt"
PS C:\Users\captain\Documents> Remove-Item -Path ".\captain-cabin\captain-wardrobe"
```

To read and display the contents of a file, we can use the `Get-Content` cmdlet, which works similarly to the `type` command in Command Prompt (or `cat` in Unix-like systems).

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-Content -Path ".\captain-hat.txt"
 _           _
| |         | |
| |__    __ _| |_
| '_ \ / _ | __|
| | | | (_| | |_
|_| |_|\__,_|\__|


Don't touch my hat!
```

To navigate to a different directory, we can use the `Set-Location` cmdlet. It changes the current directory, bringing us to the specified path, akin to the `cd` command in Command Prompt.

Terminal

```
PS C:\Users\captain> Set-Location -Path ".\Documents"
PS C:\Users\captain\Documents>
```

Piping and Sorting Data

Piping is a technique used in command-line environments that allows the output of one command to be used as the input for another. This creates a sequence of operations where the data flows from one command to the next.

For example, if you want to get a list of files in a directory and then sort them by size, you could use the following command in PowerShell:

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-ChildItem | Sort-Object Length

    Directory: C:\Users\captain\Documents\captain-cabin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          9/4/2024  12:50 PM              0 captain-boots.txt
-a----          9/4/2024  12:14 PM            264 captain-hat2.txt
-a----          9/4/2024  12:14 PM            264 captain-hat.txt
-a----          9/4/2024  12:37 PM           2116 ship-flag.txt
d-----          9/4/2024  12:50 PM                captain-wardrobe
```

`Get-ChildItem` retrieves the files (as objects), and the pipe (`|`) sends those file objects to `Sort-Object`, which then sorts them by their `Length` (size) property.

We can use the `Where-Object` cmdlet. For instance, to list only `.txt` files in a directory, we can use:

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-ChildItem | Where-Object -Property
"Extension" -eq ".txt"

    Directory: C:\Users\captain\Documents\captain-cabin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          9/4/2024  12:50 PM              0 captain-boots.txt
-a----          9/4/2024  12:14 PM            264 captain-hat.txt
-a----          9/4/2024  12:14 PM            264 captain-hat2.txt
-a----          9/4/2024  12:37 PM           2116 ship-flag.txt
```

`Where-Object` filters the files by their `Extension` property, ensuring that only files with extension equal (`-eq`) to `.txt` are listed.

The operator `-eq` (i.e. "**equal to**") is part of a set of **comparison operators** that are shared with other scripting languages (e.g. Bash, Python). To show PowerShell's filtering.

Other Operators

- `-ne`: "**not equal**". This operator can be used to exclude objects from the results based on specified criteria.
- `-gt`: "**greater than**". This operator will filter only objects which exceed a specified value. It is important to note that this is a strict comparison, meaning that objects that are equal to the specified value will be excluded from the results.
- `-ge`: "**greater than or equal to**". This is the non-strict version of the previous operator. A combination of `-gt` and `-eq`.
- `-lt`: "**less than**". Like its counterpart, "greater than", this is a strict operator. It will include only objects which are strictly below a certain value.
- `-le`: "**less than or equal to**". Just like its counterpart `-ge`, this is the non-strict version of the previous operator. A combination of `-lt` and `-eq`.

objects can also be filtered by selecting properties that match (`-like`) a specified pattern:

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-ChildItem | Where-Object -Property
"Name" -like "ship*"

    Directory: C:\Users\captain\Documents\captain-cabin
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          9/4/2024  12:37 PM           2116 ship-flag.txt
```

`Select-Object`, is used to select specific properties from objects or limit the number of objects returned. It's useful for refining the output to show only the details.

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-ChildItem | Select-Object
Name,Length

Name                Length
----                ------
captain-wardrobe
captain-boots.txt 0
captain-hat.txt    264
captain-hat2.txt   264
ship-flag.txt      2116
```

`Select-String` cmdlet searches for text patterns within files. It's commonly used for finding specific content within log files or documents.

