Configuring Hitch

Hitch can be configured either from command line arguments or from a configuration file on disk.

You can extract the usage description by invoking Hitch with the "--help" argument. An example configuration file is included in the distribution.

In general Hitch is a protocol agnostic proxy and does not need much configuration.

List of configuration items to consider:

- PEM files with key and certificate.
- Listening addresses and ports. Note the semi-odd square brackets for IPv4 addresses.
- Which backend servers to proxy towards, and if PROXY protocol should be used.
- Number of workers, usually 1. For larger setups, use one worker per core.

If you need to support legacy clients, you can consider:

- Enable SSLv3 with "--ssl" (despite RFC7568.)
- Use weaker ciphers.

Specifying ciphers

The recommended default is:

    "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH"

If you need to support legacy clients, consider the "HIGH" cipher group.

Hitch cipher list string format is identical to that of other servers, so you can use tools like https://mozilla.github.io/server-side-tls/ssl-config-generator/ to generate a set of ciphers that suits your needs.

Normally you do not have to change this.

Run environment

If you're handling a large number of connections, you'll probably want to raise *ulimit -n* before running Hitch.

If you are listening to ports under 1024 (443 comes to mind), you need to start Hitch as root. In those cases you *must* use --user/-u to set a non-privileged user *hitch* can setuid() to.

Preparing PEM files

PEM files should contain the key file, the certificate from the CA and any intermediate CAs needed.

    $ cat example.com.key example.com.crt intermediate.pem > example.com.pem

If you want to use Diffie-Hellman based ciphers for Perfect Forward Secrecy (PFS), you need to add some parameters for that as well:

    $ openssl dhparam -rand - 2048 >> example.com.pem

Hitch will complain and disable DH unless these parameters are available.

Automatic OCSP staple retrieval

Hitch has support for automated retrieval of OCSP responses from an OCSP responder.

To set this up, specify the following setting in your configuration file:

    ocsp-dir = "/var/lib/hitch-ocsp"

This can also be configured via the command line option `--ocsp-dir=mydir`.

If the loaded certificate contains an OCSP responder address and it also has the required issuer certificate as part of its chain, Hitch will automatically retrieve and refresh OCSP staples.

The *ocsp-dir* directory must be read/write accessible by the configured hitch user, and should not be read or write accessible by any other user.

The staples are fetched asynchronously, and will be loaded and ready for stapling as soon as they are available.

The variables `ocsp-connect-tmo` and `ocsp-resp-tmo` controls respectively the connect timeout and fetch transmission timeout when Hitch is talking to an OCSP responder.

Verification of OCSP staples

Hitch will optionally verify the OCSP staple, this can be done by specifying

```
ocsp-verify-staple = on
```

in the configuration file.

If you are running with a custom CA, the verification certificates can be changed by setting the *SSL_CERT_FILE* or *SSL_CERT_DIR* environment variables. *SSL_CERT_FILE* can point to a single pem file containing a chain of certificates, while the *SSL_CERT_DIR* can be a comma-separated list of directories containing pem file with symlinks by their hash key (see the man page of c_rehash from the OpenSSL library for more information).

OCSP stapling from manually pre-loaded files

Hitch also has support for stapling of OCSP responses loaded from files on disk. If configured, Hitch will include a stapled OCSP response as part of the handshake when it receives a status request from a client.

Retrieving an OCSP response suitable for use with Hitch can be done using the following *openssl* command:

```
$ openssl ocsp
    -url https://ocsp.example.com -header Host ocsp.example.com -no_nonce -resp_text -issuer issuer.pem -cert mycert.pem -respout ocspresp.der
```

This will produce a DER-encoded OCSP response which can then be loaded by Hitch.

The URL of the OCSP responder can be retrieved via

```
$ openssl x509 -ocsp_uri -in mycert.pem -noout
```

the *-issuer* argument needs to point to the OCSP issuer certificate. Typically this is the same certificate as the intermediate that signed the server certificate.

To configure Hitch to use the OCSP staple, use the following incantation when specifying the *pem-file* setting in your Hitch configuration file:

```
pem-file = {
    cert = "mycert.pem" ocsp-resp-file = "mycert-ocsp.der"
}
```

ALPN/NPN support

Hitch supports both the ALPN and the NPN TLS extension. This allows negotiation of the application layer protocol that is to be used.

This is useful if Hitch terminates TLS for HTTP/2 traffic.

To turn this on, you must supply an *alpn-protos* setting in the configuration file:

```
alpn-protos = "h2,http/1.1"
```

If the PROXY protocol is enabled (*write-proxy* = *on*), Hitch will transmit the selected protocol as part of its PROXY header.

SSL/TLS protocol setting

Hitch supports TLS (1.0, 1.1 and 1.2) and SSL 3. By default, only TLS versions 1.1 and 1.2 are enabled, while TLS 1.0 and SSLv3 are disabled. The recommended way to to select protocols is to use *tls-protos* in the configuration file:

    tls-protos = TLSv1.1 TLSv1.2

The following tokens are available for the *tls-protos* option: *SSLv3*, *TLSv1.0*, *TLSv1.1* and *TLSv1.2*.

Uninterrupted configuration reload

Issuing a SIGHUP signal to the main Hitch process will initiate a reload of Hitch's configuration file.

Adding, updating and removing PEM files (`pem-file`) and frontend listen endpoints (`frontend`) is currently supported.

Hitch will load the new configuration in its main process, and spawn a new set of child processes with the new configuration in place if successful.

The previous set of child processes will finish their handling of any live connections, and exit after they are done.

If the new configuration fails to load, an error message will be written to syslog. Operation will continue without interruption with the current set of worker processes.