Configuring Stubby

Configure Stubby

It is recommended to use the default configuration file provided which will use 'Strict' privacy mode and spread the DNS queries among several of the current DNS Privacy test servers. Note that this file contains both IPv4 and IPv6 addresses.

Note also that this file only enables a small subset of the available servers by default. Users can choose to use additional servers by uncommenting the relevant sections in the file. See DNS Privacy Test Servers for details of the available servers.

Create Custom Configuration File

Alternatively the configuration file location can be specified on the command line using the `-C` flag. Changes to the configuration file require a restart of Stubby.

From release 0.1.3 of stubby (using the 1.2 release of getdns) the configuration file format is a YAML like format, which contains detailed comments of the options. See here for an example or keep reading for a short description. *To be backwards compatible a file using the YAML format **must** have a .yml or .yaml extension.*

Essentially the options available are the same as the options that can be set on a getdns **`context.`**

- Doxygen documentation for the **`context`** options is available here.
- A detailed description of the context settings is in the API
- A detailed description of all the extensions than can also be used is also in the API.

The previously used JSON like format used internally in getdns is still supported but only when the configuration file is specified via the command line using the -C flag. This format is the same as the output returned by `stubby -i`. For example, this output can be used as a configuration file directly, but a less verbose form is also accepted.

To aid with creating a custom configuration file, an example is given below.

The config file below will configure Stubby in the following ways:

- **`resolution_type`:** Work in stub mode only (not recursive mode) - required for Stubby operation.
- **`dns_transport_list`:** Use TLS only as a transport (no fallback to UDP or TCP).
- **`tls_authentication`:** Use Strict Privacy i.e. require a TLS connection and authentication of the upstream
    - If Opportunistic mode is desired, remove the *tls_authentication: GETDNS_AUTHENTICATION_REQUIRED* field and add additional transports to the *dns_transport_list* (e.g. UDP, TCP) . In Opportunistic mode authentication of the nameserver is not required and fallback to clear text transports is permitted **if** they are in the dns_transport_list.
- **`tls_query_padding_blocksize`:** Use the EDNS0 padding option to pad DNS queries to hide their size
- **`edns_client_subnet_private`:** Use EDNS0 Client Subnet privacy so the client subnet is not sent to authoritative servers
- **`listen_address`:** have the Stubbby daemon listen on IPv4 and IPv6 on port 53 on the loopback address
- **`idle_timeout`:** Use an EDNS0 Keepalive idle timeout of 10s unless overridden by the server. This keeps idle TLS connections open to avoid the overhead of opening a new connection for every query.
- **`round_robin_upstreams`:** Round robin queries across all the configured upstream servers. Without this option Stubby will use each upstream server sequentially until it becomes unavailable and then move on to use the next.
- **`upstream_recursive_servers`:** Use the NLnet labs test DNS Privacy Server for outgoing queries. In Strict Privacy mode, at least one of the following is required for each nameserver:
    - **`tls_auth_name`:** This is the authentication domain name that will be verified against the presented certificate.
    - **`tls_pubkey_pinset`:** The sha256 SPKI pinset for the server. This is also verified against the presented certificate.

```
resolution_type: GETDNS_RESOLUTION_STUB
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
tls_authentication: GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize: 256
edns_client_subnet_private : 1
idle_timeout: 10000
listen_addresses:
  - 127.0.0.1
  -  0::1
round_robin_upstreams: 1
upstream_recursive_servers:
  - address_data: 185.49.141.38
    tls_auth_name: "getdnsapi.net"
    tls_pubkey_pinset:
      digest: "sha256"
       value: foxZRnIh9gZpWnl+zEiKa0EJ2rdCGroMWm02gaxSc9Q=
```

Additional privacy servers can be specified by adding more entries to the `upstream_recursive_servers` list above (note a separate entry must be made for the IPv4 and IPv6 addresses of a given server. More DNS Privacy test servers are listed here.

A custom port can be specified by adding the `tls_port:` attribute to the `upstream_recursive_server` in the config file.

A custom listen address port can be configured by using the <IP_address>@<port> syntax

DNSSEC

To enable DNSSEC validation when using Stubby add the following option to the configuration file

```
dnssec_return_status: GETDNS_EXTENSION_TRUE
```

A trust anchor is also required for DNSSEC validation.

getdns version 1.2 and later include support for automatic trust anchor management - which will automatically fetch a trust anchor if none is present on the system. See 'Zero configuration DNSSEC' (and below) for the specific details of key management for DNSSEC for this case.

> If using a version of *getdns* earlier than 1.2 then a trust anchor must be **manually** installed and managed on the system. We recommend using *unbound-anchor*.

Storage of Zero-config Trust anchor

When the system-level user does have a home directory, stubby will store the for Zero configuration DNSSEC dynamically acquired root trust anchor in a subdirectory called ".getdns" of that home directory. If the system-level user does not have a home directory or the home directory is not writeable or readable, stubby will fallback to the current working directory.

This can be overruled by supplying a "`appdata_dir`" in the `stubby.yml` configuration file. When a "`appdata_dir`" was specified, that directory will be used for storing data related to Zero configuration DNSSEC immediately, without the other paths being tried. It is recommended for systemd setups using the provided systemd.service file(s) to have a "`appdata_dir`" directive set to "`/var/cache/stubby`" in the stubby.yml configuration file.

> Note that using DNSSEC can add a small performance overhead because it increases the number of queries required to resolve a DNS request.

Opportunistic DoT to your local resolver

Some users may want to have a configuration for Stubby that will always use the resolver from the system configuration (most likely but not always on the local network), but with encryption used where possible. This is an **Opportunistic** mode which does not authenticate the DoT server. To configure Stubby in this mode:

1. Set the transport list and authentication parameter in the configuration to:

```
dns_transport_list:
  - GETDNS_TRANSPORT_TLS
  - GETDNS_TRANSPORT_UDP
  - GETDNS_TRANSPORT_TCP
tls_authentication: GETDNS_AUTHENTICATION_NONE
```

2. Remove (or comment out) all the upstream_resolvers. This will cause Stubby to fallback to using the system resolvers only.

Note: a future version of Stubby will most likely support a mixed mode of system resolvers and configured resolvers.

Runtime logging

In the 0.1.2 release of stubby there is runtime logging, which can be turned on by using the '-l' flag.

In the 0.1.3 release the logging level is be controlled by specifying a logging level via the '-v' flag. See stubby help ('stubby -h') for more details.