Multi-layered Security Technologies

for hyper-connected smart cities

D4.7: Application Level Security

December 2019

Grant Agreement No. 814917

Multi-layered Security technologies to ensure hyper-connected smart cities with Blockchain, BigData, Cloud and IoT

Project acronym	M-Sec
Deliverable	D4.7 Application Level Security
Work Package	WP4
Submission date	December 2019
Deliverable lead	WLI/NII
Authors	Xavier Cases (WLI), Takafumi Koumoto (NII)
Internal reviewer	Mathieu Gallissot (CEA)/Jin Nakazawa (KEIO)
Dissemination Level	Public
Type of deliverable	DEM
Version history	- V01, 09/October/2019, Xavier Cases, Creation Table of Content, Full Draft - V02, 14/November/2019, Xavier Cases, First Contribution, Demonstration 1 – Crypto Companion Database and Introduction - V03, 27/November/2019, Takafumi Koumoto, First Contribution, Demonstration 2 – Security-analysis-tool - V04, 10/December/2019, Xavier Cases, Second Contribution, Demonstration 1 – Crypto Companion Database and Introduction - V05, 16/December/2019, Takafumi Koumoto, Second Contribution, Demonstration 2 – Security-analysis-tool and Conclusion - V06, 16/December/2019, Xavier Cases, Merge of versions. - V07, 19/December/2019, Takafumi Koumoto, Third Contribution. Demonstration 2 – Security-analysis-tool in more detail. - V08, 19/December/2019, Xavier Cases, Merge and minor corrections. - V09, 20/December/2019, Mathiew Gallissot. Internal review. - V10, 24/Decembe/2019, Takafumi Koumoto, answer to comments. - V11, 29/December/2019, Jin Nakazawa. Internal review - V12, 30/December/2019, Xavier Cases&Vanessa Clemente. Version ready for submission.

The M-Sec project is jointly funded by the European Union's Horizon 2020 research and innovation programme (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

Table of Contents

List of Figures....5
Glossary....7
1. Introduction....8
1.1 Scope of the document....8
1.2 Relationship to other work packages and tasks....8
1.3 Methodology followed....8
2. Demonstration 1 – Crypto Companion Database (CCDB)....10
2.1 General Description of the Prototype....10
Components....11
Crypto Module....12
Crypto Companion DB Module....17
2.2 Package Information & Installation Instructions....24
Required Tools and dependencies....24
Install Docker....24
Install Mongo DB....26
Install Node.js and npm....26
Install Git....26

Download and Run the Demonstrator 27
User Manual 27
Licensing 30
Reference Use Case 31
Summary 31
Next Steps 31
3. Demonstration 2 – Security-analysis-tool 32
3.1 General Description of the Prototype 32
Components 33
Component Module 1: Software Security Knowledge Base 33
3.2 Package Information & Installation Instructions 39
Required Tools and dependencies 39
Install Docker 40
Install MySQL 40
Download and Run Demonstrator 40
User Manual 40
Licensing (if applicable) 40
Reference use case 40
Summary 41
Next Steps 41
4. Conclusion 42

List of Figures

Figure 1. Overall M-Sec topology 9

Figure 2. Access to distributed data. 11

Figure 3. Crypto Companion Database Module components. 12

Figure 4. Sequence diagram. Enrolment in Crypto Module. 13

Figure 5. Sequence diagram. Data encryption in Crypto Module. 14

Figure 6. Sequence diagram. Data decryption in Crypto Module. 15

Figure 7. Sequence diagram. Disenrolment in Crypto Module. 16

Figure 8. Authentication API in Crypto Companion Database Module. 17

Figure 9. Sequence diagram. Enrolment in CCDB Module. 18

Figure 10. Sequence diagram. Disenrolment in CCDB Module. 19

Figure 11. Sequence diagram. Read data in CCDB Module. 20

Figure 12. Sequence Diagram. Save data in CCDB Module. 21

Figure 13. Sequence Diagram. Delete data in CCDB Module. 22

Figure 14. Sequence diagram. Authorize in CCDB Module. 22

Figure 15. Sequence diagram. Deauthorise in CCDB Module. 23

Figure 16. Sequence diagram. Request authorization in CCDB Module. 23

Figure 17. Authentication API in Crypto Companion Database Module. 28

Figure 18. Management API in Crypto Companion Database Module. 29

Figure 19. Screenshot of the Security analysis tool. 32

Figure 20. Conceptual model of software security knowledge. 34

Figure 21. Visualized software security knowledge base. 34

Figure 22. Top page of Software Security Knowledge Base system 35

Figure 23. Threat Type Screen 36

Figure 24. Threat Type registration form 36

Figure 25. Registered Identify Spoofing in Attack Pattern knowledge 37

Figure 26. Spoofing information in Threat Type Screen 38

Figure 27. Edit form in Threat Type screen 38

Figure 28. Association of Treat Type knowledge “Spoofing” and Attack pattern knowledge “Identity Spoofing” 39

Figure 29. Example use and misuse cases. 41

Glossary

UC	Use Case
CCDB	Crypto Companion Database
WP	Work Package
DB	Database
GDPR	General Data Protection Regulation
M(number)	Month (number of the month in the project)
API	Application Programming Interface

1. Introduction

1.1 Scope of the document

This deliverable is the first version of a software demonstrator, which in this first version consists of two different demonstrators as the main outcome of Task 4.4: Application level security. The software demonstrators are accompanied by this document, which essentially provides for each prototype a description with the list of its components or modules, the installation instructions of the tools and software needed to run the demonstrator and a user manual. The next iteration of this deliverable, the second version, will be delivered in month M30.

The two demonstrators provided in this deliverable contributes to the multi-layered security in different parts.

With the Crypto Companion Database we provide security for sensitive data that cannot be stored on Blockchain. The two main goals of this demonstrator are to encrypt and save the data and to secure it. In order to fulfill these requirements a secured API is provided. This API is also connected to a developed module called Crypto Module that allows the encryption and decryption of data per user.

With the Security analysis tool we provide security requirements that only the use case diagram cannot elicit. The main goal of this demonstrator is to create a misuse case diagram that enables the association of security knowledge and elicit threats and security requirements of the use case.

1.2 Relationship to other work packages and tasks

The deliverable D4.7 regarding Application level security uses as input the need for securing data detected in the definition of the use cases in WP2. It also takes into account the architecture defined in the deliverable 3.3 and the functional and technical specifications from Deliverable 3.4.
It is also related to the other WP4 tasks: Task 4.1 (IoT Security), Task 4.2 (Cloud and data level security), Task 4.3 (P2P level security and blockchains) and Task 4.5 (End-to-end security).

1.3 Methodology followed

Data security and protection is one of the main goals of any application, and one of the main goals of the M-Sec project.

In order to fulfill GDPR, a parallel system to Blockchain has been developed. It is known as the use of the named “Companion Database”, which allows saving data linked to Blockchain with a hash of the same. This solution has been adopted, but with some additions. The data stored in the database will be encrypted per user, meaning that they will not share keys between users. The problem of a database usually is that it is centralized, so with this development the data can be distributed.

So, the main motivations of the development were:

- A user can save data.
- A user can delete data.
- A user can read data.
- A user can modify data.
- The data will be encrypted when saved.
- The data will be decrypted when read. (Conditional with the next statement)
- The data will only be accessible by the owner or an authorized one.
- The data will be distributed.

On the other side, the need for secure application development has increased with the growing number of services on the Internet. It is required for secure application development to consider various types of security concerns. But software engineers are generally not security experts. They don't know what kind of techniques are available for secure application development and how they should be utilized.

"Security Analysis Tool" provides them the function to elicit security threats and requirements based on Software Security Knowledge Base.

Below is a figure showing the position of the Crypto Companion Database and Security Analysis Tool among the other components:

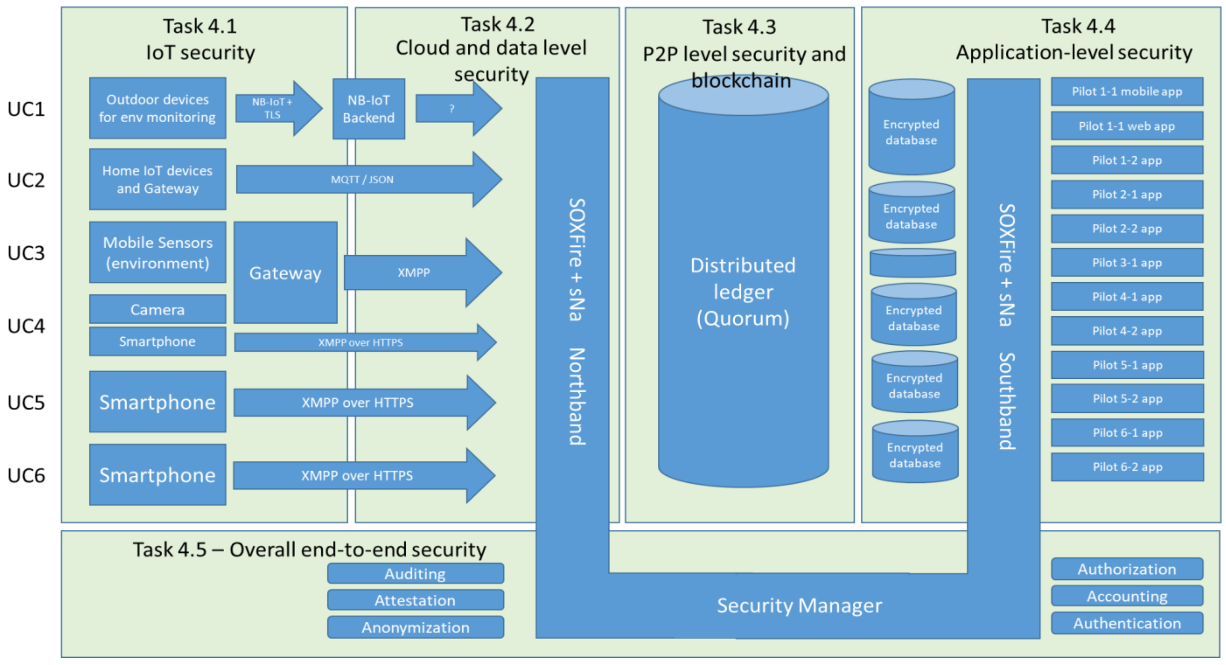


Figure 1. Overall M-Sec topology

2. Demonstration 1 – Crypto Companion Database (CCDB)

2.1 General Description of the Prototype

Since it is necessary that sensitive data stored has to be secured and private, the crypto companion database (CCDB) is proposed as a parallel system to the blockchain for the encrypted storage. The blockchain will save a hash created from the sensitive data, and the CCDB will store the sensitive data encrypted together with the hash. The hash will be used to have a connection between the transaction in the blockchain and the data stored in the database.

The CCDB will encrypt the data with an asymmetric public/private key pair.

This data could only be accessed by the owner, which will have to be authenticated, and the authorized operators allowed by the owner. The authorization is not a part of the CCBD as it will be carried by the Blockchain itself, so the component will ask the Blockchain for it.

With this database insertion, deletion and consultation of the information will be possible. The modification process will be a bit more complex, as the hash that holds the link between the blockchain and the database will change if the information changes. So if a modification is needed, it will be done by deleting the old information and inserting the new one.

Each application in the ecosystem can have its own crypto companion database; therefore data will always be distributed. In order to make it accessible and replicated if wanted, the key pair can be replicated on any system by providing the 24-word mnemonic. To reduce the amount of data held by a single database, the location of specific information can be stated in the blockchain transaction.

The following figure shows how data can be accessed.

