Blockchain Based Threats Registry User Manual

SPHINX

A Universal Cyber Security Toolkit for Health-Care Industry

Table of contents

Table of figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

1 Introduction

SPHINX BBTR acts as a notification and interconnection tool, which allows transmitting the intelligence, gathered by the other SPHINX tools, to other components and interested actors, so it acts as a transmitter of information. In particular, this component fits the necessity of sharing information about active threats in real time between the different interested parties. It uses a private Blockchain based on Hyperledger Fabric (HLF) to store this information about threats, so only authenticated actors are allowed to interact with the BBTR. These threats are stored by using a JSON format with the following fields and example values:

```
{
  "threat_id": "th_11112",
  "description": "This is a threat description",
  "priority": 3,
  "source": "source_ip of_the_threat",
  "responsible_parties": ["person1","person2"]
}
```

2 Installation/deployment

The installation of the BBTR is divided in several steps, which install the different sub-components of the BBTR. The order is important, so they must be installed in the subsequent order. The output of the installation process is an Hyperledger Fabric Blockchain with the deployed Smart Contracts running the code of the BBTR. Also, a Hyperledger Fabric Explorer is deployed, which shows information about the transactions and blocks generated in the network. An API Rest is also installed, which allows interacting with the BBTR, in particular it allows:

- Authenticating
- Submitting a new threat
- Get stored threats

These methods allow the users to search by the different fields of the threat, even using regular expressions.

Finally, the manual for the installation of the listener is provided. This listener allows getting the last threats submitted in real time, allowing the users to be updated.

2.1 Prerequisites and hardware

For the BBTR network, this manual covers the minimum hardware/software requirements to make the installation using docker containers as Blockchain nodes, and they are:

- Disk space: 128GB
- RAM memory: 8GB
- CPU: 4-cores
- Operative System: Ubuntu 18.04
- Software:
  - Docker
  - Docker Compose
  - NVM version 8.9.0

For the BBTR Smart Contracts, GO language is required.

The BBTR listener uses NodeJS to communicate with the API Rest and retrieve the lasts threats.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

2.2 BBTR Network

First of all, it is necessary to clone the BBTR network repository by running:

```
git clone https://sphinx-repo.intracom-telecom.com/blockchain/sphinx-network
```

Then, we move to the *sphinx-network* folder and we run:

```
yarn generate
yarn start
```

The network should start working automatically, even when the computer restarts.

To stop the network, `yarn stop` must be run.

To remove the installation, `yarn clean` must be run.

2.3 BBTR Explorer

To deploy the Hyperledger Fabric Explorer, it is required to access to TECNALIA Docker repository, so before deploying you must gain access to docker registry by running:

```
docker login blockchain-docker.artifact.tecnalia.com/
```

Once this is done, run:

```
yarn explorer:start
```

It runs the Hyperledger Explorer in *http://localhost:8090*

To stop the Hyperledger Explorer, run `yarn explorer:stop`

2.4 BBTR API

To deploy the Hyperledger Fabric API container, it is required to access to TECNALIA Docker repository, so before deploying you must gain access to docker registry by running:

```
docker login registry.sphinx-repo.intracom-telecom.com/blockchain/sphinx-
rest/fabric-rest-api
```

Once this is done, run:

```
yarn rest:start
```

To stop the API Rest, run `yarn rest:stop`

2.5 BBTR Smart Contracts

First, run:

```
git clone https://sphinx-repo.intracom-telecom.com/sphinx-project/blockchain-
based-threats-registry/bbtr-cc
```

In order to install the chaincode, you must install the *fabric-tecnalia-cli* tool, which requires the following steps:

- Logging in: *https://artifact.tecnalia.com/artifactory/api/npm/blockchain-npm/*
- Pasting in .npmrc the following:

```
@fabric:registry=https://artifact.tecnalia.com/artifactory/api/npm/blockchain-npm/

//artifact.tecnalia.com/artifactory/api/npm/blockchain-
npm/:_password=<BASE64_PASSWORD>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.

