jpillora / chisel Public

Notifications Fork 1.4k Star 13.2k

Code Issues 183 Pull requests 49 Actions Projects Wiki Security 1 Insights

master Go to file Code

jpillora always test with latest stable ab8f06a · 2 months ago 239 Commits

| | | |
|---|---|---|
| .github | always test with latest stable | 2 months ago |
| client | chore: remove refs to deprecated io/io... | last year |
| example | move chisel to flyio | last year |
| server | Bump to Go 1.22. Add `.rpm` `.deb` and .... | 3 months ago |
| share | chore: remove refs to deprecated io/io... | last year |
| test | chore: remove refs to deprecated io/io... | last year |
| .gitignore | chore(make): update release and all to l... | 3 years ago |
| LICENSE | Bump to Go 1.22. Add `.rpm` `.deb` and .... | 3 months ago |
| Makefile | chore(make): update release and all to l... | 3 years ago |
| README.md | Bump to Go 1.22. Add `.rpm` `.deb` and .... | 3 months ago |
| go.mod | Bump golang.org/x/sync from 0.3.0 to ... | last year |
| go.sum | Bump golang.org/x/sync from 0.3.0 to ... | last year |
| main.go | Bump to Go 1.22. Add `.rpm` `.deb` and .... | 3 months ago |