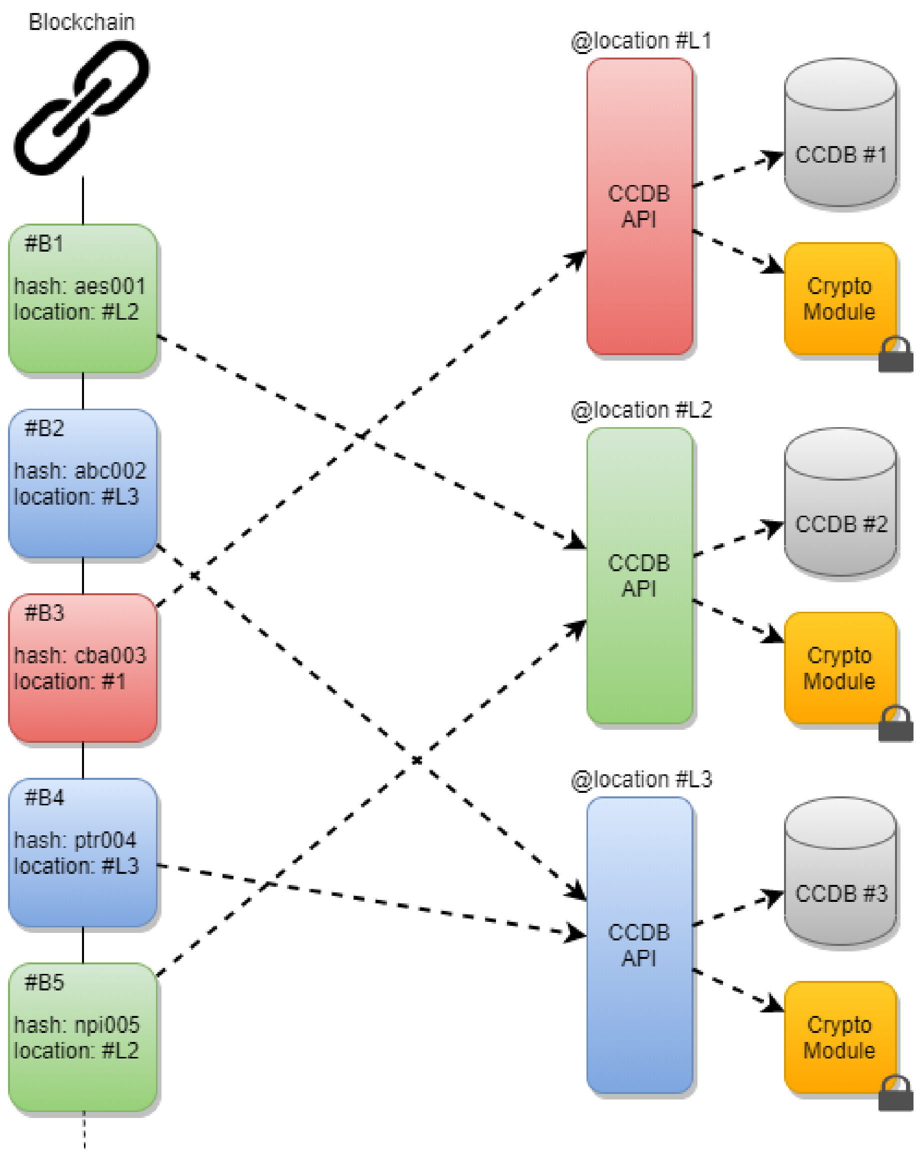


Figure 2. Access to distributed data.

Components

The components used to create the CCDB are the followings:

- Crypto Module
- Companion DB Module

The evolution of the Companion DB Module with the Crypto Module makes a secured database, as the data will be encrypted by an asymmetric key pair, so it will be called **Crypto Companion DB**.

The Crypto Module will be used independently on any type of database (currently only supports MongoDB) and in any software because it provides an API to encrypt/decrypt data. The API of this module is designed as a private API with no access to the Internet, so it does not provide any security.

The Companion DB Module has a public API that can be used to save, delete and query data. It also provides an authentication layer in order to secure the users that access the data. This module also provides an authorization layer in order to know if the owner of the data allows an operator or external user to see it. This authorization layer will make use of the Smart Contracts on Blockchain described in **Deliverable 4.5 section 2. Demonstration 1: Blockchain Framework and Middleware Services.**

The following diagram shows an overview of the components.

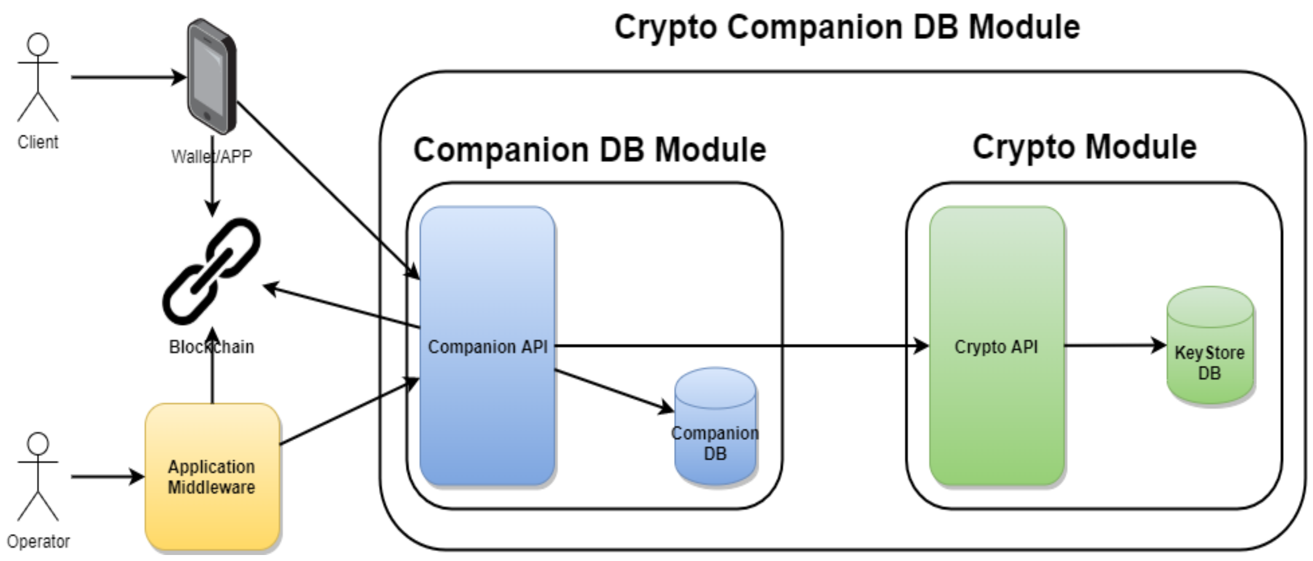


Figure 3. Crypto Companion Database Module components.

Crypto Module

This module allows a user to encrypt and decrypt data. It has two components:

- The Crypto API, that is in charge of encryption and decryption of the data with the keys stored in the KeyStore DB.
- The KeyStore DB, that is a MongoDB that holds the key pairs to encrypt and decrypt data by the users.

The Crypto API provides the following methods.

- A method to create an asymmetric key pair:

```
POST /crypto/enrol/{hash}
```

The creation of a public/private key pair can be made by scratch or by providing a 24-word mnemonic, allowing replicating the keys in other applications. It will be useful if the user wants to authorize always with the same public/private key, and also will allow a distributed system to be able to decrypt data in a distributed way.

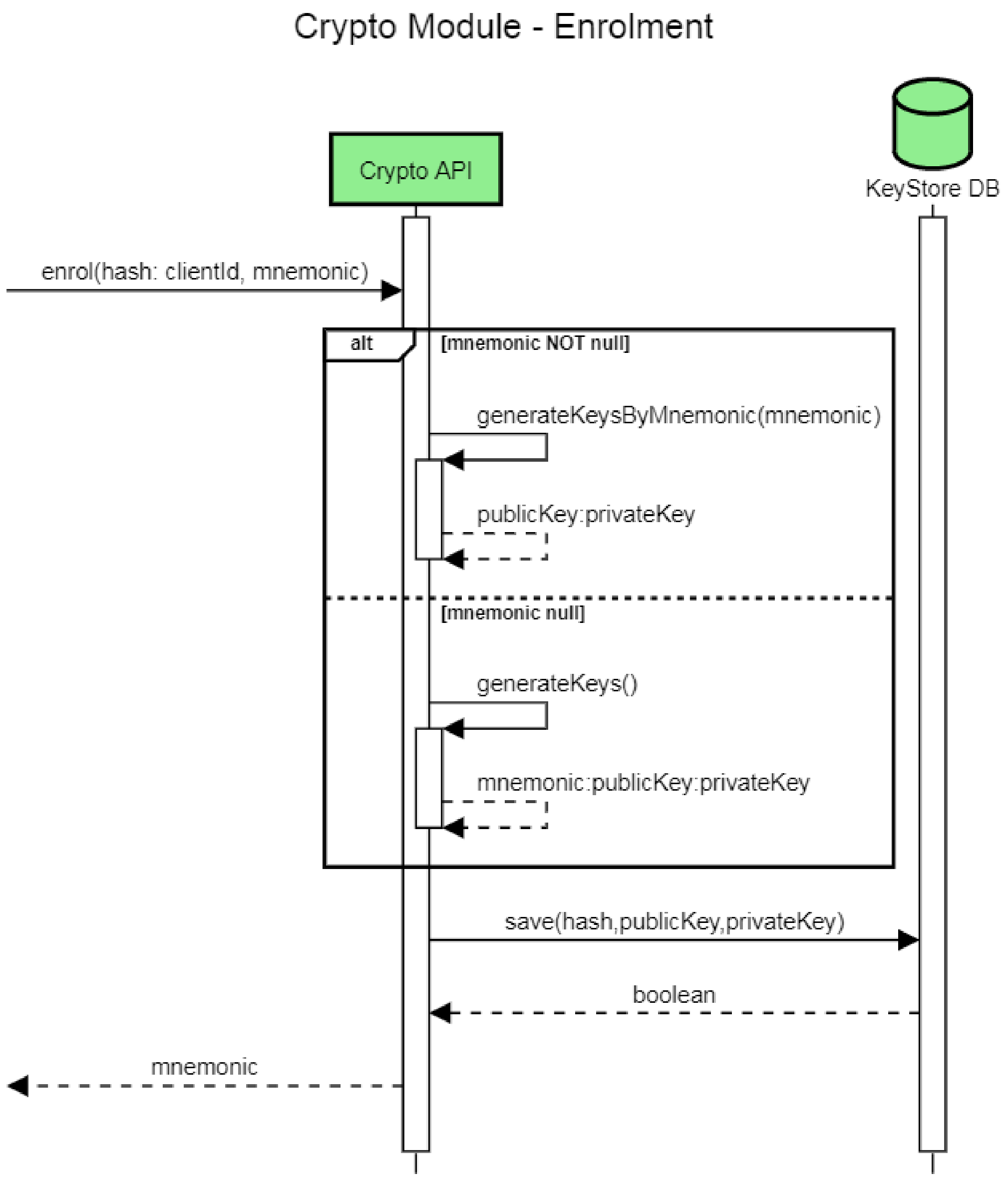
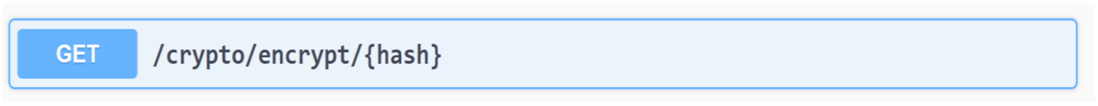

Crypto Module - Enrolment
Crypto API
KeyStore DB
enrol(hash: clientId, mnemonic)
alt
[mnemonic NOT null]
generateKeysByMnemonic(mnemonic)
publicKey:privateKey
[mnemonic null]
generateKeys()
mnemonic:publicKey:privateKey
save(hash,publicKey,privateKey)
boolean
mnemonic

Figure 4. Sequence diagram. Enrolment in Crypto Module.

- A method to encrypt data:



GET /crypto/encrypt/{hash}

This endpoint will take the private key of the user with the hash provided, and encrypt the string with the data in the payload.

