CloudCommand

Cyber Security

CloudCommand GmbH chr.schumacher@gmx.tm

CloudCommand GmbH chr.schumacher@gmx.tm

AGENDA

01 Einführung
02 Unterschiede zu Befehlszeilen
03 Features

01 Einführung

Was ist PowerShell

PowerShell besteht aus zwei Teilen:

1. einer Befehlszeilen-Shell und
2. einer Skriptsprache.

Ursprünglich war PowerShell als Framework für die Automatisierung administrativer Aufgaben unter Windows geplant. Seitdem hat sich PowerShell zu einem plattformübergreifenden Tool entwickelt, das für viele Arten von Aufgaben und Tasks eingesetzt wird.

Was ist PowerShell

Im Folgenden sind einige Vorteile einer Konsole gegenüber einer grafischen Oberfläche aufgeführt:

- Die Interaktion mit einer Konsole ist oft schneller als die mit einer grafischen Benutzeroberfläche.
- In einer Konsole können Sie Befehle in Batches ausführen. Konsolen eignen sich daher ideal für die Testautomatisierung.
- Sie können Befehle und Skripte in einer Textdatei speichern und ein Quellcode-Verwaltungssystem verwenden. Dies ist wahrscheinlich einer der größten Vorteile, denn so werden Ihre Befehle wiederholbar und überprüfbar. In vielen Systemen (insbesondere Behördensystemen) muss alles nachverfolgt und ausgewertet oder auditiert werden. Diese Überprüfungen decken jegliche Änderungen ab, von Datenbankänderungen bis hin zu skriptbasierten Änderungen.

Features einer Shell

- **Integriertes Hilfesystem:** Die meisten Shells verfügen über eine Art von Hilfesystem, mit dem Sie sich eingehender über Befehle informieren können.
- **Pipeline:** Herkömmliche Shells verwenden eine Pipeline, um viele Befehle der Reihe nach auszuführen. Die Ausgabe eines Befehls stellt dabei die Eingabe des nächsten Befehls dar.
- **Aliase:** Aliase sind alternative Namen, über die Befehle ausgeführt werden können.

02 Unterschiede zu Befehlszeilen

Unterschied zu herkömmlichen Befehlszeilen

PowerShell arbeitet mit **Objekten** anstelle von Text.

- In einer Befehlszeilenshell müssen Sie Skripts ausführen, deren Ausgabe und Eingabe sich möglicherweise unterscheiden, sodass Sie zu guter Letzt Zeit aufwenden müssen, um die Ausgabe zu formatieren und die benötigten Daten zu extrahieren. In PowerShell verwenden Sie hingegen Objekte als Ein- und Ausgabe. Dies bedeutet, dass Sie Zeit beim Formatieren und Extrahieren sparen.

Unterschied zu herkömmlichen Befehlszeilen

PowerShell verfügt über **Cmdlets**.

- Befehle in PowerShell werden Cmdlets genannt (ausgesprochen wie commandlets). Im Gegensatz zu vielen anderen Shellumgebungen bauen Cmdlets in PowerShell auf einer gemeinsamen Runtime und nicht auf separaten ausführbaren Dateien auf. Diese Eigenschaft ermöglicht einen konsistenten Ablauf bei der Parameteranalyse und im Pipelineverhalten. Cmdlets nehmen in der Regel Objekteingaben und Rückgabe Objekte. Die Kern-Cmdlets in PowerShell werden in .NET Core erstellt und sind Open Source. Sie können PowerShell erweitern, indem Sie weitere Cmdlets, Skripts und Funktionen aus der Community und anderen Quellen verwenden, oder Sie können eigene Cmdlets in .NET Core oder PowerShell erstellen.

Unterschied zu herkömmlichen Befehlszeilen

PowerShell verfügt über **viele Arten von Befehlen**.

- Befehle in PowerShell können native ausführbare Dateien, Cmdlets, Funktionen, Skripts oder Aliase sein. Jeder Befehl, den Sie ausführen, gehört zu einem dieser Typen. Die Wörter Befehl und Cmdlet werden häufig synonym verwendet, weil ein Cmdlet eine Art von Befehl ist.

03 Features

Aufbau von Befehlen/Features einer Shell

- In der PowerShell-Installation sind Tausende von Cmdlets verfügbar. Die Herausforderung besteht darin, die Cmdlets und ihre Anwendungsfälle zu erlernen.
- Cmdlets werden nach einem Verb-Substantiv-Benennungs-Standard benannt. Dieses Muster kann helfen, ihren Zweck zu verstehen, und vereinfacht die Suche nach ihnen. Außerdem ermöglicht es Cmdlet-Entwicklern, konsistente Namen zu erstellen.
- Die Liste der genehmigten Verben kann mithilfe des Cmdlets Get-Verb abgerufen werden. Die Verben sind nach Aktivitätstyp und Funktion sortiert.
    - z.B.: Add, Get, New, Set, ...

Suchen von Befehlen/Features einer Shell

Es gibt zwei Kern-Cmdlets, mit denen Sie erforschen können, welche Cmdlets es gibt und welchen Zweck sie erfüllen:

- **Get-Command:** Das Cmdlet Get-Command listet alle in Ihrem System verfügbaren Cmdlets auf. Filtern Sie die Liste, um schnell den benötigten Befehl zu finden.
- **Get-Help:** Mit dem Kern-Cmdlet Get-Help können Sie ein integriertes Hilfesystem aufrufen. Sie können ebenso einen Alias-Befehl help aufrufen, um Get-Help aufzurufen, aber zugleich die Leseansicht zu verbessern, weil die Ansicht seitenweise ausgegeben wird.

