Quick Reference for PowerShell... with CMD Bits

```
Who am I?
https://www.linkedin.com/in/propatriavigilans/
https://raymondrizzo.com
```

Check PS Version

```
PS C:\> Get-Host
PS C:\> $PSVersionTable
PS C:\> $PSVersionTable.PSVersion
```

PowerShell Execution Policy Bypass

`C:\> PowerShell -ep bypass`

Execution Policy

`PS C:\> Get-ExecutionPolicy`

Best Case

`PS C:\> Set-ExecutionPolicy [AllSigned | Restricted]`

OK Case

`PS C:\> Set-ExecutionPolicy Remote-Signed`

Please Don't

`PS C:\> Set-ExecutionPolicy Unrestricted`

Loading Modules

See what is loaded

`PS C:\> Get-Module`

Find available modules

`PS C:\> Get-Module -ListAvailable`

Import Module

`PS C:\> Import-Module [module-name]`

Finding Commandlets (Eample: Relating to Services)

`PS C:\> Get-Command -Noun service`

Find Commandlets in Module

`PS C:\> Get-Command -Module [module-name]`

Find Object Properties

`PS C:\> Get-ChildItem | Get-Member`

Common Pipelines

```
PS C:\> Get-ChildItem | Sort-Object -Decending -Property LastWriteTime
PS C:\> Get-Process -Name 'Notepad' | Stop-Process
PS C:\> Get-NetFirewallRule | Export-Csv firewall-rules.csv
PS C:\> Get-EventLog -List | Sort-Object -Descending
PS C:\> Get-EventLog -LogName Dell | Where-Object -Property Message -Like
"*integrity*"
PS C:\> Get-NetAdapter | Where-Object {$_.Name -Like "Ethernet*"} | Disable-
NetAdapter -whatif
PS C:\> Get-ADUser -Filter * -Properties LastLogonDate | Where-Object
{$_.LastLogonDate -le (Get-Date).AddDays(-60)} | Sort-Object -Descending -Property
LastLogonDate
```

Navigate to More Places

```
PS C:\> cd Env
PS C:\> cd HKLM:\
PS C:\> Get-PSDrive
```

Perform Actions on Each Result

`PS C:\> Get-ChildItem | foreach {"$($_.GetType().FullName) - $_.name"}`

Enable PowerShell Remoting

Enable PowersShell HTTP

`PS C:\> Enable-PSRemoting`

Enable PowersShell HTTPS

FQDN X.509v3 cert in Personal Store

Find Thumbprint of Cert

`PS Cert:\> Get-ChildItem` *Use WinRM to enable Use -UseSSL option to connect over SSL Use -SkipCACheck if using self signed*

Remote Commands (Stateless)

Get Remote Variable

`PS C:\> Invoke-Command -ComputerName [computer-name] -ScriptBlock { Get-ScheduledTask | Get-Member }`

Get Remote Data

`PS C:\> Invoke-Command -ComputerName [computer-name] -ScriptBlock { Get-ScheduledTask | Get-Member }`

Always filter data to leftmost commandlet to limit serilization/deserialization and transfer time.

Remote Commands (Session Based)

```
PS C:\> $TheSession = New-PSSession -ComputerName -[computer-name]
PS C:\> Get-PSSession
PS C:\> Invoke-Command -Session ##TheSession -ScriptBlock { Get-ScheduledTask |
Get-Member }
PS C:\> $TheSession | Remove-PSSession
```

Get System Information

`PS C:\> Get-ComputerInfo`

Ping Sweep

`PS C:\> 1..1255 | % {"200.0.0.$($_): $(Test-Connection -count 1 -comp 200.0.0.$($_) -quiet)"}`

Find listening ports that match file ports

listening-ports.ps1

```
$system_ports = Get-NetTCPConnection -State Listen
$text_port = Get-Content -Path C:\Users\Administrator\Desktop\ports.txt
foreach($port in $text_port){
    if($port -in $system_ports.LocalPort){
        echo $port
        }
}
```