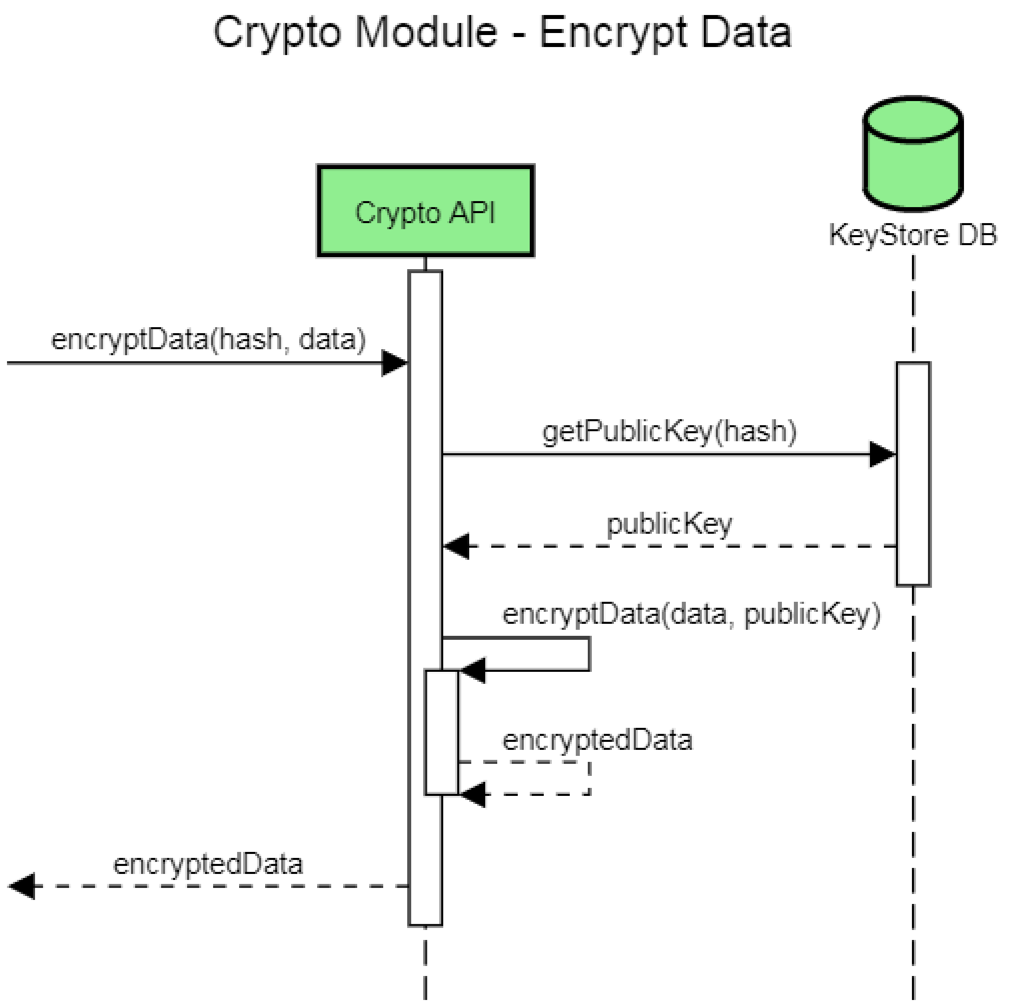


Figure 5. Sequence diagram. Data encryption in Crypto Module.

- A method to decrypt data:

```
GET /crypto/decrypt/{hash}
```

This endpoint will take the private key of the user with the hash provided, and decrypt the string with the data in the payload.

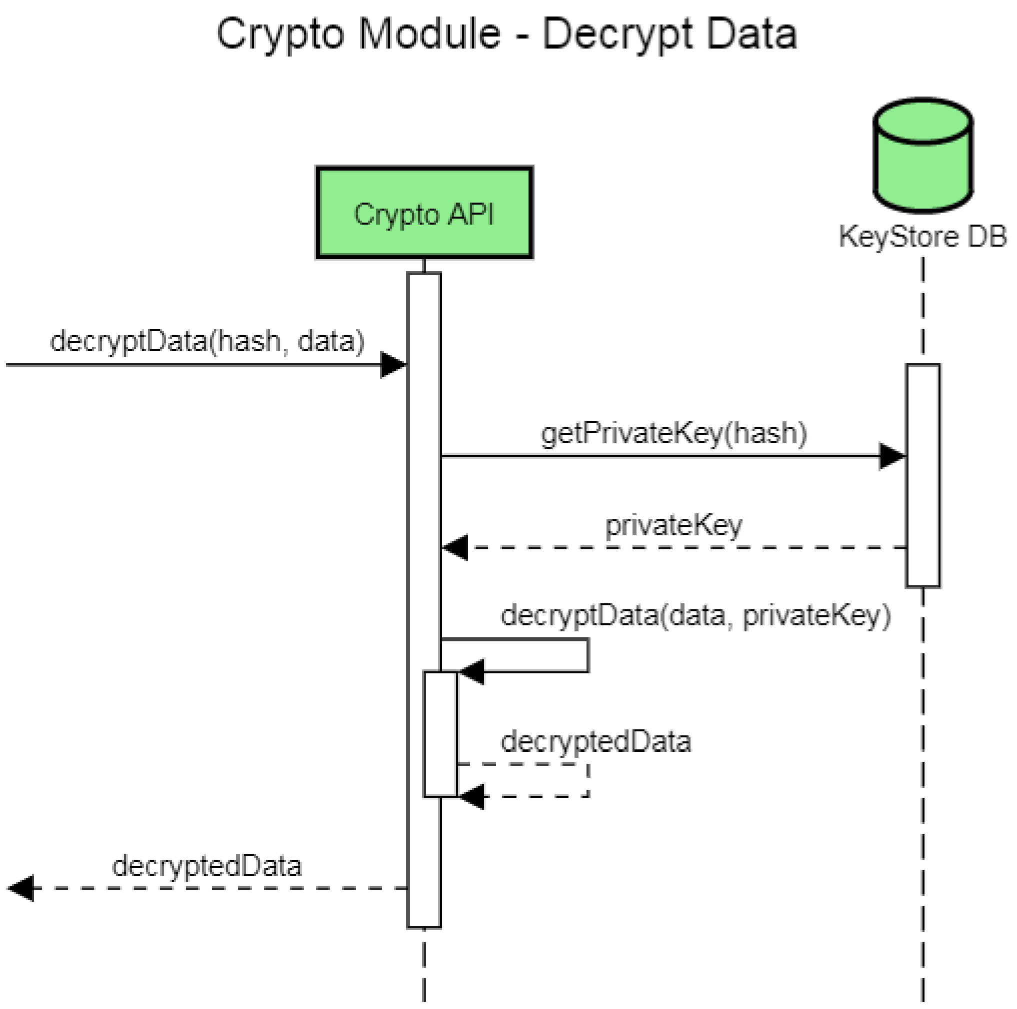
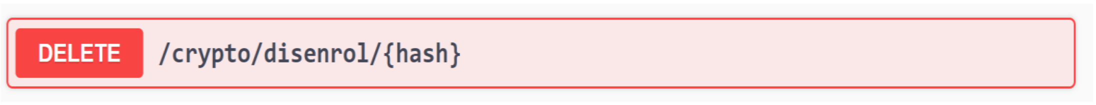


Figure 6. Sequence diagram. Data decryption in Crypto Module.

- A method to delete the keys:



DELETE /crypto/disenrol/{hash}

This endpoint will delete the public/private keys associated with the hash provided.

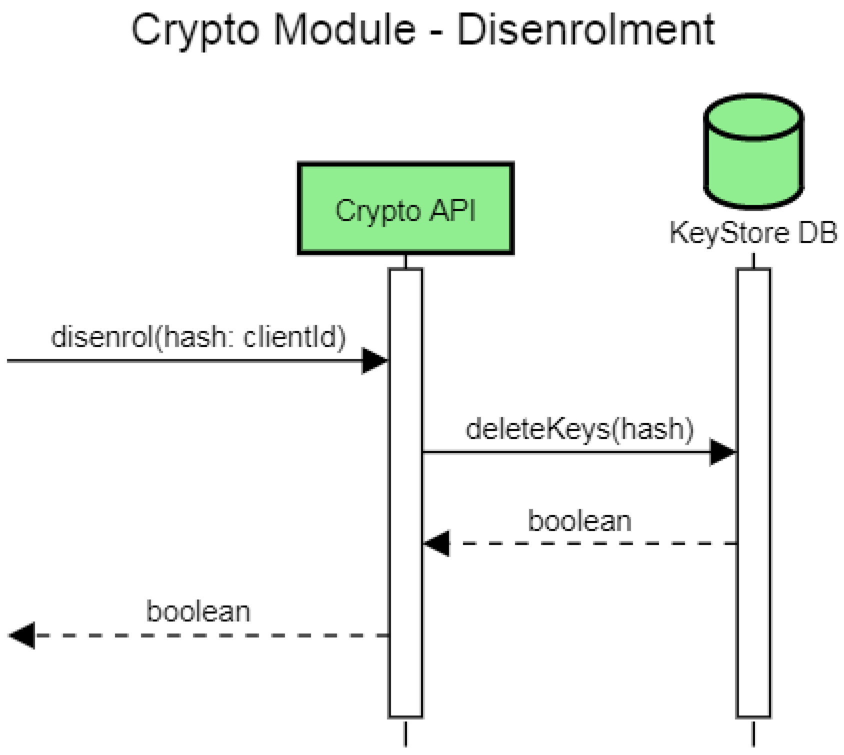


Figure 7. Sequence diagram. Disenrolment in Crypto Module.

The API of this module is intended to be private, so it does not provide any kind of security.

The KeyStore DB will store the public and private keys created by the Crypto API with a hash that will act as an identifier.

So the data stored in the database will look like:

- hash: 32-64 hexadecimal string.
- Public key: The Public key generated by an asymmetric key algorithm that matches the private key.
- Private key: The Private key generated by an asymmetric key algorithm that matches the public key.

Crypto Companion DB Module

This module allows a user to have an authentication system and save the encrypted data. It also provides other users with the possibility to read data from a user if authorized.

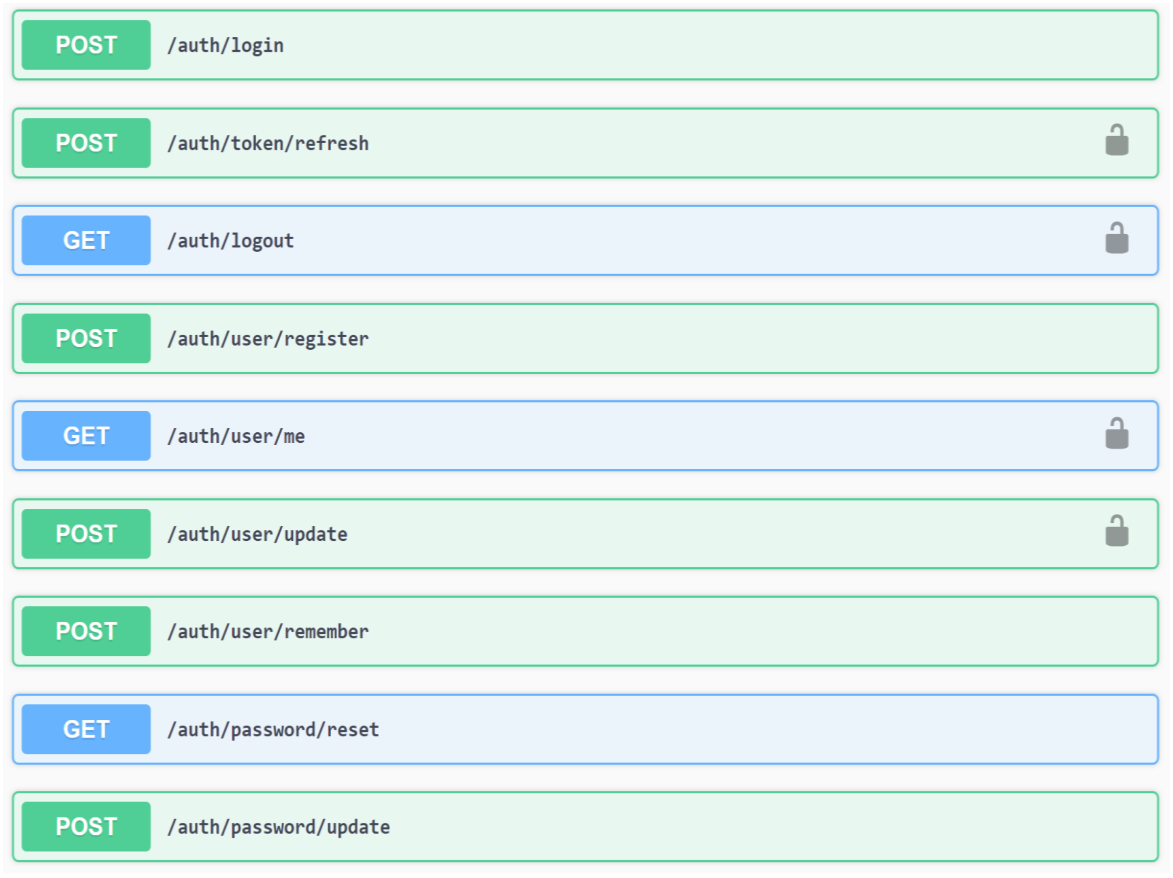
This module has two components:

- The Companion API, is in charge of authentication and managing all the data providing methods to save, read and delete data in the Companion DB.
- The Companion DB, is a MongoDB that stores the encrypted data.

