GOLANG

FOR

PENTESTERS / REDTEAMERS

BY

ANANT SHRIVASTAVA

DIRECTOR NOTSOSECURE GLOBAL SERVICES

ANANT SHRIVASTAVA

- Director NotSoSecure Global Services
- Sysadmin / Developer / Security
- Trainer / Speaker: BlackHat, Nullcon, RootConf, C0c0n, RuxCon, IpExpo
- Project Owner: AndroidTamer, CodeVigilant
- Contributor: null, G4H, OWASP and more
- @anantshri on all major social media platform
- https://anantshri.info

AGENDA

1. What is golang
2. Why Golang
3. Setup required for Golang
4. Hello World in Golang
5. Useful Modules in Golang
6. Various Usecases for Pentesters / RedTeamers
7. Advanced Usage and Optimization Tips

DISCLAIMER

- This talk is focused on Golang or Go Programming Language
- We will not be doing performance comparision or debate about other languages
- We will not be deep diving rather exploring golang to give people a headstart

GO LANGUAGE

- Go programming language or golang
- Design started in 2007
- Public release in 2009
- Widely used within google since start
- Working with Go is intended to be fast

WHY GOLANG

- Go combines
  - the ease of programming of an interpreted, dynamically typed language
  - efficiency and safety of a statically typed, compiled language
  - It also aimed to be modern, with support for networked and multicore computing
- Cross Platform (Windows/Linux/Mac)
- Fat Static Binary by default
- no 3rd party dependency not even runtime
- Can run in bare docker's or directly above kernel

WHY NOT GOLANG

- Very Young ecosystem
- dependent on github
- Large Binaries
- Not so great GUI Libraries
- monolithic binaries

More Funny why not golang ideas look at
https://speakerdeck.com/majek04/golang-sucks?slide=13

TOOLS ALREADY WRITTEN

- Gobuster (https://github.com/OJ/gobuster)
- Bettercap 2.0 (https://github.com/bettercap/bettercap)
- GoBot2 (https://github.com/SaturnsVoid/GoBot2)
- GoAT (https://github.com/petercunha/GoAT)
- Cracklord (https://github.com/jmmcatee/cracklord)
- GoCrack (https://github.com/fireeye/gocrack)
- Merlin (https://github.com/Ne0nd0g/merlin)
- Vuls (https://github.com/future-architect/vuls)
- … many more (https://github.com/topics/pentesting?l=go)

SETUP REQUIRED FOR GOLANG

- Download Binaries: https://golang.org/dl/
- Linux: If you want to stick with Distro
  - Install via apt / yum / your package manager
- Note: Add local go path to your PATH

```
echo 'export PATH=$PATH:~/go/bin' >> ~/.bashrc
```

- GOPATH: Where packages etc will be installed (<>user_home>/go)
- GOROOT: Default installation location (usr/lib)

HELLO WORLD

- HelloWorld.go

```
package main

import "fmt"

func main() {
    fmt.Printf("hello, world\n")
}
```

HOW TO HELLO WORLD

1 Direct Execution

```
$ go run HelloWorld.go
hello, world
```

2 Build and Run

```
$ go build HelloWorld.go

$ ls
HelloWorld  HelloWorld.go

$ $ ./HelloWorld
hello, world
```

EXECUTING DIFFERENT VERSION

- Download Other Versions (Takes some time)

```
$ go get golang.org/dl/go1.13.1
$ go1.13.1 download
```

- Execute other version

```
$ go1.13.1 version
```

MODULES IN GOLANG

- Now a days we don't write programs
- We collect modules and stich them together
- Language is as good as the modules it provides
- We will explore couple of interesting golang modules
- This is by no means exhaustive list

ARGPARSE

- Useful for taking commandline inputs
- Improvement over `flag` provided as default choice

Link: https://github.com/akamensky/argparse

```
$ go get -u -v github.com/akamensky/argparse
```

ARGPARSE SAMPLE CODE

```
package main
import (
    "fmt"
    "github.com/akamensky/argparse"
    "os"
)

func main() {
    // Create new parser object
    parser := argparse.NewParser("print", "Prints provided string to stdout")
    // Create string flag
    s := parser.String("s", "string", &argparse.Options{Required: true, Help: "Print
String"})
    // Parse input
    err := parser.Parse(os.Args)
    if err != nil {
        // In case of error print error and print usage
        // This can also be done by passing -h or --help flags
        fmt.Print(parser.Usage(err))
    }
    // Finally print the collected string
    fmt.Println(*s)
}
```