## 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
last month

+ 31 releases

## Packages

No packages published

## Used by 1.5k

+ 1,502

## Contributors 38

+ 24 contributors

## Languages

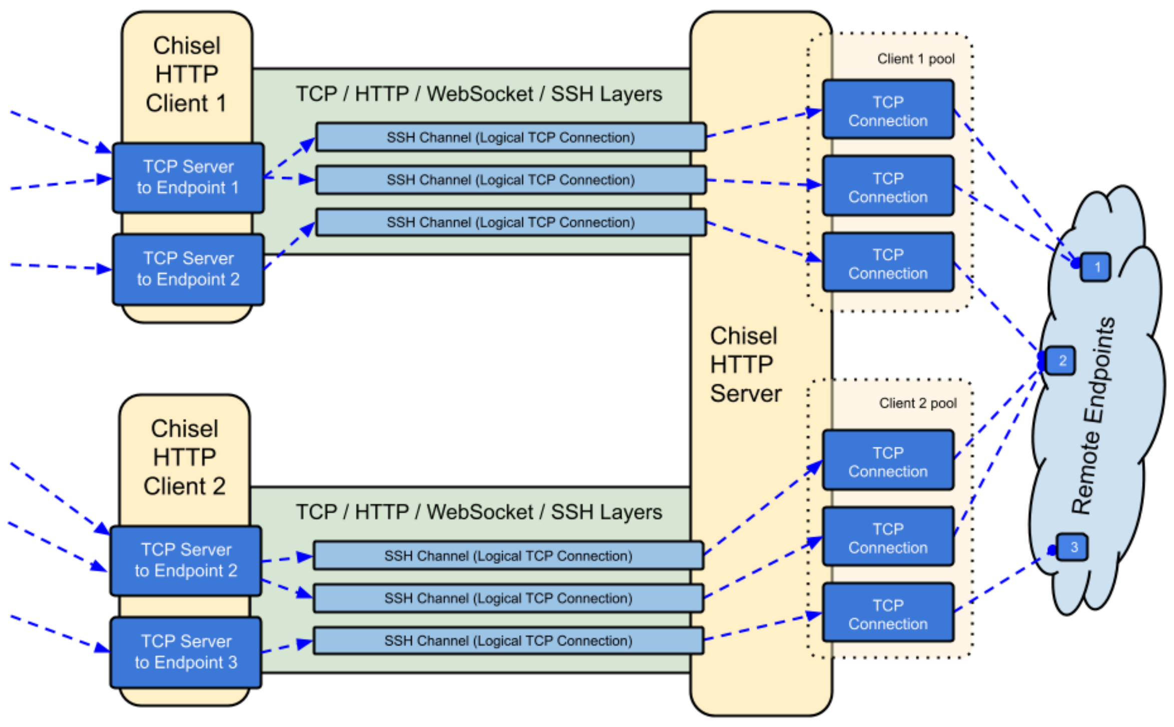
● Go 98.5% ● Makefile 1.5%

README MIT license

# 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.



## Table of Contents

## Features

- 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://example.com
# listens on $PORT, proxy web requests to http://example.com
```

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.com 3000
# connects to chisel server at https://chisel-demo.herokuapp.com,
# tunnels your localhost:3000 to the server's localhost:3000
```

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 – the network interface
    (defaults the environment variable HOST and falls back to 0.0.0.

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

    --key, (deprecated use --keygen and --keyfile instead)
    An optional string to seed the generation of a ECDSA public
    and private key pair. All communications will be secured using t
    key pair. Share the subsequent fingerprint with clients to enabl
    of man-in-the-middle attacks (defaults to the CHISEL_KEY environ
    variable, otherwise a new key is generate each run).
    --keygen, A path to write a newly generated PEM-encoded SSH priva
    If users depend on your --key fingerprint, you may also include y
    output your existing key. Use - (dash) to output the generated ke

    --keyfile, An optional path to a PEM-encoded SSH private key. Whe
    this flag is set, the --key option is ignored, and the provided p
    is used to secure all communications. (defaults to the CHISEL_KEY
    environment variable). Since ECDSA keys are short, you may also s
    to an inline base64 private key (e.g. chisel server --keygen - |

    --authfile, An optional path to a users.json file. This file shou
    be an object with users defined like:
      {
        "<user:pass>": ["<addr-regex>","<addr-regex>"]
      }
    when <user> connects, their <pass> will be verified and then
    each of the remote addresses will be compared against the list
    of address regular expressions for a match. Addresses will
    always come in the form "<remote-host>:<remote-port>" for normal
    and "R:<local-interface>:<local-port>" for reverse port forwardin
    remotes. This file will be automatically reloaded on change.

    --auth, An optional string representing a single user with full
    access, in the form of <user:pass>. It is equivalent to creating
    authfile with {"<user:pass>": [""]}. If unset, it will use the
    environment variable AUTH.

    --keepalive, An optional keepalive interval. Since the underlyin
    transport is HTTP, in many instances we'll be traversing through
    proxies, often these proxies will close idle connections. You mu
    specify a time with a unit, for example '5s' or '2m'. Defaults
    to '25s' (set to 0s to disable).

    --backend, Specifies another HTTP server to proxy requests to whe
    chisel receives a normal HTTP request. Useful for hiding chisel i
    plain sight.

    --socks5, Allow clients to access the internal SOCKS5 proxy. See
    chisel client --help for more information.

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

    --tls-key, Enables TLS and provides optional path to a PEM-encode
    TLS private key. When this flag is set, you must also set --tls-c
    and you cannot set --tls-domain.

    --tls-cert, Enables TLS and provides optional path to a PEM-enco
    TLS certificate. When this flag is set, you must also set --tls-k
    and you cannot set --tls-domain.

    --tls-domain, Enables TLS and automatically acquires a TLS key an
    certificate using LetsEncrypt. Setting --tls-domain requires por
    You may specify multiple --tls-domain flags to serve multiple dom
    The resulting files are cached in the "$HOME/.cache/chisel" dire
    You can modify this path by setting the CHISEL_LE_CACHE variable
    or disable caching by setting this variable to "-". You can opti
    provide a certificate notification email by setting CHISEL_LE_EMA

    --tls-ca, a path to a PEM encoded CA certificate bundle or a dire
    holding multiple PEM encode CA certificate bundle files, which is
    validate client connections. The provided CA certificates will be
    instead of the system roots. This is commonly used to implement m

    --pid Generate pid file in current working directory

    -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 reconnect timer
  Version:
    X.Y.Z