The Companion DB API provides the following methods.

Authentication API.

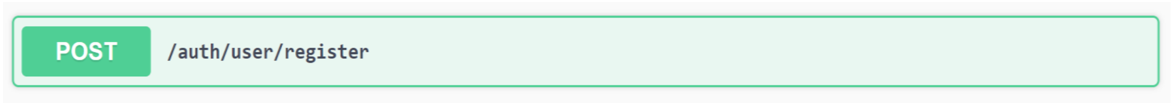
- A set of methods to register, update user information and recover a password.



POST /auth/login
POST /auth/token/refresh
GET /auth/logout
POST /auth/user/register
GET /auth/user/me
POST /auth/user/update
POST /auth/user/remember
GET /auth/password/reset
POST /auth/password/update

Figure 8. Authentication API in Crypto Companion Database Module.

- A method to register:

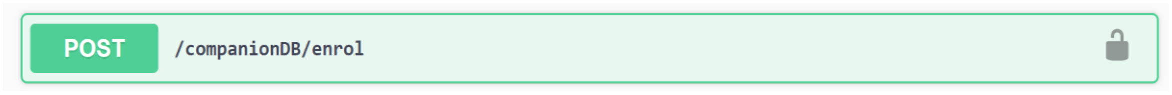


POST /auth/user/register

The registration of a user will also trigger the enrolment on the Crypto Module, so the keys will be created during the registration.

Data Management API.

- A method to enroll:



POST /companionDB/enrol

The creation of the public/private key pair can be made by scratch or by providing a 24-word mnemonic, allowing replicating the keys in other applications. It will be useful if the user wants to authorize always with the same public/private key, and also will allow a distributed system to be able to decrypt data in a distributed way.

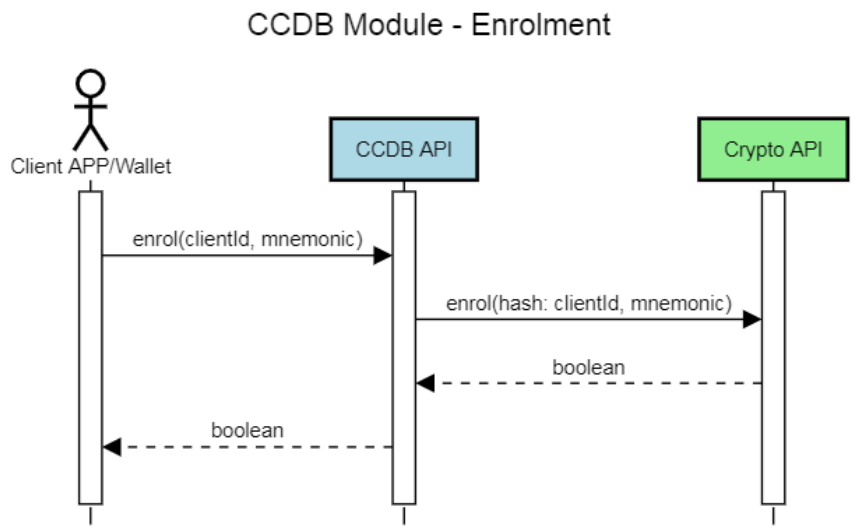
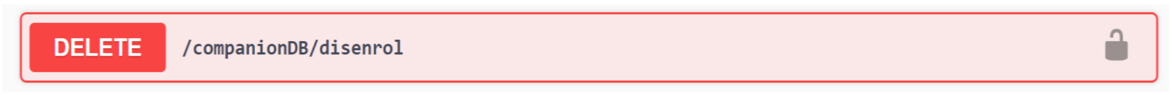


Figure 9. Sequence diagram. Enrolment in CCDB Module.

- A method to disenrol:



DELETE /companionDB/disenrol

This endpoint will delete all data associated with the user along with its public/private keys.

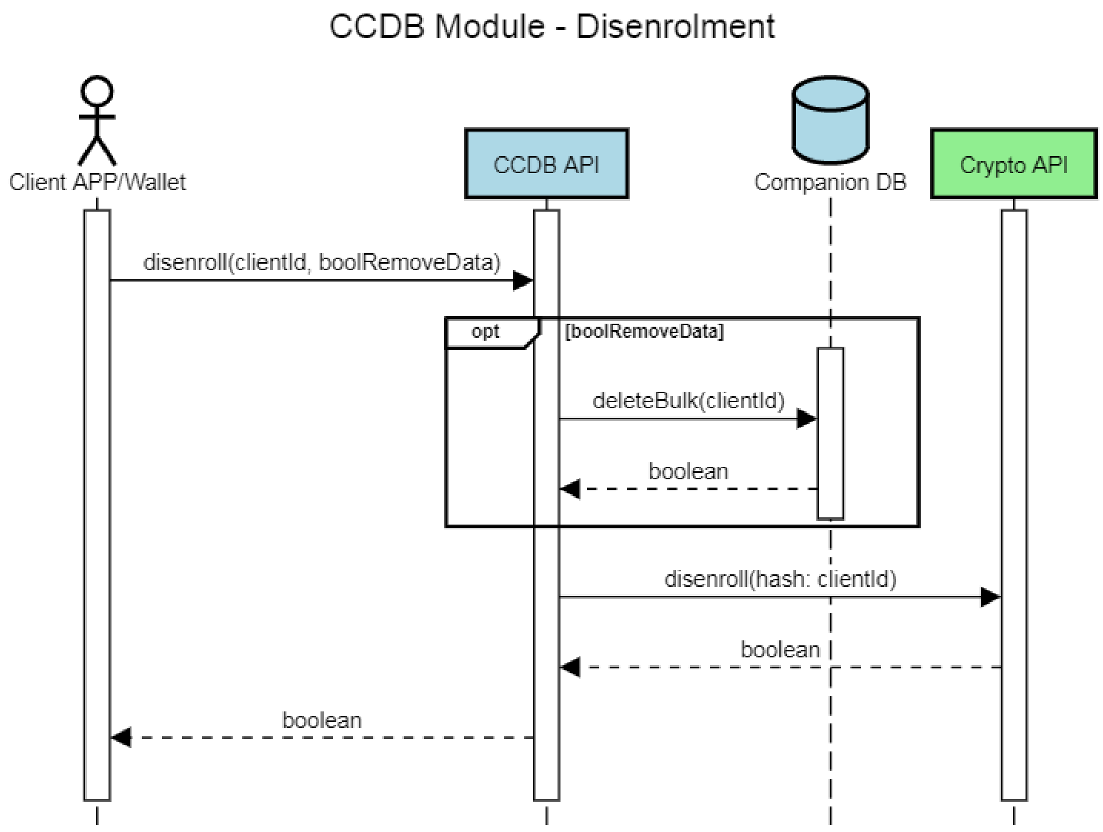
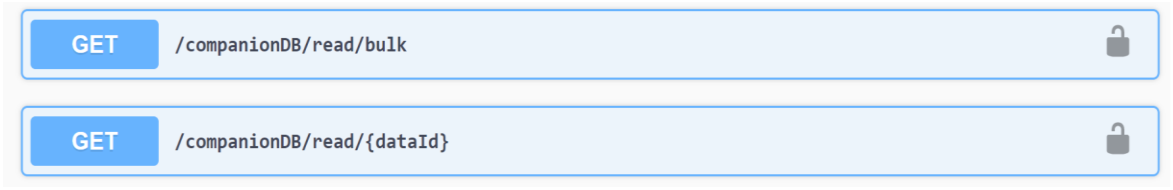


Figure 10. Sequence diagram. Disenrolment in CCDB Module.

- A method to read data:



```
/companionDB/read/bulk
```

```
/companionDB/read/{dataId}
```

This endpoints will let an owner or an authorized user to read the encrypted data.

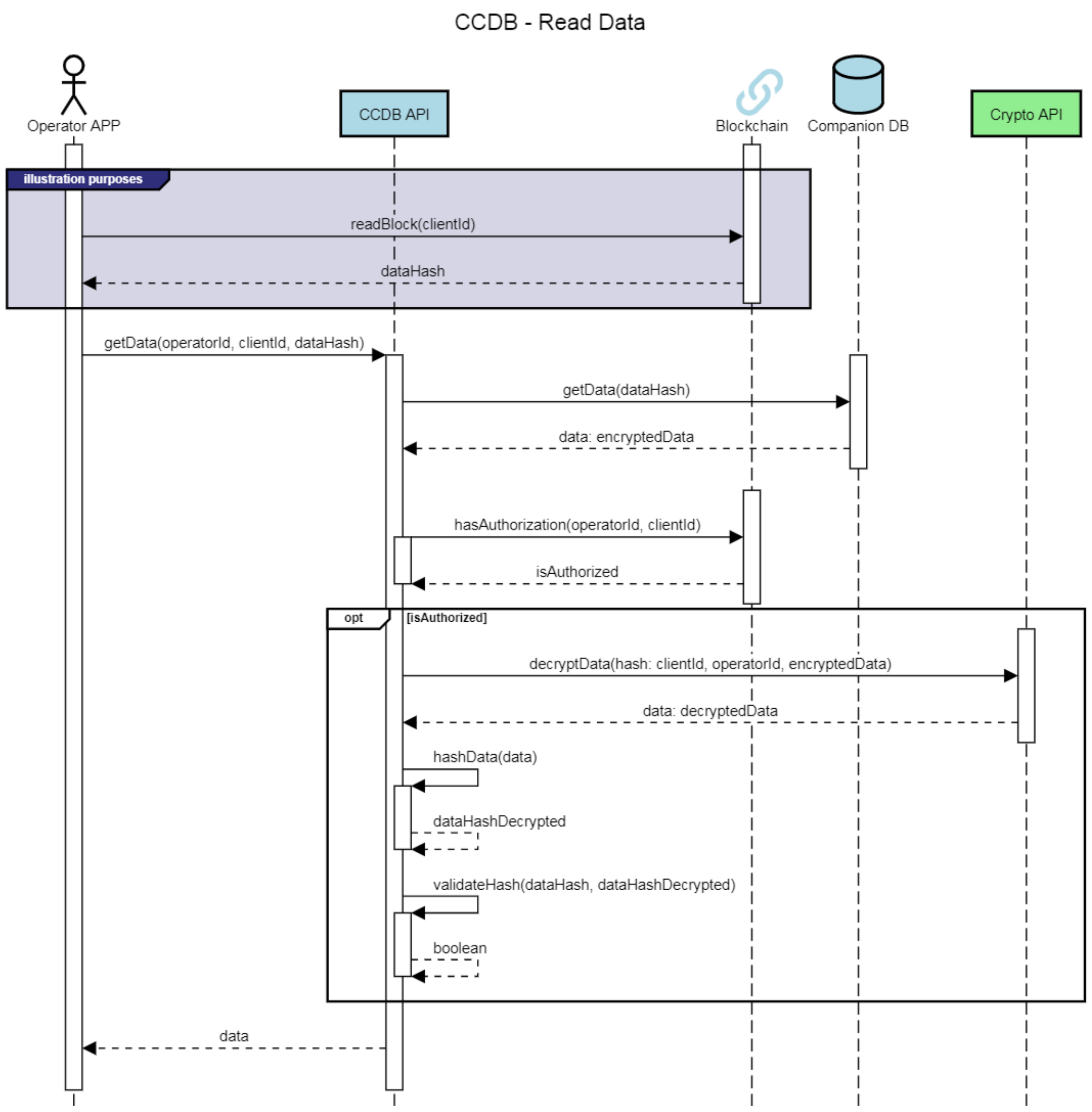
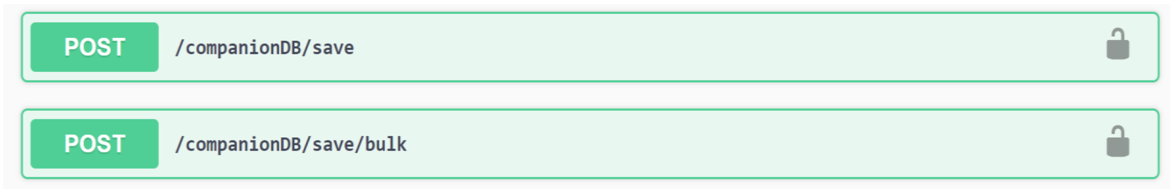


Figure 11. Sequence diagram. Read data in CCDB Module.