```
//artifact.tecnalia.com/artifactory/api/npm/blockchain-npm/:username=<USERNAME>

//artifact.tecnalia.com/artifactory/api/npm/blockchain-
npm/:email=youremail@email.com

//artifact.tecnalia.com/artifactory/api/npm/blockchain-npm/:always-auth=true
```

Then, it is possible to install de sdk globally with:

```
npm install -g @fabric/blockchain-sdk-fabric
```

Finally install and instantiate the Smart Contracts with:

```
./install.sh

./instantiate.sh
```

If there are no errors, then the Smart Contract will be correctly installed and instantiated in the peers of the Blockchain.

2.6 BBTR Listener

First, download the repository with:

```
git clone https://sphinx-repo.intracom-telecom.com/sphinx-project/blockchain-based-threats-registry/sphinx_listener
```

Then, install all the required stuff inside de downloaded folder with:

```
npm install
```

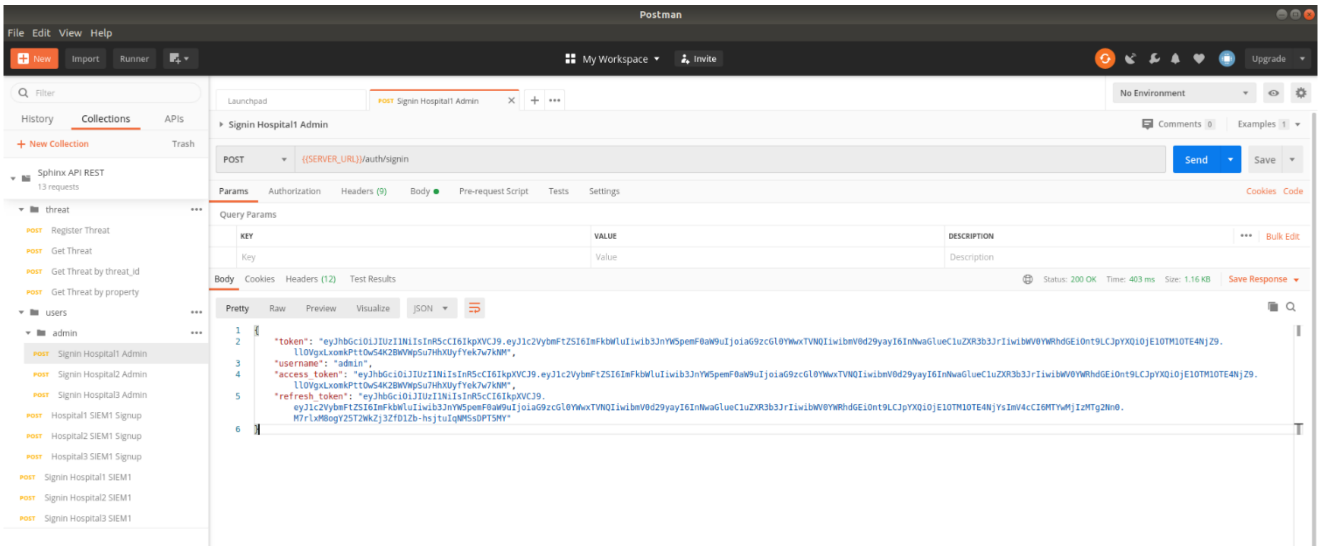
3 Operation and maintenance

For operating the BBTR, it is recommended to use Postman to work with the API REST due to its simplicity. As a consequence, Postman must be installed as a prerequisite in the system. Once it has been installed, we must import the collection into Postman. The collection file is available in the *sphinx-cc* repository, under the name of: *SPHINX API REST.postman_collection.json.* This collection includes the methods for:

- Creating BBTR users (*SignUp methods*)
- Authenticating against the BBTR (*SignIn methods*)
- Getting a threat by any field (normally threat_id)
