ZeroMQ and Encryption

Thomas Braun

byte physics e. K.

19. March 2024

What is ZeroMQ?

- Lightweight messaging kernel[1]
- Many Platforms
- Cross-Language
- Multi-Transport (TCP, UDP, websocket, ...)
- High-Performance
- Well documented

[1] Created by the late Peter Hintjens
https://en.wikipedia.org/wiki/Pieter_Hintjens

How is it used in Tango?[2]

- Sending/Receiving events
- Replaced omniORB notification service (notifd)
- Debut: Tango 8 (7/2012)
- ZeroMQ messaging pattern: Publisher/Subscriber

[2] cppTango webinar: https://youtu.be/gFUZidSfF9c?t=1841

- Tango clients can ask a device server (DS) about the ZeroMQ connection details
- Command: ZmqEventSubscriptionChange[3]
- Returns: Heartbeat/Event ZeroMQ endpoints

[3] https://tango-controls.readthedocs.io/en/latest/development/advanced/reference.html#the-zmqeventsubscriptionchange-command

Encryption options in ZeroMQ

- Supports encryption since 4.0[4]
- Two flavors exist: GSSAPI and CurveZMQ[5]

[4] cppTango requires 4.0.5

[5] http://hintjens.com/blog:49

GSSAPI

- Standard API interface for security services
- IETF standard, precursor implementation was Kerberos
- ZeroMQ RFC[6] status: Draft
- Requires external service (Principal)

[6] https://rfc.zeromq.org/spec/38/

CurveZMQ

- Based on CurveCP[7] scheme
- ZeroMQ RFC[8] status: Stable
- Supported crypto library: Libsodium[9], fork of NaCl [10]
- Algorithm: Curve25519
- Authentication (optional, on top): ZAP[11]

---

[7] `https://curvecp.org` by Daniel J. Bernstein

[8] `https://rfc.zeromq.org/spec/26`

[9] `https://doc.libsodium.org`

[10] `https://nacl.cr.yp.to`

[11] `https://rfc.zeromq.org/spec/27`

Digression: Curve25519

- Elliptic Curve[12]
- Perfect Forward Secrecy (PFS)
- Also used for TLS 1.3, SSH[13], Signal, Matrix, Threema, WhatsApp, Tor, Wireguard, ...

---

[12] https://media.ccc.de/v/31c3_-_6369_-_en_-_saal_1_-_201412272145_-_ecchacks_-_djb_-_tanja_lange

[13] KexAlgorithms: curve25519-sha256@libssh.org,curve25519-sha256

No question what we will use or ?

How does CurveZMQ work?

Server[14]:

- Public/private keypair: `zmq_curve_keypair()`
- Set private key: `zmq_setsockopt()` with `ZMQ_CURVE_SECRETKEY == ...`
- Mark as server: `zmq_setsockopt()` with `ZMQ_CURVE_SERVER == 1`

[14] The server/client roles here are **not** zmq socket types

Client:

- Public/private keypair: `zmq_curve_keypair()`
- Set private key: `zmq_setsockopt()` with `ZMQ_CURVE_SECRETKEY == ...`
- Mark as client: `zmq_setsockopt()` with `ZMQ_CURVE_SERVER == 0`
- Set client public key: `zmq_setsockopt()` with `ZMQ_CURVE_PUBLICKEY == ...`
- Set server public key: `zmq_setsockopt()` with `ZMQ_CURVE_SERVERKEY == ...`

How can we get the server public key form the server to the client?

Idea: Extend ZmqEventSubscriptionChange

- Tango client can ask for server public key by setting `argin[5]` to `"1"`
- DS checks `argin[5]` and if requested can then send the server public key in `svalue[n + 1]`[15]
- Tango client applies server public key and starts encrypted ZeroMQ messaging

[15] We could also disallow requesting the server's public key on unencrypted connections.

Questions ?