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Select-String -Path ".\captain-
hat.txt" -Pattern "hat"

captain-hat.txt:8:Don't touch my hat!
```

System and Network

The `Get-ComputerInfo` cmdlet retrieves comprehensive system information, including operating system information, hardware specifications, BIOS details, and more. It provides a snapshot of the entire system configuration in a single command. Its traditional counterpart `systeminfo` retrieves only a small set of the same details.

Terminal

```
PS C:\Users\captain> Get-ComputerInfo

WindowsBuildLabEx                                       :
20348.859.amd64fre.fe_release_svc_prod2.220707-1832
WindowsCurrentVersion                                   : 6.3
```

```
WindowsEditionId                                        : ServerDatacenter
WindowsInstallationType                                 : Server Core
WindowsInstallDateFromRegistry                          : 4/23/2024 6:36:29 PM
WindowsProductId                                        : 00454-60000-00001-AA763
WindowsProductName                                      : Windows Server 2022
Datacenter
[...]
```

`Get-LocalUser` lists all the local user accounts on the system. The default output displays, for each user, username, account status, and description.

Terminal

```
PS C:\Users\captain> Get-LocalUser

Name               Enabled Description
----               ------- -----------
Administrator      True    Built-in account for administering the computer/domain
captain            True    The beloved captain of this pirate ship.
DefaultAccount     False   A user account managed by the system.
Guest              False   Built-in account for guest access to the computer/domain
WDAGUtilityAccount False   A user account managed and used by the system for
Windows De
```

Similar to the traditional `ipconfig` command, the following two cmdlets can be used to retrieve detailed information about the system's network configuration.

`Get-NetIPConfiguration` provides detailed information about the network interfaces on the system, including IP addresses, DNS servers, and gateway configurations.

Terminal

```
PS C:\Users\captain> Get-NetIPConfiguration

InterfaceAlias       : Ethernet
InterfaceIndex       : 5
InterfaceDescription : Amazon Elastic Network Adapter
NetProfile.Name      : Network 3
IPv4Address          : 10.10.178.209
IPv6DefaultGateway   :
IPv4DefaultGateway   : 10.10.0.1
DNSServer            : 10.0.0.2
```

In case we need specific details about the IP addresses assigned to the network interfaces, the `Get-NetIPAddress` cmdlet will show details for all IP addresses configured on the system, including those that are not currently active.

Terminal

```
PS C:\Users\captain> Get-NetIPAddress

IPAddress         : fe80::3fef:360c:304:64e%5
InterfaceIndex    : 5
InterfaceAlias    : Ethernet
AddressFamily     : IPv6
Type              : Unicast
PrefixLength      : 64
PrefixOrigin      : WellKnown
SuffixOrigin      : Link
AddressState      : Preferred
ValidLifetime     : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource      : False
PolicyStore       : ActiveStore

IPAddress         : ::1
InterfaceIndex    : 1
InterfaceAlias    : Loopback Pseudo-Interface 1
AddressFamily     : IPv6
[...]

IPAddress         : 10.10.178.209
InterfaceIndex    : 5
InterfaceAlias    : Ethernet
AddressFamily     : IPv4
[...]

IPAddress         : 127.0.0.1
InterfaceIndex    : 1
InterfaceAlias    : Loopback Pseudo-Interface 1
AddressFamily     : IPv4
[...]
```

To gather more advanced system information, especially concerning dynamic aspects like running processes, services, and active network connections, we can leverage a set of cmdlets that go beyond static machine details.

`Get-Process` provides a detailed view of all currently running processes, including CPU and memory usage, making it a powerful tool for monitoring and troubleshooting.

Terminal

```
PS C:\Users\captain> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
     67       5      872        500       0.06   2340   0 AggregatorHost
     55       5      712       2672       0.02   3024   0
AM_Delta_Patch_1.417.483.0
    309      13    18312       1256       0.52   1524   0 amazon-ssm-agent
     78       6     4440        944       0.02    516   0 cmd
     94       7     1224       1744       0.31    568   0 conhost
[...]
```

Similarly, `Get-Service` allows the retrieval of information about the status of services on the machine, such as which services are running, stopped, or paused. It is used extensively in troubleshooting by system administrators, but also by forensics analysts hunting for anomalous services installed on the system.

Terminal