Find files that contain passwords

files-passwords.ps1

```
$current_directory = Get-Location
Get-ChildItem -File -Recurse -Path $current_directory | Select-String -pattern
assword
```

Loop directories at current level

```
$found_directories = Get-ChildItem -Directory
foreach($directory in $found_directories){
  echo $directory.Name
}
```

Check ports without netcat

check-ports.ps1

```
$port_range = (130..140)
foreach($port in $port_range){
  $test_result = Test-NetConnection 127.0.0.1 -Port $port
  if($test_result.TcpTestSucceeded = 'True'){
    echo $test_result.RemotePort
  }
}
```

Add user to DC admins

```
PS C:\> $s = New-PSSession -ComputerName [Domain-Controller-Name]
PS C:\> Invoke-Command -Session $s -ScriptBlock {Add-ADGroupMember -Identity
"Administrators" -Members [user-name]}+
```

Enumerate shares

`PS C:\> get-WmiObject -class Win32_Share`

Get PS Commandlet Object Properties and Methods

`PS C:\> Get-ScheduledTask | Get-Member`

Show all output for commandlet

`PS C:\> Get-ScheduledTask | Format-List`

Create a zip file with the contents of C:\Stuff\

`PS C:\> Compress-Archive -Path C:\Stuff -DestinationPath archive.zip`

Add more files to the zip file

Existing files in the zip file with the same name are replaced

`PS C:\> Compress-Archive -Path C:\OtherStuff\*.txt -Update -DestinationPath archive.zip`

Extract the zip file to C:\Destination\

`Expand-Archive -Path archive.zip -DestinationPath C:\Destination`

Get Network Information

```
C:\> netstat -ano
PS C:\> Get-NetTCPConnection
PS C:\> Get-NetTCPConnection | Where-Object -Property State -Match Listen |
Measure-Object
C:\> arp -a
PS C:\> Get-NetNeighbor
C:\> ipconfig /all
PS C:\> Get-NetIPAddress
C:\> route print
PS C:\> Get-NetRoute
```

Get Users

```
C:\> net user
C:\> net user /domain
C:\> net group /domain
C:\> net group "Domain Admins" /domain
C:\> net group "Enterprise Admins" /domain
PS C:\> Get-ADUser -Filter * -SearchBase "CN=Users,DC=THMREDTEAM,DC=COM"
PS C:\> Get-ADUser -Filter * -SearchBase "OU=THISOU,DC=THMREDTEAM,DC=COM"
PS C:\> Get-IAMUser
PS C:\> Get-LocalUser
PS C:\> Get-LocalUser | Where-Object -Property PasswordRequired -Match false
```

Check User Groups

`C:\> net user [user-name] /domain`

Check Registry

Find Specific Entry

```
PS C:\> Get-ChildItem -Path
'HKLM:\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate\'
PS C:\> Get-ChildItem -Path
'HKCU:\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\EXPLORER\DISCARDABLE\POSTSETUP\
COMPONENT CATEGORIES\{F3F18253-2050-E690-FED7-0BE7DF1E790D}\ENUM'
```

Snapshot and Compare Registry

```
PS C:\> $regSnapshot = Get-ChildItem -Recurse -ErrorAction Ignore | % name
PS C:\> $Current = Get-ChildItem -Recurse -ErrorAction Ignore | % name
PS C:\> Compare-Object $Snapshot $Current
```

Check Malware Protection

```
C:\> wmic /namespace:\\root\securitycenter2 path antivirusproduct
PS C:\> Get-CimInstance -Namespace root/SecurityCenter2 -ClassName
AntivirusProduct
PS C:\> Get-Service WinDefend
PS C:\> Get-MpThreat
PS C:\> Get-MpComputerStatus | select RealTimeProtectionEnabled
```