- A method to save data:



POST /companionDB/save

POST /companionDB/save/bulk

This endpoint will let an owner to save encrypted data.

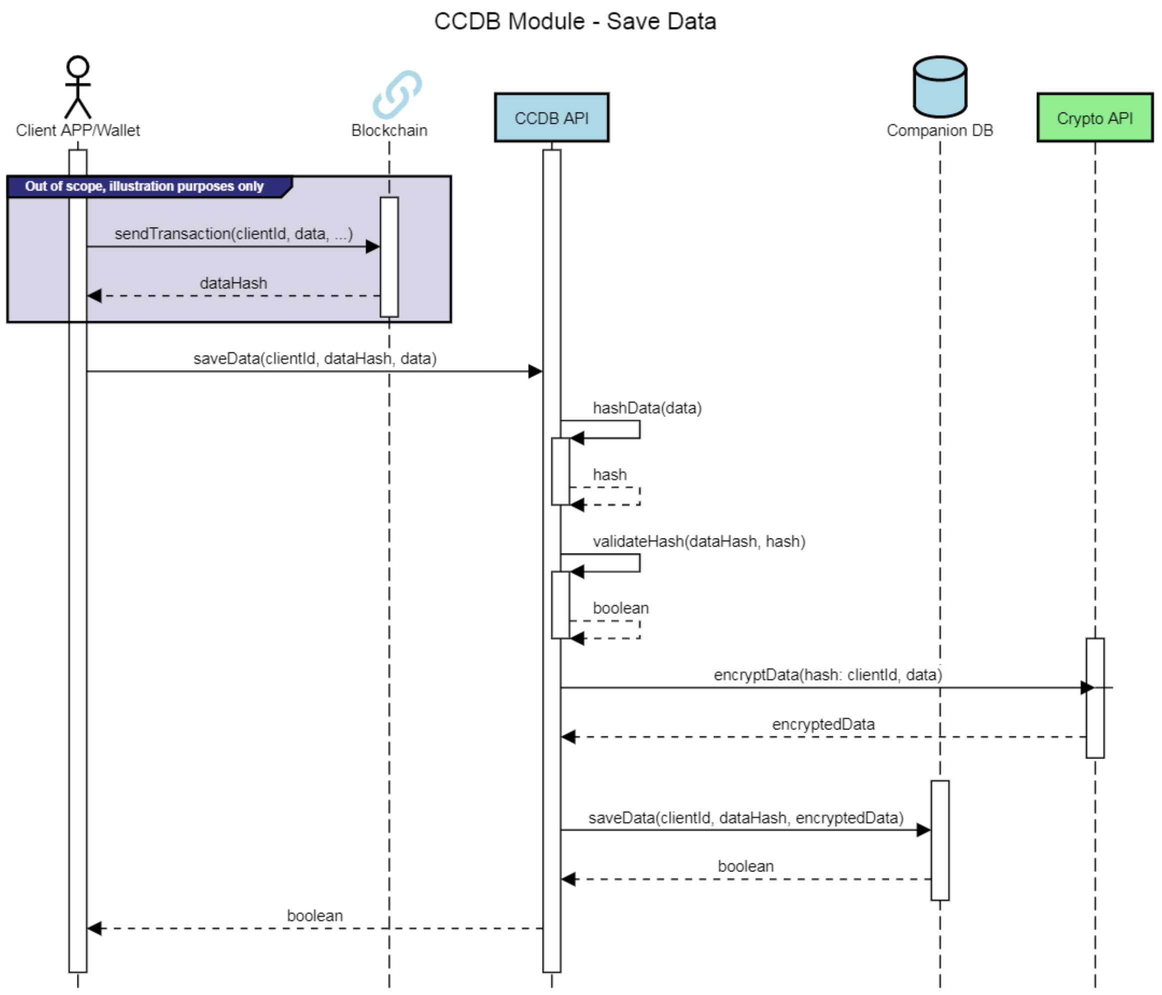
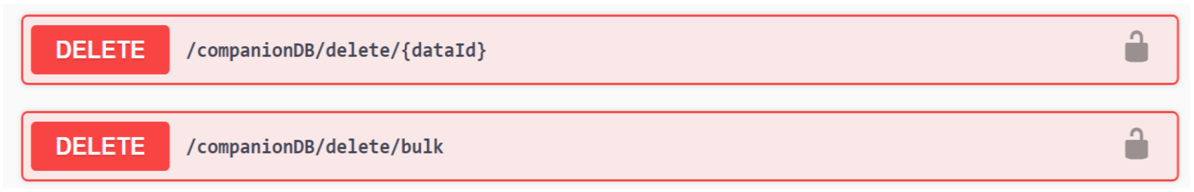

CCDB Module - Save Data
Client APP/Wallet
Blockchain
CCDB API
Companion DB
Crypto API
Out of scope, illustration purposes only
sendTransaction(clientId, data, ...)
dataHash
saveData(clientId, dataHash, data)
hashData(data)
hash
validateHash(dataHash, hash)
boolean
encryptData(hash: clientId, data)
encryptedData
saveData(clientId, dataHash, encryptedData)
boolean
boolean

Figure 12. Sequence Diagram. Save data in CCDB Module.

- A method to delete data:



DELETE `/companionDB/delete/{dataId}`

DELETE `/companionDB/delete/bulk`

This endpoint will let the owner of the data to delete it.

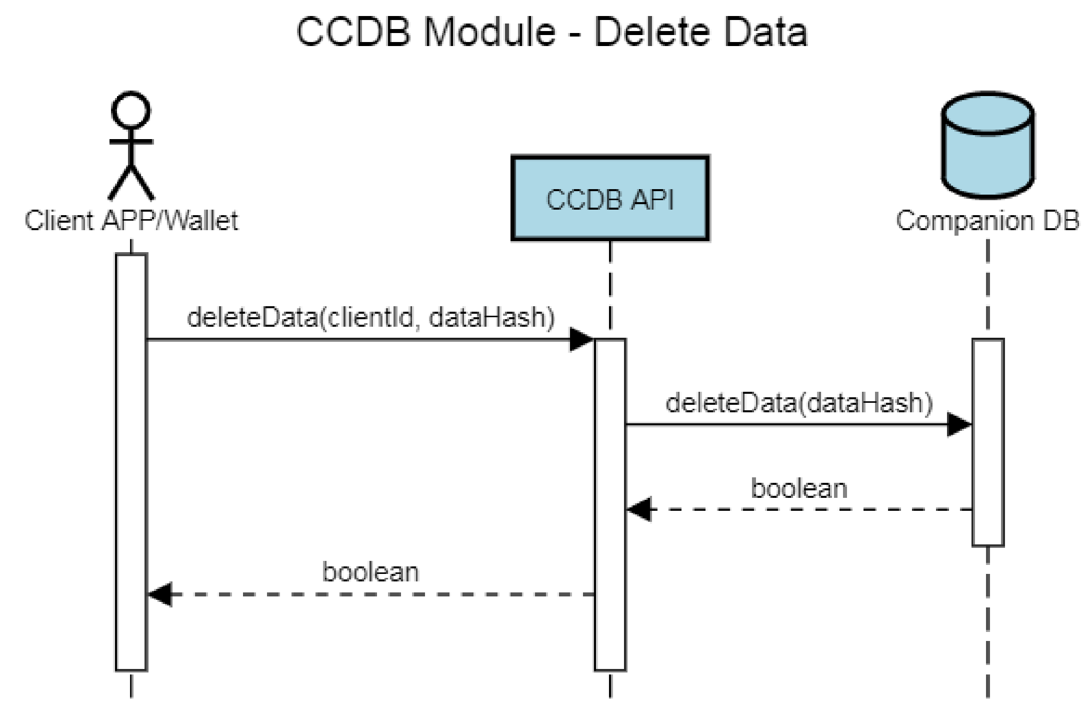
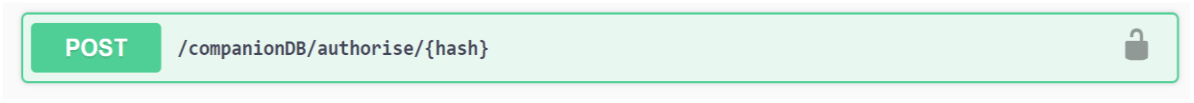


Figure 13. Sequence Diagram. Delete data in CCDB Module.

- A method to authorize a user:



```
/companionDB/authorise/{hash}
```

This endpoint will let an owner to authorize another user to decrypt its data.

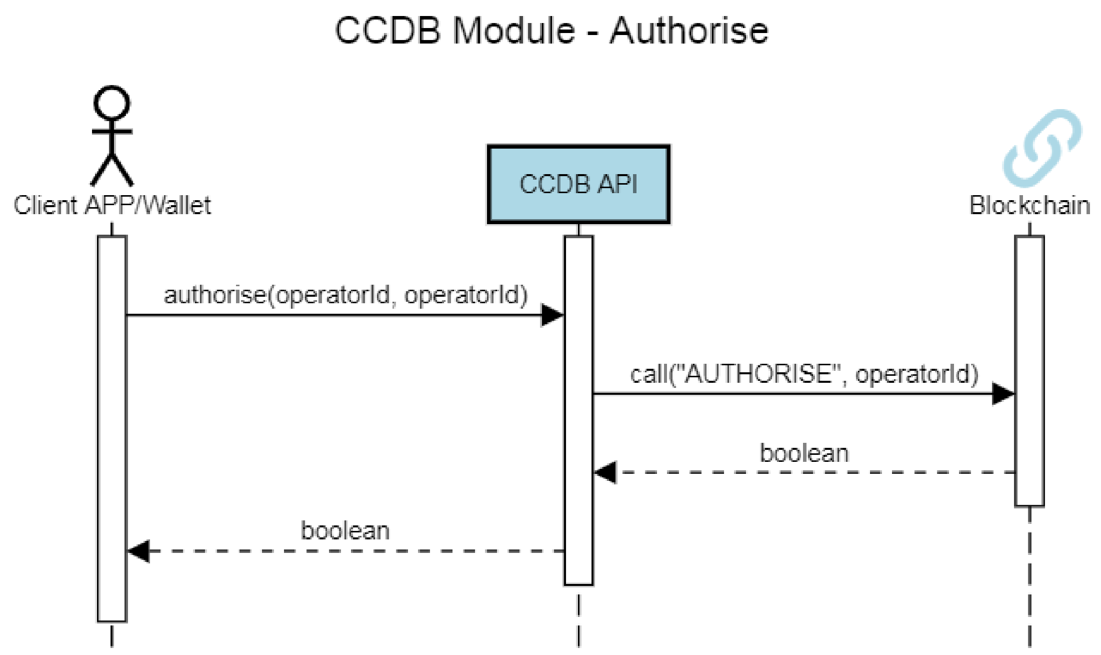
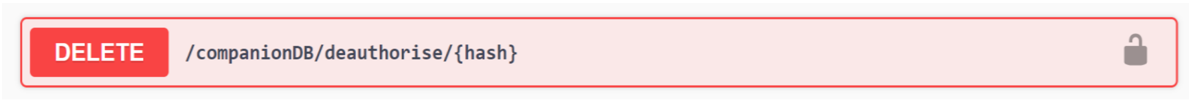


Figure 14. Sequence diagram. Authorize in CCDB Module.

- A method to deauthorize a user:



DELETE /companionDB/deauthorise/{hash}

This endpoint will let an owner to deauthorize another user to decrypt its data.

