6

Powershell Overview

Powershell is the Windows Scripting Language and shell environment that is built using the .NET framework.

This also allows Powershell to execute .NET functions directly from it's shell. Most Powershell commands, called cmdlets, are written in .NET. Unlike other scripting languages and shell environments, the output of these cmdlets and objects - making Powershell somewhat object oriented. This also means that running cmdlets allows you to perform actions on the output object (which makes it convenient to pass output from one cmdlet to another). The normal format of a cmdlet is represented using Verb-Noun; for example the cmdlet to list commands is Get-Command.

Common verbs to include

- Get
- Start
- Stop
- Read
- Write
- New
- Out

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\CMNatic> 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
Alias           Add-ProvisionedAppPackage                          3.0        Dism
Alias           Add-ProvisionedAppxPackage                         3.0        Dism
Alias           Add-ProvisioningPackage                            3.0        Provisioning
Alias           Add-TrustedProvisioningCertificate                 3.0        Provisioning
Alias           Apply-WindowsUnattend                              3.0        Dism
Alias           Disable-PhysicalDiskIndication                     2.0.0.0    Storage
Alias           Disable-StorageDiagnosticLog                       2.0.0.0    Storage
Alias           Dismount-AppPackageVolume                          2.0.1.0    Appx
Alias           Enable-PhysicalDiskIndication                      2.0.0.0    Storage
Alias           Enable-StorageDiagnosticLog                        2.0.0.0    Storage
Alias           Flush-Volume                                       2.0.0.0    Storage
Alias           Get-AppPackage                                     2.0.1.0    Appx
Alias           Get-AppPackageDefaultVolume                        2.0.1.0    Appx
Alias           Get-AppPackageLastError                            2.0.1.0    Appx
Alias           Get-AppPackageLog                                  2.0.1.0    Appx
Alias           Get-AppPackageManifest                             2.0.1.0    Appx
Alias           Get-AppPackageVolume                               2.0.1.0    Appx
Alias           Get-AppProvisionedPackage                          3.0        Dism
Alias           Get-DiskSNV                                        2.0.0.0    Storage
Alias           Get-PhysicalDiskSNV                                2.0.0.0    Storage
Alias           Get-ProvisionedAppPackage                          3.0        Dism
Alias           Get-ProvisionedAppxPackage                         3.0        Dism
Alias           Get-StorageEnclosureSNV                            2.0.0.0    Storage
Alias           Initialize-Volume                                  2.0.0.0    Storage
Alias           Mount-AppPackageVolume                             2.0.1.0    Appx
Alias           Move-AppPackage                                    2.0.1.0    Appx
Alias           Move-SmbClient                                     2.0.0.0    SmbWitness
Alias           Optimize-AppProvisionedPackages                    3.0        Dism
Alias           Optimize-ProvisionedAppPackages                    3.0        Dism
Alias           Optimize-ProvisionedAppxPackages                   3.0        Dism
Alias           Remove-AppPackage                                  2.0.1.0    Appx
Alias           Remove-AppPackageVolume                            2.0.1.0    Appx
Alias           Remove-AppProvisionedPackage                       3.0        Dism
Alias           Remove-EtwTraceSession                             1.0.0.0    EventTracingManagement
Alias           Remove-ProvisionedAppPackage                       3.0        Dism
Alias           Remove-ProvisionedAppxPackage                      3.0        Dism
Alias           Remove-ProvisioningPackage                         3.0        Provisioning
Alias           Remove-TrustedProvisioningCertificate              3.0        Provisioning
```

Introduction To Powershell Basics

Now that we've understood how cmdlets works - let's explore how to use them! The main thing to remember here is that Get-Command and Get-Help are your best friends!

Using Get-Help

Get-Help displays information about a cmdlet. To get help about a particular command, run the following.

Get-Help Command-Name

You can also understand how exactly to use the command by passing in the -examples flag. This would return output like the following

```
PS C:\Users\Administrator> Get-Help Get-Command -Examples

NAME
    Get-Command

SYNOPSIS
    Gets all commands.


    Example 1: Get cmdlets, functions, and aliases

    PS C:\>Get-Command

    This command gets the Windows PowerShell cmdlets, functions, and aliases that are installed on the computer.
    Example 2: Get commands in the current session

    PS C:\>Get-Command -ListImported

    This command uses the ListImported parameter to get only the commands in the current session.
    Example 3: Get cmdlets and display them in order
```

Get-Help Get-Command Examples

Using Get-Command

Get-Command gets all the cmdlets installed on the current device. The great thing about this cmdlet is that it allows for pattern matching like the following.

Get-Command Verb-* or Get-Command *-Noun*

Running the Get-Command New- to view all the cmdlets for the verb new displays the following.