NET

- https://golang.org/pkg/net/

```
import net
```

Function	Usage
`net.LookupIP("domainname")`	Domain to IP Address
`net.LookupCNAME("domainname")`	Domain to CNAME
`net.LookupAddr("IP_Address")`	IP to Domain
`net.LookupNS("domainname")`	Name Server of a Domain
`net.LookupMX("domainname")`	MX Record
`net.LookupTXT("domainname")`	TXT Record

TIP: If a domain is registered it will always have a NS Record

PUBLICSUFFIX

- There is no clear way of identifying domain name from a string
- https://godoc.org/golang.org/x/net/publicsuffix
- Uses https://publicsuffix.org data to make an accurate guess

```
import "golang.org/x/net/publicsuffix"
```

```
publicsuffix.EffectiveTLDPlusOne(domainname)
```

- blog.abc.com -> google.com
- test.abc.co.in -> abc.co.in

JSON ENCODING

- Encoding output in JSON Format
- https://golang.org/pkg/encoding/json/

```
import "encoding/json"
```

- Easily convert Arrays into json objects

EXAMPLE

```
package main
import (
    "encoding/json"
    "fmt"
    "os"
)
func main() {
    type ColorGroup struct {
        ID     int
        Name   string
        Colors []string
    }
    group := ColorGroup{
        ID:     1,
        Name:   "Reds",
        Colors: []string{"Crimson", "Red", "Ruby", "Maroon"},
    }
    b, err := json.Marshal(group)
    if err != nil {
        fmt.Println("error:", err)
    }
    os.Stdout.Write(b)
}
```

Output

```
{"ID":1,"Name":"Reds","Colors":["Crimson","Red","Ruby","Maroon"]}
```

PORT SCANNING

```
import net
```

```
net.Dial("tcp", "domainname:portnumber")
```

or

```
net.DialTimeout("tcp", "domainname:portnumber",Timeout)
```

- DailTimeout has a timesout after Timeout seconds
- Alternative: https://github.com/anvie/port-scanner

WEB REQUESTS

- https://github.com/anaskhan96/soup
- Inspired by BeautifulSoup of Python

```
func Get(string) (string,error){} // Takes the url as an argument, returns HTML string
func Header(string, string){} // Takes key,value pair to set as headers for the HTTP request
made in Get()
func Cookie(string, string){} // Takes key, value pair to set as cookies to be sent with the
HTTP request in Get()
func HTMLParse(string) Root {} // Takes the HTML string as an argument, returns a pointer to
the DOM constructed
func Find([]string) Root {} // Element tag,(attribute key-value pair) as argument, pointer to
first occurence returned
func FindAll([]string) []Root {} // Same as Find(), but pointers to all occurrences returned
func FindNextSibling() Root {} // Pointer to the next sibling of the Element in the DOM
returned
func FindNextElementSibling() Root {} // Pointer to the next element sibling of the Element
in the DOM returned
func FindPrevSibling() Root {} // Pointer to the previous sibling of the Element in the DOM
returned
func FindPrevElementSibling() Root {} // Pointer to the previous element sibling of the
Element in the DOM returned
func Children() []Root {} // Find all direct children of this DOM element
func Text() string {} // Full text inside a non-nested tag returned, first half returned in a
non-nested one
func FullText() string {} // Full text inside a nested/non-nested tag returned
```

EXAMPLE SOUP

```
package main


import (
    "fmt"
    "github.com/anaskhan96/soup"
    "os"
)
func main() {
    resp, err := soup.Get("https://xkcd.com")
    if err != nil {
        os.Exit(1)
    }
    doc := soup.HTMLParse(resp)
    links := doc.Find("div", "id", "comicLinks").FindAll("a")
    for _, link := range links {
        fmt.Println(link.Text(), "| Link :", link.Attrs()["href"])
    }
}
```

* Alterantive: Colly

CODE FORMATTING

- Golang follows a uniform code formatting syntax
- Instead of leaving formatting to users
- gofmt command does the code formatting as per golang standard
- Ref: https://golang.org/cmd/gofmt/
- Ref: https://golang.org/doc/effective_go.html#formatting

SIZE BINARY REDUCTION

- fat binaries are basically FAT

```
-rwxrwxr-x 1 anant anant 2.0M Sep 28 07:14 HelloWorld
```