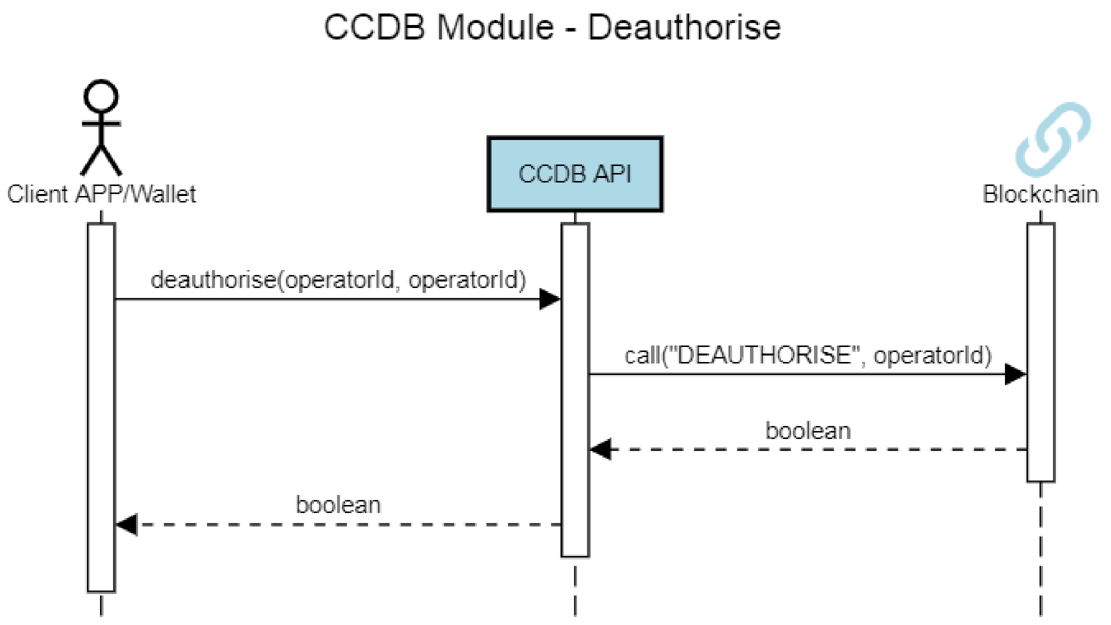
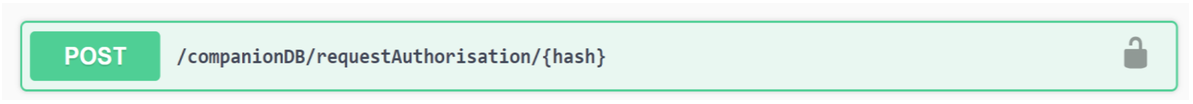

CCDB Module - Deauthorise
Client APP/Wallet
CCDB API
Blockchain
deauthorise(operatorId, operatorId)
call("DEAUTHORISE", operatorId)
boolean
boolean

Figure 15. Sequence diagram. Deauthorise in CCDB Module.

- A method to request authorization to a user:



POST /companionDB/requestAuthorisation/{hash}

This endpoint will let an external user to request authorization to access data to the owner.

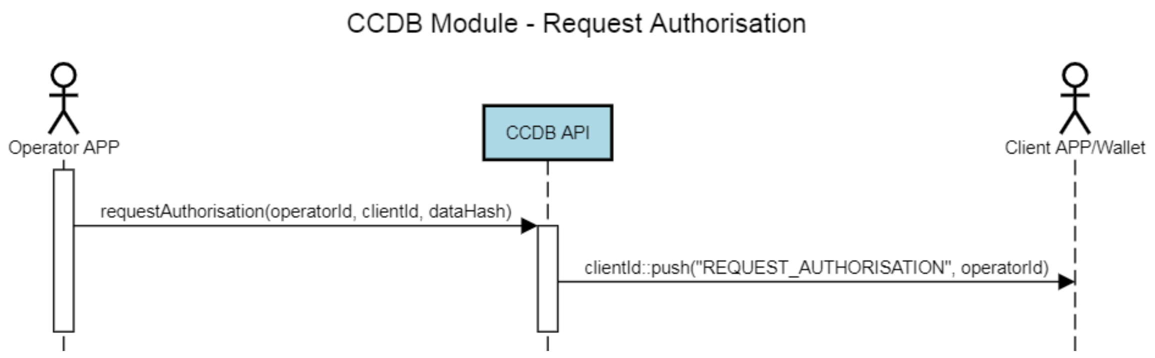

CCDB Module - Request Authorisation
Operator APP
CCDB API
Client APP/Wallet
requestAuthorisation(operatorId, clientId, dataHash)
clientId::push("REQUEST_AUTHORISATION", operatorId)

Figure 16. Sequence diagram. Request authorization in CCDB Module.

2.2 Package Information & Installation Instructions

The following section provides a list of the tools and dependencies needed for the correct operation of the crypto companion database. It also provides a set of steps to install them as well as the demonstration itself. All installation processes have been taken from the official websites of the products and modified according to the needs of the demonstrator.

Required Tools and dependencies

The Following is the list of the required tools and dependencies of the modules:

- Docker (it comes with, Kubernetes, Kitematic, Docker Manager, …)
- Docker Quickstart Terminal
- Docker Toolbox (for Windows Users only)
- Mongo DB
- Oracle VM Virtualbox
- Nodejs v10.17.0
- NPM 6.11.3
- Git

The version indicated in some tools/dependencies are important for compatibility. If the versions are not these, it might raise a problem.

In order to ease the installation, proceed with the established order in the list.

Install Docker

The installation can be found in the Docker's webpage https://docs.docker.com/v17.09/engine/installation/, but in the following there is a list of the main steps and commands for Windows and Ubuntu Linux.

Windows 10: (Source: https://docs.docker.com/v17.09/docker-for-windows/install/#start-docker-for-windows)

In order to install Docker you are requested to follow the next steps:

1. Download docker from Docker Hub:
   https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe
2. Double-click Docker for Windows Installer.exe to run the installer.
3. Follow the instructions on the installation wizard to accept the license, authorize the installer, and proceed with the install.
   When prompted, authorize the Docker Desktop Installer with your system password during the install process. Privileged access is needed to install networking components, links to the Docker apps, and manage the Hyper-V VMs.
4. Click Finish on the setup complete dialog and launch the Docker Desktop application.

5. Docker will not start automatically. To start it, search for Docker, select the app in the search results, and click it (or hit Return).

Ubuntu Xenial 16.04 LTS: (Source: https://docs.docker.com/v17.09/engine/installation/linux/docker-ce/ubuntu/)

A. Set up the repository

1. Update the `apt` package index:

```
sudo apt-get update
```

2. Install packages to allow apt to use a repository over HTTPS:

```
sudo apt-get install apt-transport-https ca-certificates curl software-
properties-common
```

3. Add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-
```

   Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

```
sudo apt-key fingerprint 0EBFCD88

pub   4096R/0EBFCD88 2017-02-22
      Key fingerprint = 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid                  Docker Release (CE deb) <docker@docker.com>
sub   4096R/F273FCD8 2017-02-22
```

4. Use the following command to set up the stable repository.

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

B. Install Docker

1. Install the ***linux-image-extra*** kernel package:

```
sudo apt-get update -y && sudo apt-get install -y linux-image-extra-
$(uname -r)
```

2. Install Docker:

```
sudo apt-get install docker-engine -y
```

3. Start Docker:

```
ludo service docker start
```

4. Verify Docker:

```
sudo docker run hello-world
```

Install Mongo DB

As docker has been installed in the section above, the installation of the Mongo DB will be as easy as executing the following command for any operating system:

```
docker run -p 27017:27017 --name mongo-nest -d mongo:4
```

Install Node.js and npm

The installation of these two tools is done together, and it can be found in the Node.js webpage https://nodejs.org/en/ , but in the following there are lists of the main steps and commands for Windows and Ubuntu, respectively.

Windows 10:

1. Download the binary from: https://nodejs.org/dist/v10.17.0/
2. Install the *msi* or *exe* file by a double-click.
3. Follow the instructions.
4. Check that Node.js is installed with the command:

```
node -v
```

5. Check that npm is installed with the command:

```
npm -v
```

Ubuntu Xenial 16.04 LTS: (check https://github.com/nodesource/distributions/blob/master/README.md#debinstall for further information)

1. Add the NodeSource package signing key:

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash -
```

2. Install Node.js

```
sudo apt-get install -y nodejs
```

Install Git

Extracted from https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Windows 10:

1. Download and install it from https://git-scm.com/download/win

Ubuntu Xenial 16.04 LTS:

1. Execute the following command :

```
sudo apt install git-all
```

Download and Run the Demonstrator

In order to download and run the demonstrator, the following steps have to be performed:

1. Clone the GitHub project in a selected folder:
   ```
   git clone https://github.com/jordiescudero/wl-bc-cs/
   ```
2. Execute the command from the installation of Mongo DB:
   ```
   docker start mongo-nest
   ```
3. Go to the root of the project:
   ```
   npm run start
   ```
4. The base URI for all the interface will be: http://localhost:3000/api/
5. The Swagger UI can be found at: http://localhost:3000/api/docs/#/

User Manual

As stated in the section "Companion DB Module" the APIs that will be published and used will be the following.

- Authentication API.

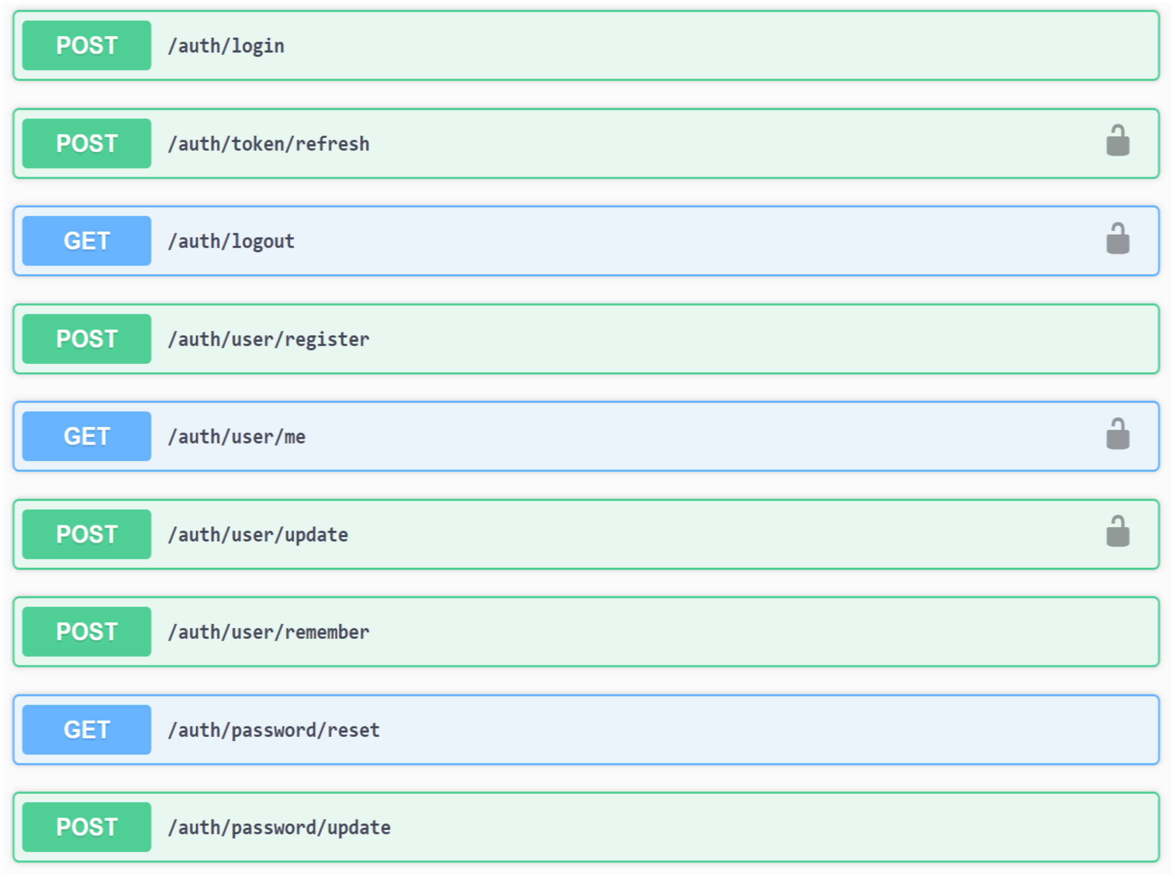

POST /auth/login
POST /auth/token/refresh
GET /auth/logout
POST /auth/user/register
GET /auth/user/me
POST /auth/user/update
POST /auth/user/remember
GET /auth/password/reset
POST /auth/password/update

Figure 17. Authentication API in Crypto Companion Database Module.

- Data Management API.