```
PS C:\Users\Administrator> Get-Command New-*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Alias           New-AWSCredentials                                 3.3.563.1  AWSPowerShell
Alias           New-EC2FlowLogs                                    3.3.563.1  AWSPowerShell
Alias           New-EC2Hosts                                       3.3.563.1  AWSPowerShell
Alias           New-RSTags                                         3.3.563.1  AWSPowerShell
Alias           New-SGTapes                                        3.3.563.1  AWSPowerShell
Function        New-AutologgerConfig                               1.0.0.0    EventTracingManagement
Function        New-DAEntryPointTableItem                          1.0.0.0    DirectAccessClientComponents
Function        New-DscChecksum                                    1.1        PSDesiredStateConfiguration
Function        New-EapConfiguration                               2.0.0.0    VpnClient
Function        New-EtwTraceSession                                1.0.0.0    EventTracingManagement
Function        New-FileShare                                      2.0.0.0    Storage
Function        New-Fixture                                        3.4.0      Pester
Function        New-Guid                                           3.1.0.0    Microsoft.PowerShell.Utility
Function        New-IscsiTargetPortal                              1.0.0.0    iSCSI
Function        New-IseSnippet                                     1.0.0.0    ISE
Function        New-MaskingSet                                     2.0.0.0    Storage
Function        New-NetAdapterAdvancedProperty                     2.0.0.0    NetAdapter
Function        New-NetEventSession                                1.0.0.0    NetEventPacketCapture
Function        New-NetFirewallRule                                2.0.0.0    NetSecurity
Function        New-NetIPAddress                                   1.0.0.0    NetTCPIP
Function        New-NetIPHttpsConfiguration                        1.0.0.0    NetworkTransition
Function        New-NetIPsecDospSetting                            2.0.0.0    NetSecurity
Function        New-NetIPsecMainModeCryptoSet                      2.0.0.0    NetSecurity
Function        New-NetIPsecMainModeRule                           2.0.0.0    NetSecurity
Function        New-NetIPsecPhase1AuthSet                          2.0.0.0    NetSecurity
Function        New-NetIPsecPhase2AuthSet                          2.0.0.0    NetSecurity
Function        New-NetIPsecQuickModeCryptoSet                     2.0.0.0    NetSecurity
Function        New-NetIPsecRule                                   2.0.0.0    NetSecurity
```

*Get-Command New-**

Object Manipulation

In the previous task, we saw how the output of every cmdlet is an object. If we want to actually manipulate the output, we need to figure out a few things.

- Passing ouput to other cmdlets
- Using specific object cmdlets to extract information

The Pipeline(|) is used to pass output from one cmdlet to another. A major difference compared to other shells is that instead of passing text or string to the command after the pipe, powershell passes an object to the next cmdlet. Like every object in object oriented frameworks, an object will contain methods and properties. You can think of methods as functions that can be applied to output from the cmdlet and you can think of properties as variables in the output from a cmdlet. To view these

details, pass the output of a cmdlet to the Get-Member cmdlet.
Verb-Noun | Get-Member

An example of running to view the members for Get-Command.

Get-Command | Get-Member -MemberType Method

```
PS C:\Users\Administrator> Get-Command | Get-Member -MemberType Method


   TypeName: System.Management.Automation.AliasInfo

Name             MemberType Definition
----             ---------- ----------
Equals           Method     bool Equals(System.Object obj)
GetHashCode      Method     int GetHashCode()
GetType          Method     type GetType()
ResolveParameter Method     System.Management.Automation.ParameterMetadata ResolveParameter(string name)
ToString         Method     string ToString()


   TypeName: System.Management.Automation.FunctionInfo

Name             MemberType Definition
----             ---------- ----------
Equals           Method     bool Equals(System.Object obj)
GetHashCode      Method     int GetHashCode()
GetType          Method     type GetType()
ResolveParameter Method     System.Management.Automation.ParameterMetadata ResolveParameter(string name)
ToString         Method     string ToString()


   TypeName: System.Management.Automation.CmdletInfo

Name             MemberType Definition
----             ---------- ----------
Equals           Method     bool Equals(System.Object obj)
GetHashCode      Method     int GetHashCode()
GetType          Method     type GetType()
ResolveParameter Method     System.Management.Automation.ParameterMetadata ResolveParameter(string name)
ToString         Method     string ToString()
```

Get-Command | Get-Member -MemberType Method

From the above flag in the command, you can see that you can also select between methods and properties.

Creating Objects From Previous cmdlets
One way of manipulating objects is pulling out the properties from the output of a cmdlet and creating a new object. This is done using the Select-Object cmdlet.
Here's an example of listing the directories and just selecting the mode and the name.