Check Firewall Protection

```
PS C:\> Get-NetFirewallProfile | Format-Table Name, Enabled
PS C:\> Set-NetFirewallProfile -Profile Domain, Public, Private -Enabled False
PS C:\> Get-NetFirewallProfile | Format-Table Name, Enabled
PS C:\> Get-NetFirewallRule | select DisplayName, Enabled, Description
```

Check Opn Ports

`PS C:\> Test-NetConnection -ComputerName 127.0.0.1 -Port 80`

Check Event Logs

```
C:\> eventvwr.msc (Event Viewer GUI-based application)
C:\> Wevtutil.exe (command-line tool)
    - https://docs.microsoft.com/en-us/windows-server/administration/windows-
commands/wevtutil
    C:\>  wevtutil qe Application /c:3 /rd:true /f:text
PS C:\> Get-EventLog -List
PS C:\> Get-WinEvent$regSnapshot = Get-ChildItem -Recurse -ErrorAction Ignore | %
name
    - https://docs.microsoft.com/en-
```

```
us/PowerShell/module/microsoft.PowerShell.diagnostics/get-winevent?
view=PowerShell-5.1
    PS C:\> Get-WinEvent -LogName Application | Where-Object { $_.ProviderName -
Match 'WLMS' }
    - https://docs.microsoft.com/en-us/PowerShell/scripting/samples/Creating-Get-
WinEvent-queries-with-FilterHashtable?view=PowerShell-7.1
    - https://docs.microsoft.com/en-
us/PowerShell/module/microsoft.PowerShell.core/about/about_hash_tables?
view=PowerShell-7.1
        PS C:\> Get-WinEvent -FilterHashable @{ Logname='Application'
ProviderName='MsiInstaller' ID=11707 }
        PS C:\> Get-WinEvent -FilterHashtable @{LogName='Microsoft-Windows-
PowerShell/Operational'; ID=4104} | Select-Object -Property Message | Select-
String -Pattern 'SecureString'
    - https://docs.microsoft.com/en-us/windows/win32/wes/consuming-events##xpath-
10-limitations
    - https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-
4.0/ms256115(v=vs.100)
        PS C:\> Get-WinEvent -LogName Application -FilterXPath
'*/System/EventID=100'
        PS C:\> Get-WinEvent -LogName Application -FilterXPath
'*/System/Provider[@Name="WLMS"]'
        PS C:\> Get-WinEvent -LogName Application -FilterXPath
'*/System/EventID=101 and */System/Provider[@Name="WLMS"]'
        PS C:\> Get-WinEvent -LogName Security -FilterXPath
'*/EventData/Data[@Name="TargetUserName"]="System"'
        PS C:\> Get-WinEvent -LogName Application -FilterXPath
'*/System/Provider[@Name="WLMS"] and */System/TimeCreated[@SystemTime=”2020-12-
15T01:09:08.940277500Z”]'
        PS C:\> Get-WinEvent -LogName Security -FilterXPath
'*/EventData/Data[@Name="TargetUserName"]="Sam" and */System/EventID=4720'
- Windows Logging References
    -
https://static1.squarespace.com/static/552092d5e4b0661088167e5c/t/580595db9f745688
bc7477f6/1476761074992/Windows+Logging+Cheat+Sheet_ver_Oct_2016.pdf
    - https://apps.nsa.gov/iaarchive/library/reports/spotting-the-adversary-with-
windows-event-log-monitoring.cfm
    - https://docs.microsoft.com/en-us/windows-server/identity/ad-
ds/plan/appendix-l####events-to-monitor
    - https://www.microsoft.com/en-us/download/confirmation.aspx?id=52630
- Note: Some events will not be generated by default, and certain features will
need to be enabled/configured on the endpoint, such as PowerShell logging. This
feature can be enabled via Group Policy or the Registry:
    - Local Computer Policy > Computer Configuration > Administrative Templates >
Windows Components > Windows PowerShell
- Another feature to enable/configure is Audit Process Creation, which will
generate event ID 4688. This will enable command-line process auditing.
    - Local Computer Policy > Computer Configuration > Administrative Templates >
System > Audit Process Creation
- https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/manage/component-
updates/command-line-process-auditing##try-this-explore-command-line-process-
auditing
```