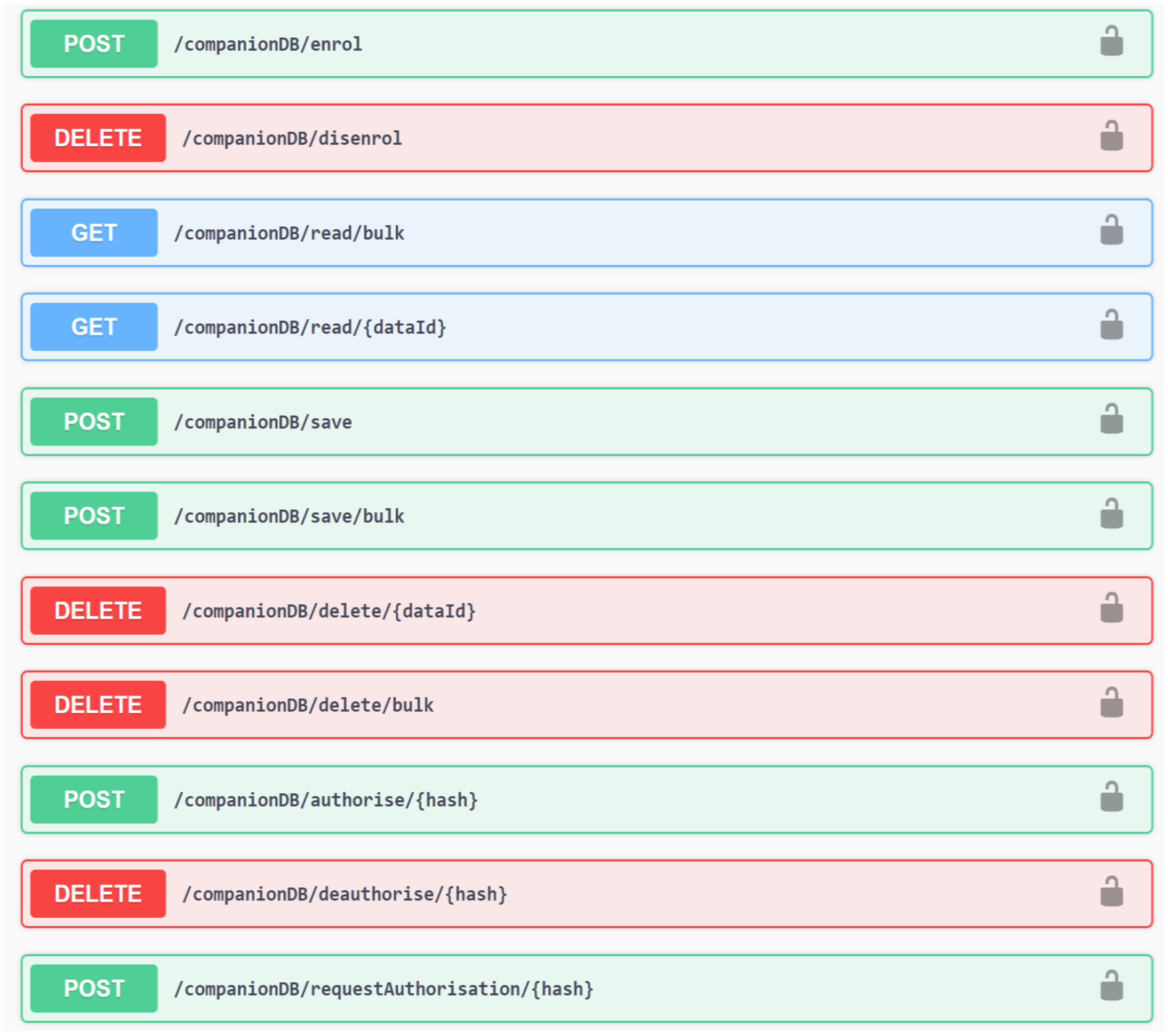

POST /companionDB/enrol
DELETE /companionDB/disenrol
GET /companionDB/read/bulk
GET /companionDB/read/{dataId}
POST /companionDB/save
POST /companionDB/save/bulk
DELETE /companionDB/delete/{dataId}
DELETE /companionDB/delete/bulk
POST /companionDB/authorise/{hash}
DELETE /companionDB/deauthorise/{hash}
POST /companionDB/requestAuthorisation/{hash}

Figure 18. Management API in Crypto Companion Database Module.

The Swagger UI provides enough information to let the developer know how to use this API, but some examples were put together as a starting point.

- **GET /*companionDB*/read/{dataId}**

```
curl -X GET
"http://localhost:3000/api/companionDB/read/hashhashhashhashhash" -H
"accept: application/json"
```

- **POST /*companionDB*/save**

```
curl -X POST "http://localhost:3000/api/companionDB/save" -H "accept:
application/json" -H "Content-Type: application/json" -d "{ \"name\":
\"Name\", \"email\": \"email@email.com\", \"birht_date\": \"01/01/2001\",
\"gender\": \"Other\", \"city\": \"Barcelona\"}"
```

The json beautified:

```
{
  "name": "Name",
  "email": "email@email.com",
  "birht_date": "01/01/2001",
  "gender": "Other",
  "city": "Barcelona"
}
```

- **DELETE /*companionDB*/delete/{dataId}**

```
curl -X DELETE "http://localhost:3000/api/companionDB/delete/hashhashhash"
-H "accept: application/json"
```

- **POST /*companionDB*/authorise/{hash}**

```
curl -X POST
"http://localhost:3000/api/companionDB/authorise/hashhashhash" -H "accept:
application/json" -H "Content-Type: application/json" -d "{ \"authHash\":
\"authorisedHash\"}"
```

The json beautified:

```
{
  "authHash": "authorisedHash"
}
```

Licensing

Licensing for all the components/software used:

- Docker is under Apache License 2.0 (https://www.apache.org/licenses/LICENSE-2.0) form more detail go to https://www.docker.com/legal/components-licenses.
- Mongo DB is under Server Side Public License (https://www.mongodb.com/licensing/server-side-public-license)
- NPM is under Artistic License 2.0 (https://www.npmjs.com/policies/npm-license)
- Git is under GNU General Public License version 2.0 (https://opensource.org/licenses/GPL-2.0)

- Oracle VirtualBox is under GNU General Public License, version 2 (https://www.gnu.org/licenses/old-licenses/gpl-2.0.html)
- Software developed is under MIT (https://github.com/jordiescudero/wl-bc-cs/blob/master/LICENSE)

Reference Use Case

Any Use Case can make use of the Crypto Companion DB as they can interact with the Blockchain and thus try to save or query sensitive data.

Summary

To sum up, in this demonstrator it is shown that the Crypto Companion DB can save, delete and query sensitive data if and only if you are successfully authenticated and authorized.

Any application can make use of this component securing its data by a user.

The possibility to encrypt and decrypt data by user, makes the database very secure. If some malicious intruder steal the database it would be impossible to decrypt it, and if he or she could, only a few rows will be compromised.

Next Steps

The next steps during next year, will be to enhance the functionalities by adding new endpoints, to make the existing and the new easier and more friendly to use.

The installation is also a part to improve, as there are several steps to install and start the component, so it will be “*dockerized*” in order to make it easier.

At the moment, a best practice has been stated in order to have distributed data by adding the link to the Companion Database together with the hash of the data. In the future will be analyzed the creation of a Smart Contract to help find the distributed Companion databases of the different applications, so the access to them will be easier, and in the future might be used to synchronize data if needed.

The aim of the companion database is to provide security and privacy, not to force a user to use a specific brand of the database. Because of that, different connectors to different types of database can be developed and integrated into the development.

3. Demonstration 2 – Security-analysis-tool

3.1 General Description of the Prototype

It is crucial for secure software development to use various types of security knowledge. NII provides security requirements modeling support system (Security analysis tool) for a misuse case diagram that enables the association of security knowledge with elements that constitute the diagram.

The functions provided by the Security analysis tool are the following.

1. to draw diagrams
2. to associate security knowledge with elements of a diagram
3. to browse security knowledge in the knowledge base
4. review function
5. artifact management

The following diagram shows an overall screenshot of the Security Analysis Tool.

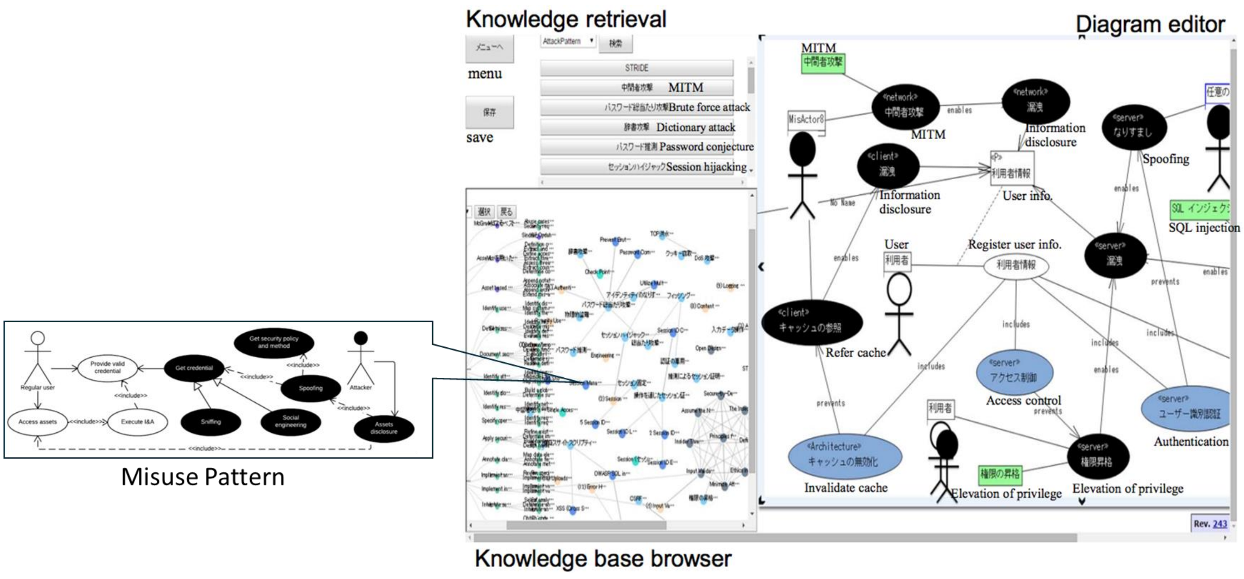


Figure 19. Screenshot of the Security analysis tool.

On the right side of this figure is an editor that creates a misuse case diagram. The lower left side of this figure represents the entire structure of the knowledge base. When a node is selected, the detailed information regarding the knowledge is shown in a sub-window.

Components

Component Module 1: Software Security Knowledge Base

Software security knowledge base consists of several knowledge catalogues as follows:

- Principle is a statement of general security wisdom derived from experience.
- Attack pattern is developed by reasoning over large sets of software exploits.
- Guideline is a recommendation for things to do or to avoid during software development, described at the semantic level.
- Rule is a recommendation for things to do or to avoid during software development, described at the syntactic level.
- Vulnerability is the result of an analysis against a software that an attacker can use to gain illegal access to – or negatively affect the security of – a computer system.
- Exploit is a particular instance of an attack on a computer system that leverages a specific vulnerability or set of vulnerabilities.
- A historical risk is a risk identified in the course of an actual software development effort.
- Process/Methodology/Standard is a process, methodology, or standard regarding the development of secure software.
- Component is an element that consists of the aforementioned process, methodology, or standard. For example, the CLASP process consists of “Activity,” and/or “Sub-activity”.
- Security pattern is a solution to the recurring problem for security.

Software security knowledge base forms a highly complex graph structure, in which knowledge is represented by links. The knowledge base is visually represented in Figure 20. There are fourteen types of knowledge, and knowledge instances are identified by colors allocated to each knowledge type. When a node (knowledge instance) is selected, the detailed information is shown as in the bottom right side of Figure 21.