```
PS C:\Users\captain> Get-Service

Status   Name               DisplayName
------   ----               -----------
Stopped  Amazon EC2Launch   Amazon EC2Launch
Running  AmazonSSMAgent     Amazon SSM Agent
Stopped  AppIDSvc           Application Identity
Running  BFE                Base Filtering Engine
Running  CertPropSvc        Certificate Propagation
Stopped  ClipSVC            Client License Service (ClipSVC)
[...]
```

To monitor active network connections, `Get-NetTCPConnection` displays current TCP connections, giving insights into both local and remote endpoints. This useful during an incident response or malware analysis task, as it can uncover hidden backdoors or established connections towards an attacker-controlled server.

Terminal

```
PS C:\Users\captain> Get-NetTCPConnection

LocalAddress        LocalPort RemoteAddress       RemotePort State
AppliedSetting OwningProcess
------------        --------- -------------       ---------- -----       ----------
```

```
---- -------------
[...]
::                   22       ::                   0          Listen
1444
10.10.178.209        49695    199.232.26.172       80         TimeWait
0
0.0.0.0              49668    0.0.0.0              0          Listen
424
0.0.0.0              49667    0.0.0.0              0          Listen
652
0.0.0.0              49666    0.0.0.0              0          Listen
388
0.0.0.0              49665    0.0.0.0              0          Listen
560
0.0.0.0              49664    0.0.0.0              0          Listen
672
0.0.0.0              3389     0.0.0.0              0          Listen
980
10.10.178.209        139      0.0.0.0              0          Listen
4
0.0.0.0              135      0.0.0.0              0          Listen
908
10.10.178.209        22       10.14.87.60          53523      Established Internet
1444
0.0.0.0              22       0.0.0.0              0          Listen
```

`Get-FileHash` generates file hashes, which is valuable in incident response, threat hunting, and malware analysis, as it helps verify file integrity and detect potential tampering.

Terminal

```
PS C:\Users\captain\Documents\captain-cabin> Get-FileHash -Path .\ship-flag.txt

Algorithm       Hash                          Path
---------       ----                          ----
SHA256          54D2EC3C12BF3D[...]           C:\Users\captain\Documents\captain-
cabin\ship
```

Scripting

Scripting is the process of writing and executing a series of commands contained in a text file, known as a script, to automate tasks that one would generally perform manually in a shell, like PowerShell.

`Invoke-Command` is essential for executing commands on remote systems, making it fundamental for system administrators, security engineers and penetration testers. `Invoke-Command` enables efficient remote management and—combining it with scripting—automation of tasks across multiple machines. It can also be used to execute payloads or commands on target systems during an engagement by penetration testers—or attackers.

`Get-Help` "examples" page:

Terminal

```
PS C:\Users\captain> Get-Help Invoke-Command -examples

NAME
    Invoke-Command

SYNOPSIS
    Runs commands on local and remote computers.

    ------------- Example 1: Run a script on a server -------------

    Invoke-Command -FilePath c:\scripts\test.ps1 -ComputerName Server01

    The FilePath parameter specifies a script that is located on the local
computer. The script runs on the remote computer and the results are returned to
the local computer.

    --------- Example 2: Run a command on a remote server ---------

    Invoke-Command -ComputerName Server01 -Credential Domain01\User01 -ScriptBlock
{ Get-Culture }

    The ComputerName parameter specifies the name of the remote computer. The
Credential parameter is used to run the command in the security context of
Domain01\User01, a user who has permission to run commands. The ScriptBlock
parameter specifies the command to be run on the remote computer.

    In response, PowerShell requests the password and an authentication method for
the User01 account. It then runs the command on the Server01 computer and returns
the result.
[...]
```