- Getting threats by description (including regular expressions)
- Storing a threat in the BBTR

Postman has been used because of its simplicity. However, other software can be used if it supports working with HTTPS APIs. Even a *curl* call using the Unix terminal is possible to be used. In the next figure we can see the Postman interface, where the authentication process is taking place. The SignIn method is called and a token is issued by the BBTR to perform further operations against the platform. This token must be written down and put in the “Bearer token” field in subsequent calls.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.



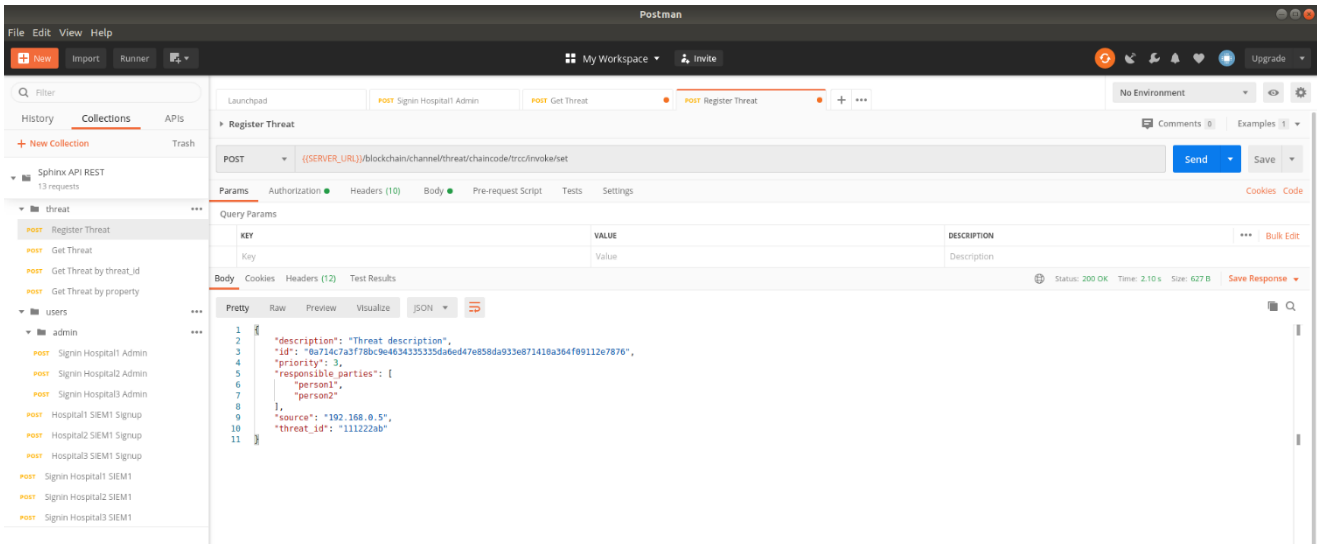
POST {{SERVER_URL}}/auth/signin

Status: 200 OK Time: 403 ms Size: 1.16 KB

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwib3JnYW5pemF0aW9uIjoiaG9zcGl0YWwxTVNQIiwibmV0d29yayI6InNwaGlueC1uZXR3b3JrIiwibWV0YWRhdGEiOnt9LCJpYXQiOjE1OTM1OTE4NjZ9.
       ll0VgxLxomkPttOwS4K2BWVWpSu7HhXUyfYek7w7kNM",
    "username": "admin",
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwib3JnYW5pemF0aW9uIjoiaG9zcGl0YWwxTVNQIiwibmV0d29yayI6InNwaGlueC1uZXR3b3JrIiwibWV0YWRhdGEiOnt9LCJpYXQiOjE1OTM1OTE4NjZ9.
       ll0VgxLxomkPttOwS4K2BWVWpSu7HhXUyfYek7w7kNM",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
       eyJ1c2VybmFtZSI6ImFkbWluIiwib3JnYW5pemF0aW9uIjoiaG9zcGl0YWwxTVNQIiwibmV0d29yayI6InNwaGlueC1uZXR3b3JrIiwibWV0YWRhdGEiOnt9LCJpYXQiOjE1OTM1OTE4NjYsImV4cCI6MTYwMjIzMTg2Nn0.
       M7rlxM8ogY25T2WkZj3ZfD1Zb-hsjtuIqNMSsDPT5MY"
}
```