```
PS C:\Users\Administrator> Get-ChildItem | Select-Object -Property Mode, Name

Mode   Name
----   ----
d-r--- Contacts
d-r--- Desktop
d-r--- Documents
d-r--- Downloads
d-r--- Favorites
d-r--- Links
d-r--- Music
d-r--- Pictures
d-r--- Saved Games
d-r--- Searches
d-r--- Videos
```

Get-ChildItem | Select-Object -Property Mode, Name

You can also use the following flags to select particular information.

first - gets the first x object

last - gets the last x object

unique - shows the unique objects

skip - skips x objects

Filtering Objects

When retrieving output objects, you may want to select objects that match a very specific value. You can do this using the Where-Object to filter based on the value of properties.

The general format using this cmdlet is

Verb-Noun | Where-Object -Property PropertyName -operator Value

Verb-Noun | Where-Object {$*.PropertyName -operator Value}*

The second version uses the $ operator to iterate through every object passed to the Where-Object cmdlet.

Where -operator is a list of the of the following operators.

Contains - If any item in the property value is an exact match for the specified value/

EQ - If the property value is the same as the specified value.

GT - If the property value is greater than the specified value

For a full list of operators, use this link.

Here's an example of checking the stopped processes:

```
PS C:\Users\Administrator> Get-Service | Where-Object -Property Status -eq Stopped

Status   Name               DisplayName
------   ----               -----------
Stopped  AJRouter           AllJoyn Router Service
Stopped  ALG                Application Layer Gateway Service
Stopped  AppIDSvc           Application Identity
Stopped  AppMgmt            Application Management
Stopped  AppReadiness       App Readiness
Stopped  AppVClient         Microsoft App-V Client
Stopped  AppXSvc            AppX Deployment Service (AppXSVC)
Stopped  AudioEndpointBu... Windows Audio Endpoint Builder
Stopped  Audiosrv           Windows Audio
Stopped  AxInstSV           ActiveX Installer (AxInstSV)
Stopped  BITS               Background Intelligent Transfer Ser...
Stopped  Browser            Computer Browser
Stopped  bthserv            Bluetooth Support Service
Stopped  CDPSvc             Connected Devices Platform Service
Stopped  cfn-hup            CloudFormation cfn-hup
Stopped  ClipSVC            Client License Service (ClipSVC)
Stopped  COMSysApp          COM+ System Application
Stopped  CscService         Offline Files
Stopped  DcpSvc             DataCollectionPublishingService
Stopped  defragsvc          Optimize drives
Stopped  DeviceAssociati... Device Association Service
Stopped  DeviceInstall      Device Install Service
Stopped  DevQueryBroker     DevQuery Background Discovery Broker
Stopped  diagnosticshub.... Microsoft (R) Diagnostics Hub Stand...
Stopped  DiagTrack          Connected User Experiences and Tele...
Stopped  DmEnrollmentSvc    Device Management Enrollment Service
Stopped  dmwappushservice   dmwappushsvc
Stopped  dot3svc            Wired AutoConfig
Stopped  DsmSvc             Device Setup Manager
Stopped  DsSvc              Data Sharing Service
Stopped  Eaphost            Extensible Authentication Protocol
```

Get-Service | Where-Object -Property Status -eq Stopped

Sort Object

When a cmdlet outputs a lot of information, you may need to sort it to extract the information more efficiently. You do this by pipe lining the output of a cmdlet to the Sort-Object cmdlet.

The format of the command would be
Verb-Noun | Sort-Object

Here's an example of sorting the list of directories.

