crypto
docs
logger
netio
operate
option
socks5
.editorconfig
.gitignore
.goreleaser.yml
CHANGELOG
LICENSE
README.md
go.mod
go.sum
main.go

About

Tool for port forwarding & intranet proxy

proxy intranet pentest golang-tools traffic-forwarding udp-forwarding

Readme
MIT license
Activity
1k stars
23 watching
186 forks
Report repository

Releases 5

v0.4 release Latest
on Sep 22, 2020

+ 4 releases

Packages

No packages published

Languages

● Go 61.0% ● Assembly 39.0%

README MIT license

iox

English | 中文

Tool for port forward & intranet proxy, just like `lcx` / `ew`, but better

Why write?

`lcx` and `ew` are awesome, but can be improved.

When I first used them, I can't remember these complicated parameters for a long time, such as `tran, slave, rcsocks, sssocks...`. The work mode is clear, why do they design parameters like this(especially `ew`'s `-l -d -e -f -g -h`)

Besides, I think the net programming logic could be optimized.

For example, while running `lcx -listen 8888 9999` command, client must connect to `:8888` first, then `:9999`, in `iox`, there's no limit to the order in two ports. And while running `lcx -slave 1.1.1.1 8888 1.1.1.1 9999` command, `lcx` will connect two hosts serially, but it's more efficient to connect in concurrent, as `iox` does.

What's more, `iox` provides traffic encryption feature (it's useful when there is a IDS on target). Actually, you can use `iox` as a simple ShadowSocks.

And `iox` also provides UDP traffic forward.

Of course, because `iox` is written in Go, the static-link-program is a little large, raw program is 2.2MB (800KB after UPX compression)

Features

- Traffic encryption (optional)
- Humanized CLI option
- Logic optimization
- UDP traffic forward
- TCP multiplexing in reverse proxy mode

Usage

You can see, all params are uniform. `-l/--local` means listen on a local port; `-r/--remote` means connect to remote host

Note: after v0.4, `-l/--local` could specify which IP to listen on. If only ports are specified, the default is `0.0.0.0:PORT`

```
-l 127.0.0.1:9999      -l *127.0.0.1:9999      # 127.0.0.1:9999
-l 9999                -l *9999                # 0.0.0.0:9999

`-l :9999` is also OK, but it's not recommended. Because `-l *:9999`
```

Working mode

fwd

Listen on `0.0.0.0:8888` and `0.0.0.0:9999`, forward traffic between 2 connections

```
./iox fwd -l 8888 -l 9999
```

Listen on `0.0.0.0:8888`, forward traffic to `1.1.1.1:9999`

```
./iox fwd -l 8888 -r 1.1.1.1:9999
```

Connect `1.1.1.1:8888` and `1.1.1.1:9999`, forward between 2 connection

```
./iox fwd -r 1.1.1.1:8888 -r 1.1.1.1:9999
```

proxy

Start Socks5 server on `0.0.0.0:1080`

```
./iox proxy -l 1080
```

Start Socks5 server on be-controlled host, then forward to internet VPS

VPS forward `0.0.0.0:9999` to `0.0.0.0:1080`

You must use in a pair, because it contains a simple protocol to control connecting back

```
./iox proxy -r 1.1.1.1:9999
./iox proxy -l 9999 -l 1080       // notice, the two port are in ord


for ew:
./ew -s rcsocks -l 1080 -e 9999
./ew -s rssocks -d 1.1.1.1 -e 9999
```

Then connect intranet host

```
# proxychains.conf
# socks5://1.1.1.1:1080

$ proxychains rdesktop 192.168.0.100:3389
```

Enable encryption

For example, we forward 3389 port in the intranet to our VPS

```
// be-controller host
./iox fwd -r 192.168.0.100:3389 -r *1.1.1.1:8888 -k 656565


// our VPS
./iox fwd -l *8888 -l 33890 -k 656565
```

It's easy to understand: traffic between be-controlled host and our VPS:8888 will be encrypted, the pre-shared secret key is 'AAA', `iox` will use it to generate seed key and nonce **(Normally, nonce shouldn't be reused. But consider that iox's encryption is only for bypassing IDS, in order not to allocate extra space, the TCP stream encryption will reuse the nonce)**, then encrypt with Xchacha20 (replace AES-CTR with Xchacha20 in v0.3 version)

So, the `*` should be used in pairs

```
./iox fwd -l 1000 -r *127.0.0.1:1001 -k 000102
./iox fwd -l *1001 -r *127.0.0.1:1002 -k 000102
./iox fwd -l *1002 -r *127.0.0.1:1003 -k 000102
./iox proxy -l *1003 -k 000102


$ curl google.com -x socks5://127.0.0.1:1000
```

Using `iox` as a simple ShadowSocks

```
// ssserver
./iox proxy -l *9999 -k 000102


// sslocal
./iox fwd -l 1080 -r *VPS:9999 -k 000102
```

UDP forward

Only need to add CLI option `-u`

```
./iox fwd -l 53 -r *127.0.0.1:8888 -k 000102 -u
./iox fwd -l *8888 -l *9999 -k 000102 -u
./iox fwd -r *127.0.0.1:9999 -r 8.8.8.8:53 -k 000102 -u
```

NOTICE: When you make a multistage connection, the `Remote2Remote-UDP-mode` must be started last, which is the No.3 command in above example

UDP forwarding may have behavior that is not as you expected. Actually, on GitHub now, there are only examples of forwarding a local listener to a remote host, so I can only implement them with my understanding

You can find why in the source code. If you have any ideas, PR / issue are welcomed

License

The MIT license