Check for Sysmon

```
PS C:\> Get-Process | Where-Object { $_.ProcessName -eq "Sysmon" }
PS C:\> CGet-CimInstance win32_service -Filter "Description = 'System Monitor service'"
PS C:\> Get-Service | Where-Object {$_.DisplayName -like "*sysm*"}
C:\> reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\Microsoft-Windows-Sysmon/Operational
C:\> findstr /si '<ProcessCreate onmatch="exclude">' C:\tools\*
```

Check for EDRs

```
https://github.com/PwnDexter/Invoke-EDRChecker
https://github.com/PwnDexter/SharpEDRChecker
```

Check Installed Programs

`C:\> wmic product get name,version`

Check Installed Patches

`PowerShell C:\> get-wmiobject -class win32_quickfixengineering`

Check all services

```
PS C:\> Get-Service | Where-Object -Property Status -eq Running
PS C:\> Get-Service | Where-Object{$_.status -eq "running"}
```

Check running services

Find the Process ID of a service

`PS C:\> Get-Process -Name thm-demo`

See if the process is listening on any ports

`C:\> netstat -noa |findstr "LISTENING" | findstr "3212"`

Check running processes

`PS C:\> Get-Process`

Check for Hidden Items

`PS C:\> Get-ChildItem -Hidden -Path C:\Users\kkidd\Desktop\`

Find Files

```
PS C:\> Get-ChildItem -File -Include *.txt -Path c:\ -ErrorAction SilentlyContinue
-Recurse | Where-Object -Property Name -EQ interesting-file.txt
PS C:\> Get-ChildItem -File -Path c:\ -ErrorAction SilentlyContinue -Recurse |
Where-Object -Property Name -Like *file*
PS C:\> Get-ChildItem -Path C:\ -Include *.txt -File -Recurse -ErrorAction
SilentlyContinue | Where-Object -Property Name -EQ interesting-file.txt
PS C:\> $FindDate=Get-Date -Year 2016 -Month 06 -Day 24
PS C:\> Get-ChildItem -Path C:\ -File -Recurse -ErrorAction SilentlyContinue |
Where-Object { $_.LastWriteTime -ge $FindDate }
PS C:\> Get-ChildItem . | Select-Object -last 1
PS C:\> Get-Command | Where-Object -Property CommandType -EQ Cmdlet | Measure-
Object
PS C:\> Test-Path C:\Users\Administrator\Documents\Passwords
```

Search in all files

`PS C:\> Get-ChildItem -File -Recurse -Path C:\ -Exclude *.dll | Select-String -pattern API_KEY`

Perform DNS zone transfer using the Microsoft tool is nslookup.exe

`C:\> nslookup.exe`

Once we execute it, we provide the DNS server that we need to ask, which in this case is the target machine

`> server 10.10.235.137`

Now let's try the DNS zone transfer on the domain we find in the AD environment.

`> ls -d thmredteam.com`

Check scheduled tasks

```
PS C:\> Get-ScheduledTask
PS C:\> Get-ScheduledTask | Where-Object -Property TaskName -Like *new*
```

Note: *Check scheduled task logs in C:\ Program Files (x86)\SystemScheduler\Events*

Find file owner

`PS C:\> Get-ACL C:\ | Format-List`

Hashing and Encryption

Base64 Decode

```
PS C:\> $MYTEXT = 'VABoAGkAcwAgAGkAcwAgAG0AeQAgAHMAZQBjAHIAZQB0ACAAdABlAHgAdAA='
PS C:\> $DECODED =
[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($MYTE
XT))
PS C:\> Write-Output $DECODED
```