- this extra size is because of the runtime intigration in code
- Some fat can be trimmed
- compilier takes following commandline arguments
  - -s: Omit the symbol table and debug information.
  - -w: Omit the DWARF symbol table.

```
$ go build -ldflags="-s -w" -o slimhello HelloWorld.go
$ ls -lh slimhello
-rwxrwxr-x 1 anant anant 1.2M Sep 28 09:41 slimhello
```

- Ref: https://blog.filippo.io/shrink-your-go-binaries-with-this-one-weird-trick/

DOCKER DEPLOYMENT

- As fat binaries have no dependencies they can run in Scratch
- DockerFile

```
FROM scratch
COPY slimhello /hello
ENTRYPOINT ["/hello"]
```

- Autobuild

```
@echo -e "FROM scratch\nCOPY slimhello /hello\nENTRYPOINT ["/hello"]" > Dockerfile
docker build -t "${PROJECT}" .
```

X COMPILING

- `env GOOS=linux GOARCH=amd64 go build HelloWorld.go`

$GOOS	$GOARCH	$GOOS	$GOARCH	$GOOS	$GOARCH	$GOOS	$GOARCH
aix	ppc64	android	386	linux	arm64	linux	ppc64
android	amd64	android	arm	linux	ppc64le	linux	mips
android	arm64	darwin	386	linux	mipsle	linux	mips64
darwin	amd64	darwin	arm	darwin	arm64	dragonfly	amd64
freebsd	386	freebsd	amd64	freebsd	arm	illumos	amd64
js	wasm	linux	386	linux	amd64	linux	arm
linux	mips64le	linux	s390x	netbsd	386	netbsd	amd64
netbsd	arm	openbsd	386	openbsd	amd64	openbsd	arm
openbsd	arm64	solaris	amd64	windows	386	windows	amd64

SECURE YOUR CODE

- You can run periodic checks on your own code
- gosec is one good code review tool for golang
- https://github.com/securego/gosec
- Multiple Rules available : https://github.com/securego/gosec#available-rules

```
# Run recursively in current Directory and below
$ gosec ./...

# Run a specific set of rules
$ gosec -include=G101,G203,G401 ./...

# Run everything except for rule G303
$ gosec -exclude=G303 ./...
```

- Snyk has recently introduced Go module dependency Checks

UPDATE MANAGEMENT

- Simmilar to other software systems dev need to keep a check on dependences
- Suggested path is to build after updating the depdency to ensure you are building against latest
- you may need a lot more then just simple testing.
- how to update
  - `go get -u <URL>`

AUTOMATE IT

```
PROJECT=project_name

linux:
    env GOOS=linux GOARCH=amd64 go build -ldflags="-s -w" -o bin/${PROJECT}-lin-amd64 ./

windows:
    env GOOS=windows GOARCH=amd64 go build -ldflags="-s -w" -o bin/${PROJECT}-win-amd64 ./

mac:
    env GOOS=darwin GOARCH=amd64 go build -ldflags="-s -w" -o bin/${PROJECT}-mac-amd64 ./

docker: linux
    @echo "FROM scratch\nCOPY bin/${PROJECT}-lin-amd64 /${PROJECT}\nENTRYPOINT
[\"/${PROJECT}\"]" > Dockerfile
    docker build -t "${PROJECT}" .

clean:
    rm -rf bin/${PROJECT}*

check: updatedep
    gosec ./

updatedep:
    go get -u "github.com/akamensky/argparse"
    go get -u "golang.org/x/net/publicsuffix"

all: linux mac windows

complete: all docker
```

REVERSING GO APPLICATIONS

- being FAT binaries go apps are hard to reverse engineer
- Suggested reading in case you want to explore that
- Ref: https://rednaga.io/2016/09/21/reversing_go_binaries_like_a_pro/
- Ref: https://medium.com/@nishanmaharjan17/reversing-golang-binaries-part-1-c273b2ca5333
- Ref: https://medium.com/@nishanmaharjan17/reversing-golang-binaries-part-2-26f522264d01

GOLANG REPRODUCABLE BUILD

- Reproducable build's ensure the source code can produce byte by byte matching binaries at different systems
- The trick is maintain a consistent build environment
  - Provide it: Docker build
  - Maintain: Ensuring environment is replicated.
- Ref: https://blog.filippo.io/reproducing-go-binaries-byte-by-byte/

ANY QUESTIONS

https://github.com/Go4Security

- Twitter: @anantshri
- Linkedin: linkedin.com/in/anantshri