```
PS C:\Users\Administrator> Get-ChildItem | Sort-Object


    Directory: C:\Users\Administrator


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-r---        10/3/2019   5:11 PM                Contacts
d-r---        10/3/2019   5:11 PM                Desktop
d-r---        10/3/2019   5:11 PM                Documents
d-r---        10/3/2019   5:11 PM                Downloads
d-r---        10/3/2019   5:11 PM                Favorites
d-r---        10/3/2019   5:11 PM                Links
d-r---        10/3/2019   5:11 PM                Music
d-r---        10/3/2019   5:11 PM                Pictures
d-r---        10/3/2019   5:11 PM                Saved Games
d-r---        10/3/2019   5:11 PM                Searches
d-r---        10/3/2019   5:11 PM                Videos
```

Get-ChildItem | Sort-Object

Now that we understand the basics of Powershell we can get to offensively using Powershell to enumerate and exploit.

Introduction to Offensive Powershell
Well we have all this information now how can we apply it to attacking a windows network? We can utilize offensive powershell to enumerate and attack Windows and Windows Active Directory.
Basic Offensive Powershell
A majority of offensive Powershell will come from using Modules like ActiveDirectory and PowerView to enumerate and exploit however powershell also has a few cmdlets that you can use to your offensively.

Using Modules in Powershell
Powershell has the ability to import modules such as ActiveDirectory and PowerView to expand the list of cmdlets available. To import a module you can either use Import-Module or you can use dot space dot backslash (. .\Module).
Examples of importing modules
    Import-Module Module
    . .\Module.ps1
Note: . .\ will only work with powershell script files. All other modules will need to be imported with Import-Module for example ActiveDirectory can only be imported with Import-Module.

Get-ADDomain
Get-ADDomain is a commandlet that pulls a large majority of the information about the Domain you're attacking. It can list all of the Domain Controllers for a given environment, tell you the NetBIOS Domain name, the FQDN (Fully Qualified Domain name) and much more. Using the Select-Object command, we can filter out some of the unnecessary objects that may be displayed (like COntainers, Group Policy Objects, and much more)
Get-ADDomain | Select-Object NetBIOSName, DNSRoot, InfrastructureMaster

```
Select Windows PowerShell
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Import-Module ActiveDirectory
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Get-ADDomain | Select-Object NetBIOSName,DNSRoot,InfrastructureMaster

NetBIOSName DNSRoot         InfrastructureMaster
----------- -------         --------------------
THROWBACK   THROWBACK.local [redacted] THROWBACK.local


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

Get-ADDomain | Select-Object NetBIOSName, DNSRoot, InfrastrucuteMaster

Get-ADForest

Get-ADForest is another commandlet that pulls all the Domains within a Forest and lists them out to the user. This may be useful if a bidirectional trust is setup, it may allow you to gain a foothold in another domain on the LAN. Just like Get-ADDomain, there is a lot of output, so we will be using Select-Object to trim the output down.

Get-ADForest | Select-Object Domains

```
PS C:\Users[redacted]\Powersploit\PowerSploit-master\Recon> Get-ADForest | Select-Object Domains

Domains
-------
{[redacted].local, THROWBACK.local}


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

Get-ADForest | Select-Object Domains

Get-ADTrust

Get-ADTrust is the last built in Powershell commandlet that we will be discussing, after this, we will move over to Powerview. Get-ADTrust provides a ton of information about the Trusts within the AD Domain. It can tell you if it's a one way or bidirectional trust, who the source is, who the target is, and much more. One required field is -Filter, this is required in the event that you want to filter on a specific Domain/Trust, if you do not (like in most circumstances), you can simply provide a * to wildcard the results.

Get-ADTrust -Filter * | Select-Object Direction,Source,Target

```
Windows PowerShell
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Get-ADTrust -Filter * | Select-Object Direction,Source,Target

    Direction Source                Target
    --------- ------                ------
BiDirectional DC=THROWBACK,DC=local [redacted] local