Get-Help/Features einer Shell

- Das Cmdlet **Get-Help** zeigt Informationen zu PowerShell-Konzepten und -Befehlen an, einschließlich Cmdlets, Funktionen, Aliassen, Skripten und einiges mehr.
- Verwenden Sie Get-Help „Befehl“, um eine Hilfe zu einem bestimmten Befehl zu erhalten.
    - Mithilfe eines * erhält man alle zugehörigen Befehle angezeigt
- Mit dem Parameter –examples werden konkrete Beispiele angezeigt

Get-Help/Features einer Shell

```
PS C:\Users\Patrick> get-help *command

Name                              Category  Module                    Synopsis
----                              --------  ------                    --------
Get-Command                       Cmdlet    Microsoft.PowerShell.Core Gets all commands.
Invoke-Command                    Cmdlet    Microsoft.PowerShell.Core Runs commands on local and
Measure-Command                   Cmdlet    Microsoft.PowerShell.U... Measures the time it takes
Show-Command                      Cmdlet    Microsoft.PowerShell.U... Displays PowerShell comman
Trace-Command                     Cmdlet    Microsoft.PowerShell.U... Configures and starts a tr
SafeGetCommand                    Function  Pester                    ...
Find-Command                      Function  PowerShellGet             ...
```

```
PS C:\Users\Patrick> get-help get-command -examples

NAME
    Get-Command

ÜBERSICHT
    Gets all commands.


    -------- Example 1: Get cmdlets, functions, and aliases --------

    Get-Command


    -------- Example 2: Get commands in the current session --------

    Get-Command -ListImported


    ------- Example 3: Get cmdlets and display them in order -------

    Get-Command -Type Cmdlet | Sort-Object -Property Noun | Format-Table -GroupBy Noun
```

Get-Command/Features einer Shell

- Wenn Sie das Cmdlet **Get-Command** ausführen, erhalten Sie eine Liste aller in PowerShell installierten Befehle. Da Tausende von Befehlen installiert sind, benötigen Sie eine Möglichkeit zum Filtern der Antwort, um den benötigten Befehl schnell zu finden.
- Bewährt hat sich die Nutzung des *
- Mit der Pipe | und Select-Object lässt sich die Ausgabe formatieren

Get-Command/Features einer Shell

```
PS C:\Users\Patrick> get-command *policy

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Get-DnsClientNrptPolicy                            1.0.0.0    DnsClient
Function        Get-NetPrefixPolicy                                1.0.0.0    NetTCPIP
Function        Get-NetQosPolicy                                   2.0.0.0    NetQos
Function        Get-VolumeScrubPolicy                              2.0.0.0    Storage
Function        New-NetQosPolicy                                   2.0.0.0    NetQos
Function        Remove-NetQosPolicy                                2.0.0.0    NetQos
Function        Set-NetQosPolicy                                   2.0.0.0    NetQos
Function        Set-VolumeScrubPolicy                              2.0.0.0    Storage
Cmdlet          ConvertFrom-CIPolicy                               1.0        ConfigCI
Cmdlet          ConvertTo-ProcessMitigationPolicy                  1.0.12     ProcessMitigations
Cmdlet          Get-AppLockerPolicy                                2.0.0.0    AppLocker
Cmdlet          Get-CertificateAutoEnrollmentPolicy                1.0.0.0    PKI
Cmdlet          Get-CIPolicy                                       1.0        ConfigCI
Cmdlet          Get-ExecutionPolicy                                3.0.0.0    Microsoft.PowerShell.Security
Cmdlet          Get-HgsAttestationBaselinePolicy                   1.0.0.0    HgsClient
Cmdlet          Get-NonRemovableAppsPolicy                         3.0        Dism
Cmdlet          Get-SecureBootPolicy                               2.0.0.0    SecureBoot
Cmdlet          Get-WheaMemoryPolicy                               2.0.0.0    Whea
Cmdlet          Merge-CIPolicy                                     1.0        ConfigCI
Cmdlet          New-AppLockerPolicy                                2.0.0.0    AppLocker
Cmdlet          New-CIPolicy                                       1.0        ConfigCI
Cmdlet          Set-AppBackgroundTaskResourcePolicy                1.0.0.0    AppBackgroundTask
Cmdlet          Set-AppLockerPolicy                                2.0.0.0    AppLocker
Cmdlet          Set-CertificateAutoEnrollmentPolicy                1.0.0.0    PKI
Cmdlet          Set-ExecutionPolicy                                3.0.0.0    Microsoft.PowerShell.Security
Cmdlet          Set-NonRemovableAppsPolicy                         3.0        Dism
Cmdlet          Set-VMSecurityPolicy                               2.0.0.0    Hyper-V
Cmdlet          Set-WheaMemoryPolicy                               2.0.0.0    Whea
Cmdlet          Test-AppLockerPolicy                               2.0.0.0    AppLocker
```

Get-Command/Features einer Shell

```
PS C:\Users\Patrick> get-command *policy | Select-Object name,source

Name                                Source
----                                ------
Get-DnsClientNrptPolicy             DnsClient
Get-NetPrefixPolicy                 NetTCPIP
Get-NetQosPolicy                    NetQos
Get-VolumeScrubPolicy               Storage
New-NetQosPolicy                    NetQos
Remove-NetQosPolicy                 NetQos
Set-NetQosPolicy                    NetQos
Set-VolumeScrubPolicy               Storage
ConvertFrom-CIPolicy                ConfigCI
ConvertTo-ProcessMitigationPolicy   ProcessMitigations
```

DANKE!

Gibt es noch Fragen?

CloudCommand