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

```
$ chisel client --help

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

  <server> is the URL to the chisel server.

  <remote>s are remote connections tunneled through the server, each
  which come in the form:

    <local-host>:<local-port>:<remote-host>:<remote-port>/<protocol>

    ■ local-host defaults to 0.0.0.0 (all interfaces).
    ■ local-port defaults to remote-port.
    ■ remote-port is required*.
    ■ remote-host defaults to 0.0.0.0 (server localhost).
    ■ protocol defaults to tcp.

  which shares <remote-host>:<remote-port> from the server to the cl
  as <local-host>:<local-port>, or:

    R:<local-interface>:<local-port>:<remote-host>:<remote-port>/<pr

  which does reverse port forwarding, sharing <remote-host>:<remote-
  from the client to the server's <local-interface>:<local-port>.

    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, remotes can
    specify "socks" in place of remote-host and remote-port.
    The default local host and port for a "socks" remote is
    127.0.0.1:1080. Connections to this remote will terminate
    at the server's internal SOCKS5 proxy.

    When the chisel server has --reverse enabled, remotes can
    be prefixed with R to denote that they are reversed. That
    is, the server will listen and accept connections, and they
    will be proxied through the client which specified the remote.
    Reverse remotes specifying "R:socks" will listen on the server's
    default socks port (1080) and terminate the connection at the
    client's internal SOCKS5 proxy.

    When stdio is used as local-host, the tunnel will connect standa
    input/output of this program with the remote. This is useful whe
    combined with ssh ProxyCommand. You can use
      ssh -o ProxyCommand='chisel client chiselserver stdio:%h:%p' \
          user@example.com
    to connect to an SSH server through the tunnel.

  Options:

    --fingerprint, A *strongly recommended* fingerprint string
    to perform host-key validation against the server's public key.
        Fingerprint mismatches will close the connection.
        Fingerprints are generated by hashing the ECDSA public key u
        SHA256 and encoding the result in base64.
        Fingerprints must be 44 characters containing a trailing equ

    --auth, An optional username and password (client authentication
    in the form: "<user>:<pass>". These credentials are compared to
    the credentials inside the server's --authfile. defaults to the
    AUTH environment variable.

    --keepalive, An optional keepalive interval. Since the underlyin
    transport is HTTP, in many instances we'll be traversing through
    proxies, often these proxies will close idle connections. You mu
    specify a time with a unit, for example '5s' or '2m'. Defaults
    to '25s' (set to 0s to disable).

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

    --max-retry-interval, Maximum wait time before retrying after a
    disconnection. Defaults to 5 minutes.

    --proxy, An optional HTTP CONNECT or SOCKS5 proxy which will be
    used to reach the chisel server. Authentication can be specified
    inside the URL.
    For example, http://admin:password@my-server.com:8081
            or: socks://admin:password@my-server.com:1080

    --header, Set a custom header in the form "HeaderName: HeaderCon
    Can be used multiple times. (e.g --header "Foo: Bar" --header "H

    --hostname, Optionally set the 'Host' header (defaults to the ho
    found in the server url).

    --sni, Override the ServerName when using TLS (defaults to the
    hostname).

    --tls-ca, An optional root certificate bundle used to verify the
    chisel server. Only valid when connecting to the server with
    "https" or "wss". By default, the operating system CAs will be u

    --tls-skip-verify, Skip server TLS certificate verification of
    chain and host name (if TLS is used for transport connections to
    server). If set, client accepts any TLS certificate presented by
    the server and any host name in that certificate. This only affe
    transport https (wss) connection. Chisel server's public key
    may be still verified (see --fingerprint) after inner connection
    is established.

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

    --tls-cert, a path to a PEM encoded certificate matching the pro
    private key. The certificate must have client authentication
    enabled (mutual-TLS).

    --pid Generate pid file in current working directory

    -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 reconnect timer

  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 string or file path:
```

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

```
chisel client --fingerprint '<see server output>' <server-addres:
```

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

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information