Thin client specification

David Stainton

Abstract

This document describes the design of the new Katzenpost mixnet *thin client*. In particular we discuss its multiplexing and privilege-separation design elements as well as the protocol used by the thin-client library.

Introduction

A Katzenpost mixnet client has several responsibilities at a minimum:

- composing Sphinx packets
- decrypting SURB replies
- sending and receiving Noise protocol messages
- keeping up to date with the latest PKI document

The thin-client library enables applications to talk with the connector daemon and in that way interact with the mix network. The library itself does not do any mixnet-related cryptography since that is already handled by the thin-client daemon. In particular, the daemon strips cryptographic signatures from the PKI document before passing data to clients using the connector library. Noise- and Sphinx-related cryptography are also handled by the daemon.

Thin-client library and daemon protocol

The thin-client daemon protocol uses a local network socket, either a Unix domain socket or TCP.

Messages

There are two protocol message types, both CBOR encoded.

- The client sends `Request` messages.
- The daemon sends `Response` messages.

Messages are length-prefixed CBOR blobs, that is, a blob is prefixed with a big-endian unsigned four-byte integer (uint32) that encodes the blob length.

Protocol description

The protocol has two phases.

Phase 1

Upon connecting to the daemon socket, the thin client waits for two messages.

- **First message**
  - The `is_status` field must be set to **true**
  - The `is_connected` field must indicate whether or not the daemon has a mixnet PQ Noise protocol connection to a mixnet gateway node.
- **Second message**
  - The message contains the PKI document in its `payload` field. This marks the end of the initial connection sequence.
  - The PKI document is stripped of cryptographic signatures as noted above.

Phase 2

- The thin client may send `Request` messages to the daemon, which encapsulates the provided payload in a Sphinx packet and sends it to the gateway node.
- The daemon may send `Response` messages to the client at any time during this phase. `Response` messages may communicate a connection status change, a new PKI document, or a message-sent or message-reply event.