Navigate Certs

```
PS C:\> cd cert
PS Cert:\>
```

Hash File

```
PS C:\> Get-FileHash $pshome\powershell.exe | Format-List
 Algorithm : SHA256
 Hash      : 6A785ADC0263238DAB3EB37F4C185C8FBA7FEB5D425D034CA9864F1BE1C1B473
 Path      : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

MD5 Check File

```
PS C:\> Get-FileHash $pshome\powershell.exe -Algorithm MD5 | Format-List
 Algorithm : SHA256
 Hash      : 6A785ADC0263238DAB3EB37F4C185C8FBA7FEB5D425D034CA9864F1BE1C1B473
 Path      : C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Signing Scripts

Check Code Signing Certificate and note location in store

`PS C:\> Get-ChildItem cert:\CurrentUser\My -codesign`

Set Variable to Cert

`PS C:\> $MyCert=(cert:currentuser\my\ -Ccodesign[x])`

Sign PowerShell Script

`PS C:\> Set-AuthenticodeSignature c:\[Script-Name].ps1 $MyCert`

Server Core

Join Domain and rename server

```
PS C:\> Install-WindowsFeature AD-Domain-Services -IncludeManagementTools
```

Add GUI to Windows Server Core

```
PS C:\> Install-WinndowsFeature Server-Gui-Mgmt-Infra, Server-Gui-Shell,
PowerShell-ISE -Source:wim:d:\sources\install.wim:2
or
PS C:\> Add-WindowsFeature -Name Server-Gui-Mgmt-Infra, Server-Gui-Shell,
PowerShell-ISE -computer [ComputerName]
```

Hyper-V

Create VLANS

```
    PS C:\> New-VLAN -Name "VLan200" -Number 200 -Switch "Hyper-V Switch"
    PS C:\> New-VLAN -Name "VLan201" -Number 201 -Switch "Hyper-V Switch"
```

Create VM NICs

Create the VLM200 ports (4 like it's physical counterpart)

```
For ($Count=0; $Count -le 3; $Count ++)
{
Add-VMNetworkadapter -VMName VLM200-1 -Name "Eth$Count"
}
```

Rename Virtual NICs

```
For ($Count=0; $Count -le 3; $Count ++)
{
    $VMName = Get-VM -Name VLM200-1
    $VMName.NetworkAdapters[$Count].Name = "Eth$Count"
}
or
PS C:\> $VMNetAdap = Get-VMNetworkAdapter -VMName '1234 Market MG-SIP'
PS C:\> $VMNetAdap[0]
PS C:\> rename-VMNetworkAdapter -VMNetworkAdapter $VMNetAdap[0] -newname 'NEC
Vlan'
PS C:\> rename-VMNetworkAdapter -VMNetworkAdapter $VMNetAdap[1] -newname 'VZN
Vlan'
```

Connect v-switch

```
PS C:\> Connect-VMNetworkAdapter -VMName 'VyOS Router' -VMNetworkAdapterName eth1 -SwitchName 'Wireless Virtual Switch'
```

Set access or trunk ports

```
PS C:\> Set-VMNetworkAdapterVlan -VMName SV9500-HyperV -VMNetworkAdapterName 'LAN 1' -Trunk -AllowedVlanIdList '2, 4, 40' -NativeVlanId 20
PS C:\> Set-VMNetworkAdapterVlan -VMName SV9500-HyperV -VMNetworkAdapterName 'LAN 2' -Trunk -AllowedVlanIdList '2, 4, 40' -NativeVlanId 20
PS C:\> Set-VMNetworkAdapterVlan -VMName SV9500-HyperV -VMNetworkAdapterName 'LAN 1' -Access -VlanId 20
PS C:\> Set-VMNetworkAdapterVlan -VMName SV9500-HyperV -VMNetworkAdapterName 'LAN 2' -Access -VlanId 20
```

Install Modules with PowerShellGallery

```
PS C:\> Install-Module -Name PowerShellGet -Force -AllowClobber
PS C:\> Install-Module -Name MicrosoftTeams -Force -AllowClobber
PS C:\> Update-Module -Name MicrosoftTeams
```