Sign in

jpillora / chisel Public

Notifications Fork 1.4k Star 13.2k

Code Issues 183 Pull requests 49 Actions Projects Wiki Security Insi

master

Go to file Code

- .github
- client
- example
- server
- share
- test
- .gitignore
- LICENSE
- Makefile
- README.md
- go.mod
- go.sum
- main.go

README MIT license

About

A fast TCP/UDP tunnel over HTTP

tunnel golang http tcp

Readme

MIT license

Activity

13.2k stars

205 watching

1.4k forks

Report repository

Releases 32

v1.10.1 Latest

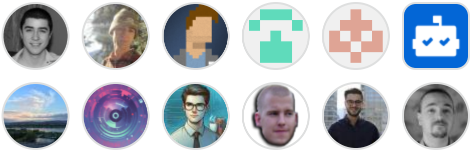
27 days ago

+ 31 releases

Packages

No packages published

Contributors 38



Chisel

GO reference CI passing

Chisel is a fast TCP/UDP tunnel, transported over HTTP, secured via SSH. Single executable including both client and server. Written in Go (golang). Chisel is mainly useful for passing through firewalls, though it can also be used to provide a secure endpoint into your network.

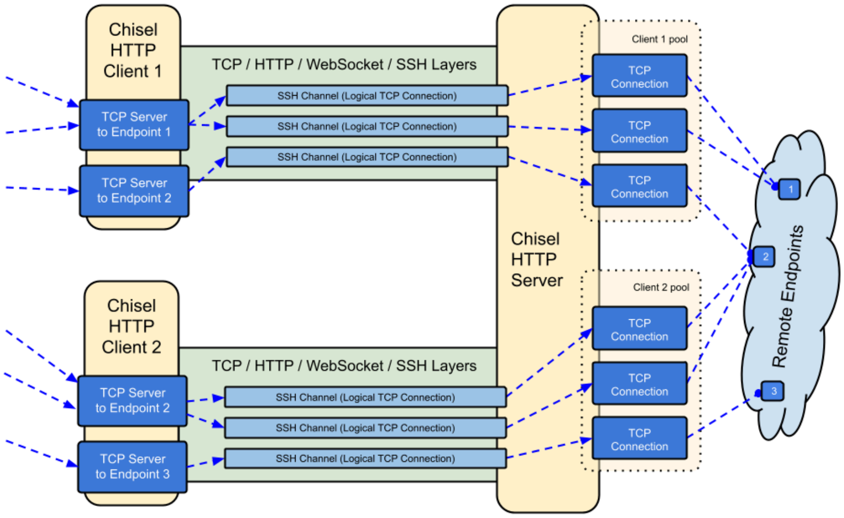

Chisel HTTP Client 1
TCP / HTTP / WebSocket / SSH Layers
SSH Channel (Logical TCP Connection)
SSH Channel (Logical TCP Connection)
SSH Channel (Logical TCP Connection)
TCP Server to Endpoint 1
TCP Server to Endpoint 2
Client 1 pool
TCP Connection
TCP Connection
TCP Connection
Chisel HTTP Server
Client 2 pool
TCP Connection
TCP Connection
TCP Connection
Remote Endpoints
1
2
3
Chisel HTTP Client 2
TCP / HTTP / WebSocket / SSH Layers
TCP Server to Endpoint 2
TCP Server to Endpoint 3
SSH Channel (Logical TCP Connection)
SSH Channel (Logical TCP Connection)
SSH Channel (Logical TCP Connection)

Table of Contents

- Features
- Install
- Demo
- Usage
- Contributing
- Changelog
- License

Features

+ 24 contributors

Languages

● Go 98.5% ● Makefile 1.5%

- Easy to use
- Performant*
- Encrypted connections using the SSH protocol (via `crypto/ssh`)
- Authenticated connections; authenticated client connections with a users config file, authenticated server connections with fingerprint matching.
- Client auto-reconnects with exponential backoff
- Clients can create multiple tunnel endpoints over one TCP connection
- Clients can optionally pass through SOCKS or HTTP CONNECT proxies
- Reverse port forwarding (Connections go through the server and out the client)
- Server optionally doubles as a reverse proxy
- Server optionally allows SOCKS5 connections (See guide below)
- Clients optionally allow SOCKS5 connections from a reversed port forward
- Client connections over stdio which supports `ssh -o ProxyCommand` providing SSH over HTTP

Install

Binaries

release v1.10.1 downloads 3M

See the latest release or download and install it now with `curl https://i.jpillora.com/chisel! | bash`

Docker

docker pulls 19M image size 7.82 MB

```
docker run --rm -it jpillora/chisel --help
```

Fedora

The package is maintained by the Fedora community. If you encounter issues related to the usage of the RPM, please use this issue tracker.

```
sudo dnf -y install chisel
```

Source

```
$ go install github.com/jpillora/chisel@latest
```

Demo

A demo app on Heroku is running this `chisel server`:

```
$ chisel server --port $PORT --proxy http://exa
# listens on $PORT, proxy web requests to http:/
```

This demo app is also running a simple file server on `:3000`, which is normally inaccessible due to Heroku's firewall. However, if we tunnel in with:

```
$ chisel client https://chisel-demo.herokuapp.c
# connects to chisel server at https://chisel-d
# tunnels your localhost:3000 to the server's l
```

and then visit localhost:3000, we should see a directory listing. Also, if we visit the demo app in the browser we should hit the server's default proxy and see a copy of example.com.

