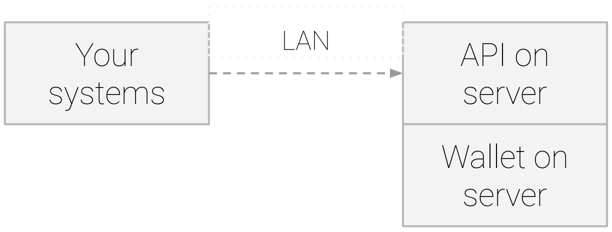
Secure Container Release
Technical details of the API

How does the API work together with the other components?

The API is the entry point for your internal systems to retrieve release data and transfer releases electronically. Your internal systems communicate within your local network with the API on your server



On the server is also a wallet that keeps your organisations' keypair. This keypair is used to sign transactions on the blockchain network and receive and send encrypted data to your business partners.

All functionality, like the wallet, the trust store that keeps the list of your trusted business partners, a cache, etc is integrated and thus fully transparent.

An installer to guide the installation process and a monitoring tool to check the status of your server are also included.

All these components are maintained and supported by the internal development team of T-Mining, which also provides second line support to your IT department or service provider.

Server requirements

These are the minimum requirements for the server on which you can install the API:

- 1 vCPU
- 8 GB RAM
- 80 GB SSD
- OS: Ubuntu LTS 18.04 (or higher), CentOS 8 (or higher), Windows Server 2012 (or higher)
- Internet access (see below)
- mDNS and NTP enabled

Network requirements

The server will connect to the nodes in the blockchain network. At least the following IP-addresses should be reachable on port 443: 37.252.121.190, 84.22.107.112, 37.252.127.71 , 84.22.115.76 on port 443.

This communication is always initiated by the server so a fixed IP or opening up inbound connections is not necessary.

Authentication

This section describes the authentication between a system at your end and the T-Mining API gateway.

After registration of the organization, the administrator must:

- install the wallet and the API gateway, which should run on the same machine
- set either a secret or a public key for the system at your end in the wallet; this can be done using the wallet CLI
- export the public key of the wallet (again using the wallet CLI) and make sure the resulting file is stored where it is readable by the system at your end

For each request to the API gateway:

- the system at your end should generate a JWT that it signs either with HMAC or with its private key, depending on the configuration of the wallet, as explained above
- the JWT must have a payload with the following claim:
  - exp: expiration time, which should be very short as the JWT should be used only once
- the JWT must be put as a bearer token in the authorization header of the request sent to the API gateway
- the API gateway will verify the signature of the JWT using the configured secret or public key in its wallet; if invalid the request is rejected

For each call made by the API gateway to the system at your end (e.g. subscription callbacks), the same principle applies, but this time the JWT is generated and signed by the API gateway.

More information on
www.securecontainerrelease.com/scr-api