Figure 1 BBTR – operation and maintenance

Once the authentication has been successfully made, it is possible to interact with the BBTR, like for example to submit a new threat, as can be seen in the next figure.



POST {{SERVER_URL}}/blockchain/channel/threat/chaincode/trcc/invoke/set

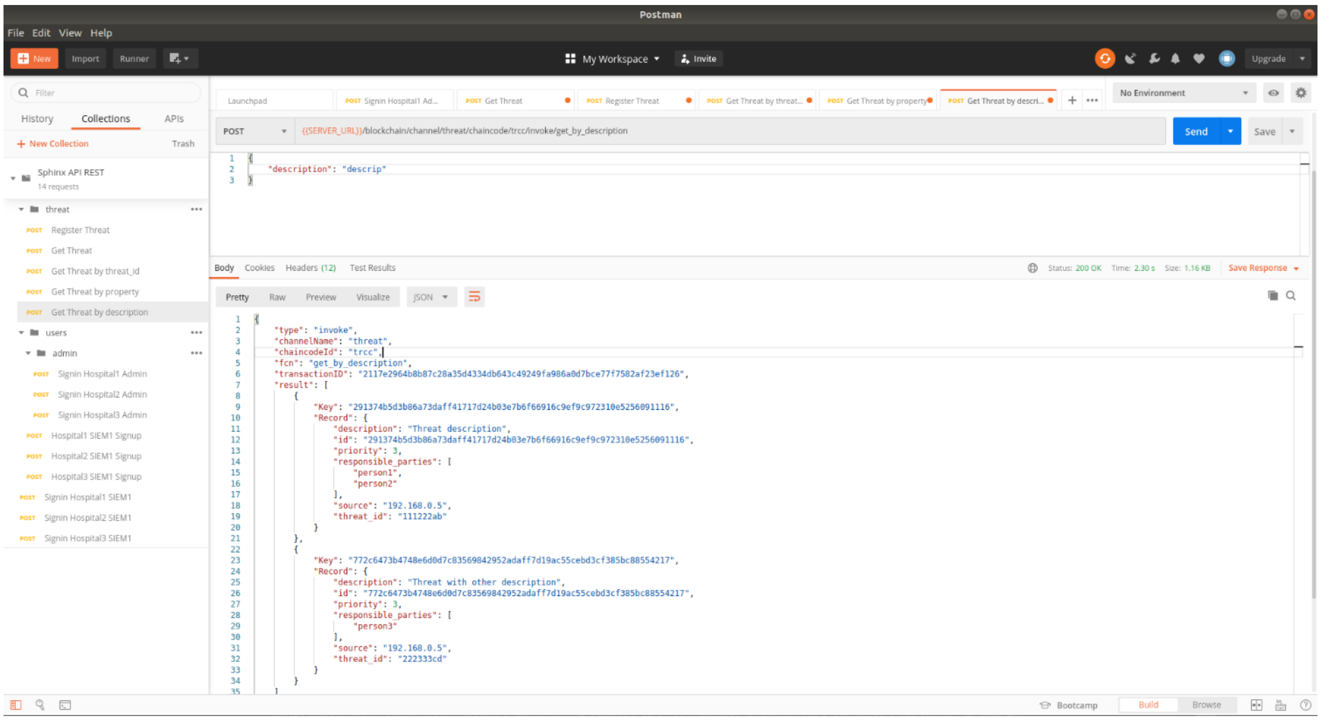
Status: 200 OK Time: 2.10 s Size: 627 B

```
{
    "description": "Threat description",
    "id": "0a714c7a3f78bc9e4634335335da6ed47e858da933e871410a364f09112e7876",
    "priority": 3,
    "responsible_parties": [
        "person1",
        "person2"
    ],
    "source": "192.168.0.5",
    "threat_id": "111222ab"
}
```

Figure 2 BBTR – operation and maintenance fig2

This method returns a 200 code if the process was successful. When issueing a threat, the format must be a JSON file. The same can be done to retrieve a group of threats which match with a query. In the next figure, we can see the "get by description" method as an example, which allows us to get a subset of threats that match with the regular expression put in the description field in the query.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.



```
POST {{SERVER_URL}}/blockchain/channel/threat/chaincode/trcc/invoke/get_by_description
```

```
{
    "description": "descrip"
}
```

```
{
    "type": "invoke",
    "channelName": "threat",
    "chaincodeId": "trcc",
    "fcn": "get_by_description",
    "transactionID": "2117e2964b8b87c28a35d4334db643c49249fa986a0d7bce77f7582af23ef126",
    "result": [
        {
            "Key": "291374b5d3b86a73daff41717d24b03e7b6f66916c9ef9c972310e5256091116",
            "Record": {
                "description": "Threat description",
                "id": "291374b5d3b86a73daff41717d24b03e7b6f66916c9ef9c972310e5256091116",
                "priority": 3,
                "responsible_parties": [
                    "person1",
                    "person2"
                ],
                "source": "192.168.0.5",
                "threat_id": "111222ab"
            }
        },
        {
            "Key": "772c6473b4748e6d0d7c83569842952adaff7d19ac55cebd3cf385bc88554217",
            "Record": {
                "description": "Threat with other description",
                "id": "772c6473b4748e6d0d7c83569842952adaff7d19ac55cebd3cf385bc88554217",
                "priority": 3,
                "responsible_parties": [
                    "person3"
                ],
                "source": "192.168.0.5",
                "threat_id": "222333cd"
            }
        }
    ]
```

Figure 3 BBTR – operation and maintenance fig 3

4 Using the listener

First, inside de *sphinx_listener* folder, you need to edit *src/config/config.json, and modify the user-password combination with a valid BBTR user:*

```
"appAdmin": "admin"
"appAdminSecret": "passwd"
```

Then, you need to copy the *cc.network-profile.yaml* from the repository *bbtr-network* (under *network-profiles/* folder*)* to *src/config,* inside the *sphinx_listener* repository. This file will tell the listener which network will connect with.

Then, you need to enrol the admin with:

```
node src/enrollAdmin.js
```

It creates a proper wallet, which stores the crypto material for connecting to the BBTR and listening for events. Finally, run the listener in CLI mode with:

```
node src/eventListener.js
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826183 - Digital Society, Trust & Cyber Security E-Health, Well-being and Ageing.