Usage

```
$ chisel --help

  Usage: chisel [command] [--help]

  Version: X.Y.Z

  Commands:
    server - runs chisel in server mode
    client - runs chisel in client mode

  Read more:
    https://github.com/jpillora/chisel
```

```
$ chisel server --help

  Usage: chisel server [options]

  Options:

    --host, Defines the HTTP listening host – th
    (defaults the environment variable HOST and

    --port, -p, Defines the HTTP listening port
    variable PORT and fallsback to port 8080).

    --key, (deprecated use --keygen and --keyfil
    An optional string to seed the generation of
    and private key pair. All communications wil
    key pair. Share the subsequent fingerprint w
    of man-in-the-middle attacks (defaults to th
    variable, otherwise a new key is generate ea

    --keygen, A path to write a newly generated
    If users depend on your --key fingerprint, y
    output your existing key. Use - (dash) to ou

    --keyfile, An optional path to a PEM-encoded
    this flag is set, the --key option is ignore
    is used to secure all communications. (defau
    environment variable). Since ECDSA keys are
    to an inline base64 private key (e.g. chisel

    --authfile, An optional path to a users.json
    be an object with users defined like:
      {
        "<user:pass>": ["<addr-regex>","<addr-r
      }
    when <user> connects, their <pass> will be
    each of the remote addresses will be compar
    of address regular expressions for a match.
    always come in the form "<remote-host>:<rem
    and "R:<local-interface>:<local-port>" for
    remotes. This file will be automatically re

    --auth, An optional string representing a s
    access, in the form of <user:pass>. It is e
    authfile with {"<user:pass>": [""]}. If uns
    environment variable AUTH.

    --keepalive, An optional keepalive interval
    transport is HTTP, in many instances we'll
    proxies, often these proxies will close idl
    specify a time with a unit, for example '5s
    to '25s' (set to 0s to disable).

    --backend, Specifies another HTTP server to
    chisel receives a normal HTTP request. Usef
    plain sight.

    --socks5, Allow clients to access the inter
    chisel client --help for more information.

    --reverse, Allow clients to specify reverse
    in addition to normal remotes.

    --tls-key, Enables TLS and provides optiona
    TLS private key. When this flag is set, you
    and you cannot set --tls-domain.

    --tls-cert, Enables TLS and provides option
    TLS certificate. When this flag is set, you
    and you cannot set --tls-domain.

    --tls-domain, Enables TLS and automatically
    certificate using LetsEncrypt. Setting --tl
    You may specify multiple --tls-domain flags
    The resulting files are cached in the "$HOM
    You can modify this path by setting the CHI
    or disable caching by setting this variable
    provide a certificate notification email by

    --tls-ca, a path to a PEM encoded CA certifi
    holding multiple PEM encode CA certificate b
    validate client connections. The provided CA
    instead of the system roots. This is commonl

    --pid Generate pid file in current working d

    -v, Enable verbose logging

    --help, This help text

  Signals:
    The chisel process is listening for:
      a SIGUSR2 to print process stats, and
      a SIGHUP to short-circuit the client recon

  Version:
    X.Y.Z

  Read more:
    https://github.com/jpillora/chisel
```

```
$ chisel client --help

  Usage: chisel client [options] <server> <remot

  <server> is the URL to the chisel server.

  <remote>s are remote connections tunneled thro
  which come in the form:

    <local-host>:<local-port>:<remote-host>:<rem

    ■ local-host defaults to 0.0.0.0 (all inter
    ■ local-port defaults to remote-port.
    ■ remote-port is required*.
    ■ remote-host defaults to 0.0.0.0 (server lo
    ■ protocol defaults to tcp.

  which shares <remote-host>:<remote-port> from
  as <local-host>:<local-port>, or:
    R:<local-interface>:<local-port>:<remote-ho

  which does reverse port forwarding, sharing <
  from the client to the server's <local-interf

    example remotes

      3000
      example.com:3000
      3000:google.com:80
      192.168.0.5:3000:google.com:80
      socks
      5000:socks
      R:2222:localhost:22
      R:socks
      R:5000:socks
      stdio:example.com:22
      1.1.1.1:53/udp

    When the chisel server has --socks5 enabled
    specify "socks" in place of remote-host and
    The default local host and port for a "sock
    127.0.0.1:1080. Connections to this remote
    at the server's internal SOCKS5 proxy.

    When the chisel server has --reverse enable
    be prefixed with R to denote that they are
    is, the server will listen and accept conne
    will be proxied through the client which sp
    Reverse remotes specifying "R:socks" will l
    default socks port (1080) and terminate the
    client's internal SOCKS5 proxy.

    When stdio is used as local-host, the tunne
    input/output of this program with the remot
    combined with ssh ProxyCommand. You can use
      ssh -o ProxyCommand='chisel client chisel
          user@example.com
    to connect to an SSH server through the tun

  Options:

    --fingerprint, A *strongly recommended* fin
    to perform host-key validation against the
        Fingerprint mismatches will close the c
        Fingerprints are generated by hashing t
        SHA256 and encoding the result in base6
        Fingerprints must be 44 characters conta

    --auth, An optional username and password (
    in the form: "<user>:<pass>". These credent
    the credentials inside the server's --authf
    AUTH environment variable.

    --keepalive, An optional keepalive interval
    transport is HTTP, in many instances we'll 
    proxies, often these proxies will close idl
    specify a time with a unit, for example '5s
    to '25s' (set to 0s to disable).

    --max-retry-count, Maximum number of times 
    Defaults to unlimited.

    --max-retry-interval, Maximum wait time bef
    disconnection. Defaults to 5 minutes.

    --proxy, An optional HTTP CONNECT or SOCKS5
    used to reach the chisel server. Authentica
    inside the URL.
    For example, http://admin:password@my-serve
            or: socks://admin:password@my-serve

    --header, Set a custom header in the form "
    Can be used multiple times. (e.g --header "

    --hostname, Optionally set the 'Host' heade
    found in the server url).

    --sni, Override the ServerName when using T
    hostname).

    --tls-ca, An optional root certificate bund
    chisel server. Only valid when connecting t
    "https" or "wss". By default, the operating

    --tls-skip-verify, Skip server TLS certific
    chain and host name (if TLS is used for tra
    server). If set, client accepts any TLS cer
    the server and any host name in that certif
    transport https (wss) connection. Chisel se
    may be still verified (see --fingerprint) a
    is established.
```

```
    --tls-key, a path to a PEM encoded private
    authentication (mutual-TLS).

    --tls-cert, a path to a PEM encoded certifi
    private key. The certificate must have clie
    enabled (mutual-TLS).

    --pid Generate pid file in current working

    -v, Enable verbose logging

    --help, This help text

  Signals:
    The chisel process is listening for:
      a SIGUSR2 to print process stats, and
      a SIGHUP to short-circuit the client reco

  Version:
    X.Y.Z

  Read more:
    https://github.com/jpillora/chisel
```