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

*Get-ADTrust -Filter * | Select-Object Direction,Source,Target*

Introduction to PowerView

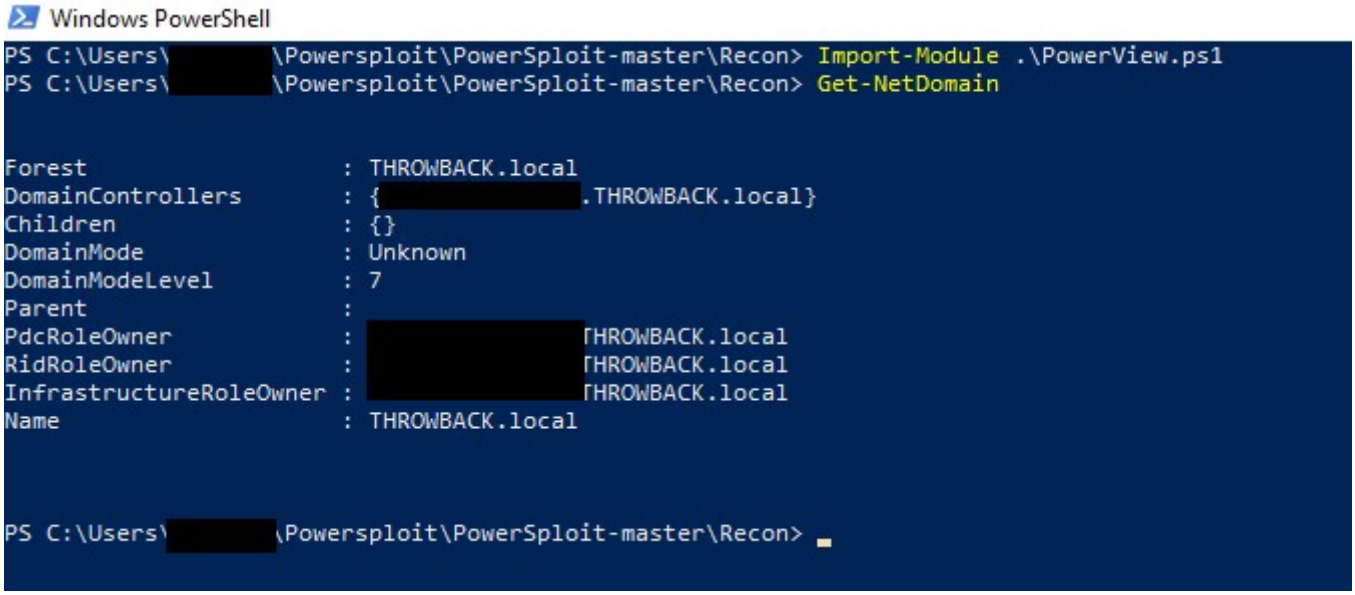
Powerview (part of PowerSploit by PowerShellMafia) is an excellent suite of tools that can be used for enumeration, and exploitation of an AD Domain, today we're only going to cover Powerview's ability to

enumerate information about the domain and their associated trusts, you can get the .ps1 here.

Get-NetDomain

Get-NetDomain is similar to the ActiveDirectory module's Get-ADDomain but contains a lot less information, which can be better. Basic info such as the Forest, Domain Controllers, and Domain Name are enumerated.

Get-NetDomain



Get-NetDomain

Get-NetDomainController

Get-NetDomainController is another useful cmdlet that will list all of the Domain Controllers within the network. This is incredibly useful for initial reconnaissance, especially if you do not have a Windows device that's joined to the domain.

Get-NetDomainController

```
Windows PowerShell
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Import-Module .\PowerView.ps1
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Get-NetDomain


Forest                  : THROWBACK.local
DomainControllers       : {[redacted].THROWBACK.local}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : [redacted]THROWBACK.local
RidRoleOwner            : [redacted]THROWBACK.local
InfrastructureRoleOwner : [redacted]THROWBACK.local
Name                    : THROWBACK.local


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

Get-NetDomainControllers

Get-NetForest

Get-NetForest is similar to Get-ADForest, and provides similar output. It provides all the associated Domains, the root domain, as well as the Domain Controllers for the root domain.

Get-NetForest

```
Windows PowerShell
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Get-NetForest


RootDomainSid         : S-1-5-21-3906589501-690843102-3982269896
Name                  : THROWBACK.local
Sites                 : {THROWBACK}
Domains               : {[redacted], THROWBACK.local}
GlobalCatalogs        :
ApplicationPartitions : {DC=DomainDnsZones,DC=THROWBACK,DC=local, DC=ForestDnsZones,DC=THROWBACK,DC=local}
ForestModeLevel       : 7
ForestMode            : Unknown
RootDomain            : THROWBACK.local
Schema                : CN=Schema,CN=Configuration,DC=THROWBACK,DC=local
SchemaRoleOwner       : [redacted].THROWBACK.local
NamingRoleOwner       : [redacted].THROWBACK.local


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

Get-NetForest

Get-NetDomainTrust

Get-NetDomainTrust is similar to Get-ADTrust with our SelectObject filter applied to it. It’s short, sweet and to the point!

Get-NetDomainTrust

```
PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon> Get-NetDomainTrust

SourceName      TargetName      TrustType TrustDirection
----------      ----------      --------- --------------
THROWBACK.local [redacted]       TreeRoot  Bidirectional


PS C:\Users\[redacted]\Powersploit\PowerSploit-master\Recon>
```

Get-NetDomainTrust

Answer the questions below

Read the above and take note of the commands for future tasks.

Question Done