Security

Encryption is always enabled. When you start up a chisel server, it will generate an in-memory ECDSA public/private key pair. The public key fingerprint (base64 encoded SHA256) will be displayed as the server starts. Instead of generating a random key, the server may optionally specify a key file, using the `--keyfile` option. When clients connect, they will also display the server's public key fingerprint. The client can force a particular fingerprint using the `--fingerprint` option. See the `--help` above for more information.

Authentication

Using the `--authfile` option, the server may optionally provide a `user.json` configuration file to create a list of accepted users. The client then authenticates using the `--auth` option. See users.json for an example authentication configuration file. See the `--help` above for more information.

Internally, this is done using the *Password* authentication method provided by SSH. Learn more about `crypto/ssh` here http://blog.gopheracademy.com/go-and-ssh/.

SOCKS5 Guide with Docker

1. Print a new private key to the terminal

```
chisel server --keygen -
# or save it to disk --keygen /path/to/mykey
```

2. Start your chisel server

```
jpillora/chisel server --keyfile '<ck-base64
```

3. Connect your chisel client (using server's fingerprint)

```
chisel client --fingerprint '<see server out
```

4. Point your SOCKS5 clients (e.g. OS/Browser) to:

```
<client-address>:1080
```

5. Now you have an encrypted, authenticated SOCKS5 connection over HTTP

Caveats

Since WebSockets support is required:

- IaaS providers all will support WebSockets (unless an unsupporting HTTP proxy has been forced in front of you, in which case I'd argue that you've been downgraded to PaaS)

- PaaS providers vary in their support for WebSockets
  - Heroku has full support
  - Openshift has full support though connections are only accepted on ports 8443 and 8080
  - Google App Engine has **no** support (Track this on their repo)

Contributing

- http://golang.org/doc/code.html
- http://golang.org/doc/effective_go.html
- `github.com/jpillora/chisel/share` contains the shared package
- `github.com/jpillora/chisel/server` contains the server package
- `github.com/jpillora/chisel/client` contains the client package

Changelog

- `1.0` - Initial release
- `1.1` - Replaced simple symmetric encryption for ECDSA SSH
- `1.2` - Added SOCKS5 (server) and HTTP CONNECT (client) support
- `1.3` - Added reverse tunnelling support
- `1.4` - Added arbitrary HTTP header support
- `1.5` - Added reverse SOCKS support (by @aus)
- `1.6` - Added client stdio support (by @BoleynSu)
- `1.7` - Added UDP support
- `1.8` - Move to a `scratch` Docker image

- `1.9` - Bump to Go 1.21. Switch from `--key` seed to P256 key strings with `--key{gen,file}` (by @cmenginnz)
- `1.10` - Bump to Go 1.22. Add `.rpm` `.deb` and `.akp` to