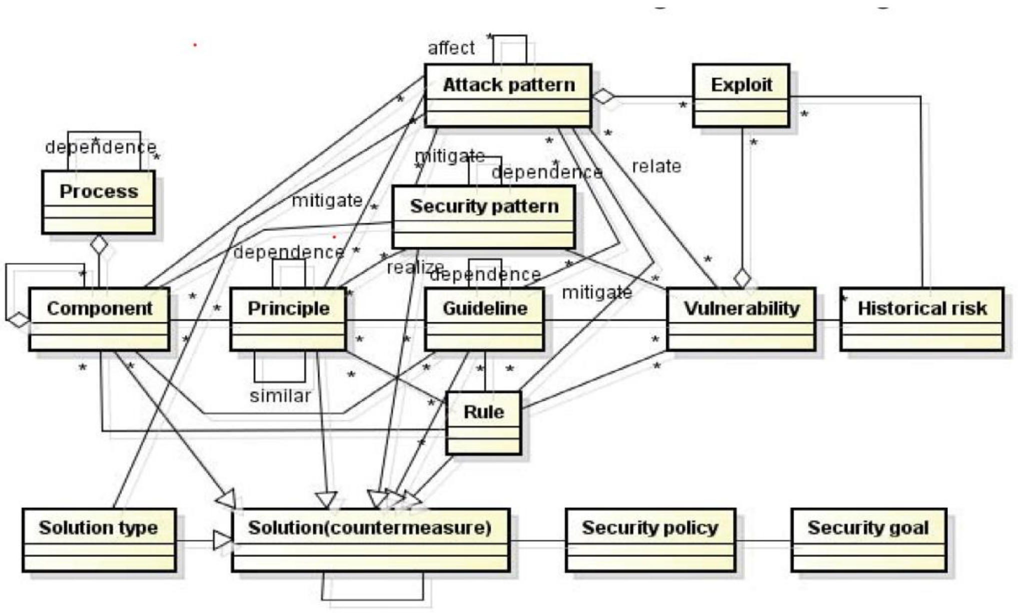
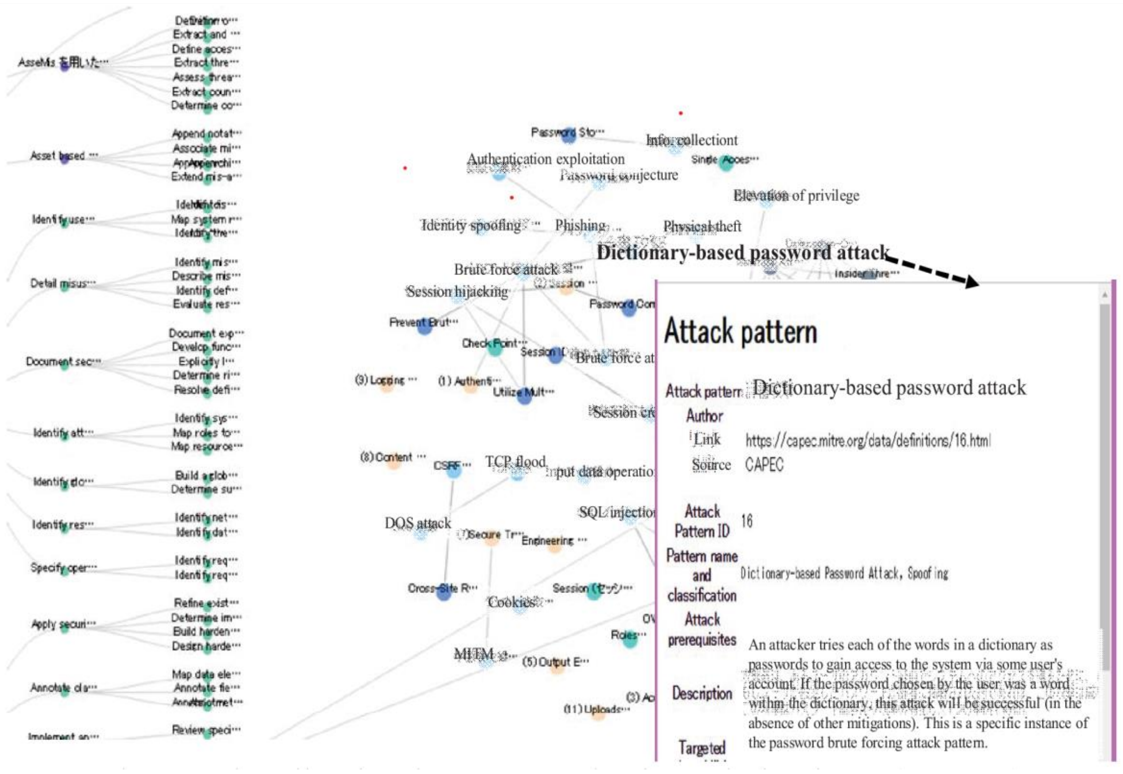


Figure 20. Conceptual model of software security knowledge.



Attack pattern

Attack pattern	Dictionary-based password attack
Author	
Link	https://capec.mitre.org/data/definitions/16.html
Source	CAPEC
Attack Pattern ID	16
Pattern name and classification	Dictionary-based Password Attack, Spoofing
Attack prerequisites	
Description	An attacker tries each of the words in a dictionary as passwords to gain access to the system via some user's account. If the password chosen by the user was a word within the dictionary, this attack will be successful (in the absence of other mitigations). This is a specific instance of the password brute forcing attack pattern.
Targeted	

Figure 21. Visualized software security knowledge base.

We describe the operations for registering knowledge in the software security knowledge base system and associating the registered knowledge.

For example, we create “Spoofing” information in Threat Type knowledge catalog: “Microsoft STRIDE”. And we create “Identity Spoofing” in “Attack Pattern” knowledge catalog: “Common Attack Pattern Enumeration and Classification (CAPEC)” in the same way. We associate “Spoofing” information in Threat Type knowledge with “Identify Spoofing” in Attack Pattern knowledge.

1. At first, Select “Threat Type” in the Top page of Software Security Knowledge Base. (Figure 22)

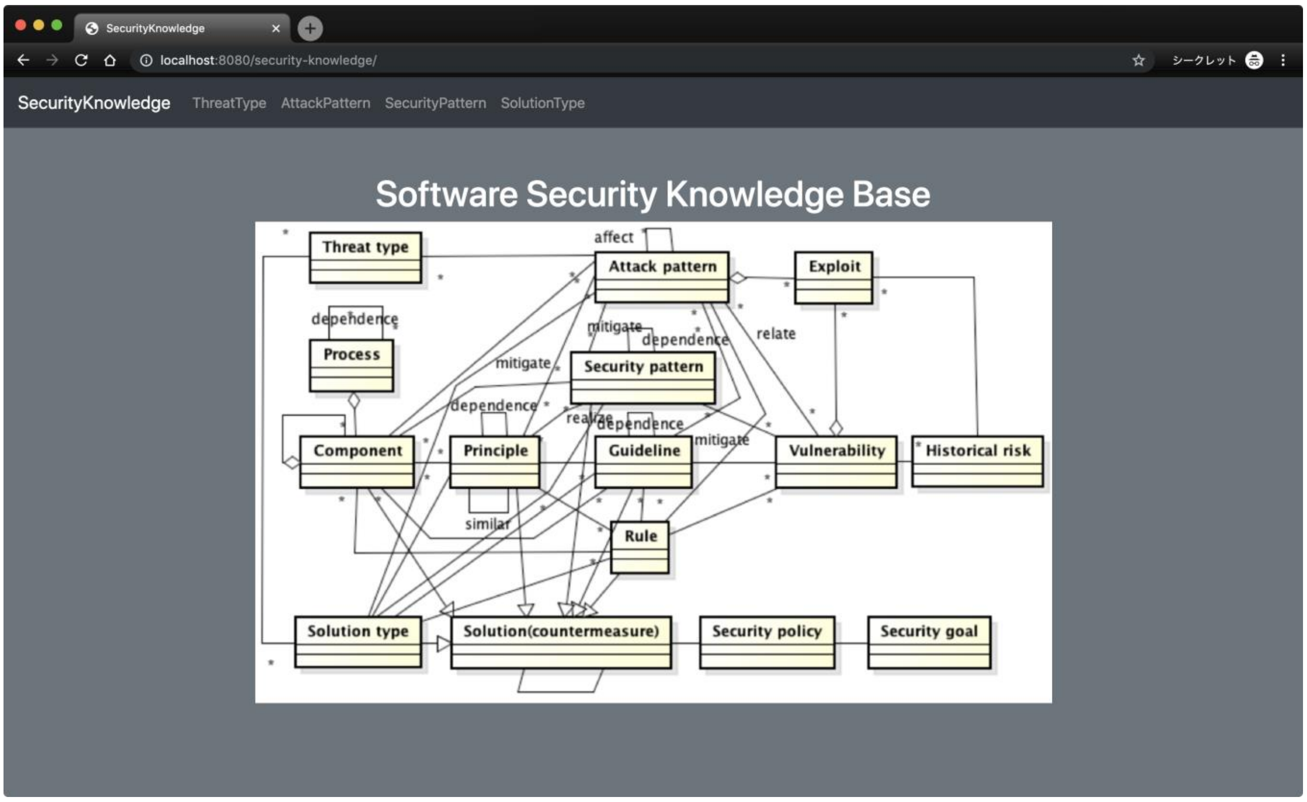
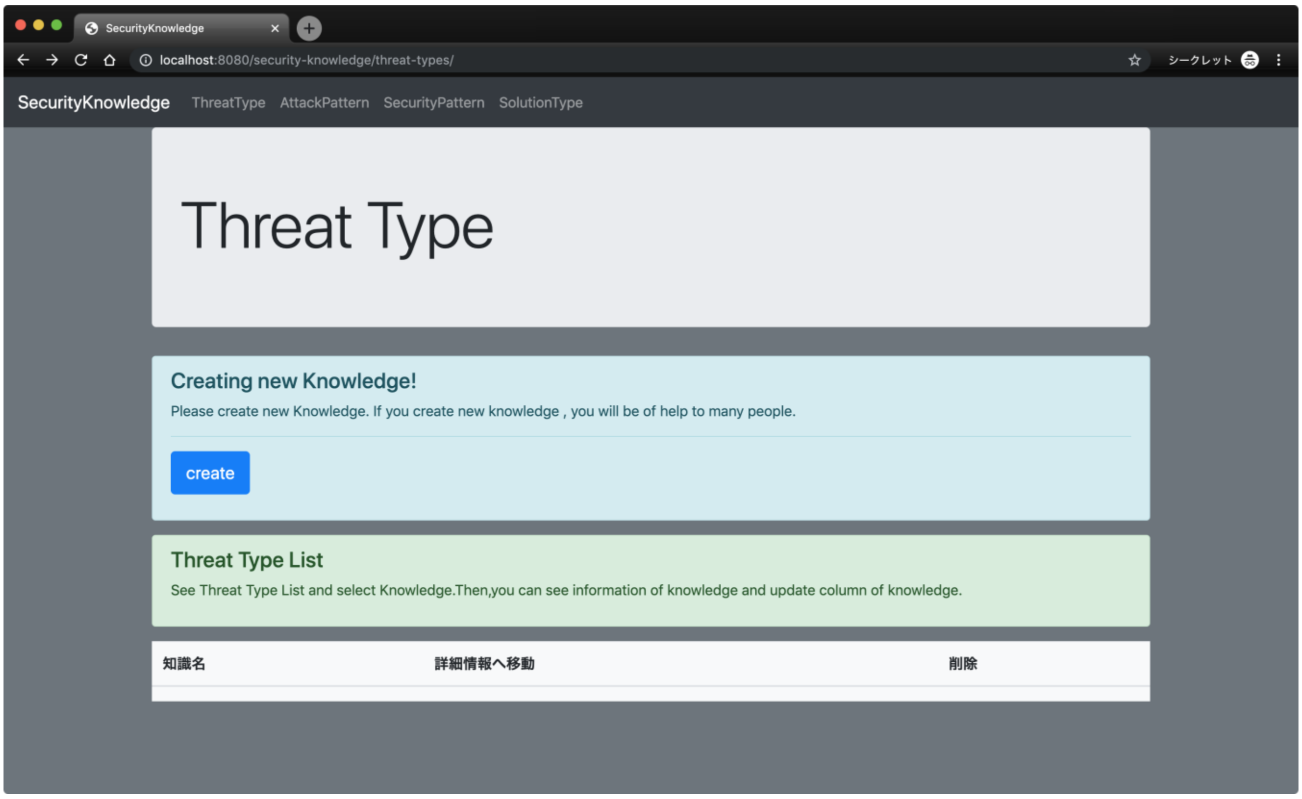


Figure 22. Top page of Software Security Knowledge Base system

2. Click “create” bottom (Figure 23) and fill the Spoofing information on the registration form (Figure 24).


SecurityKnowledge

localhost:8080/security-knowledge/threat-types/

SecurityKnowledge ThreatType AttackPattern SecurityPattern SolutionType

Threat Type

Creating new Knowledge!

Please create new Knowledge. If you create new knowledge , you will be of help to many people.

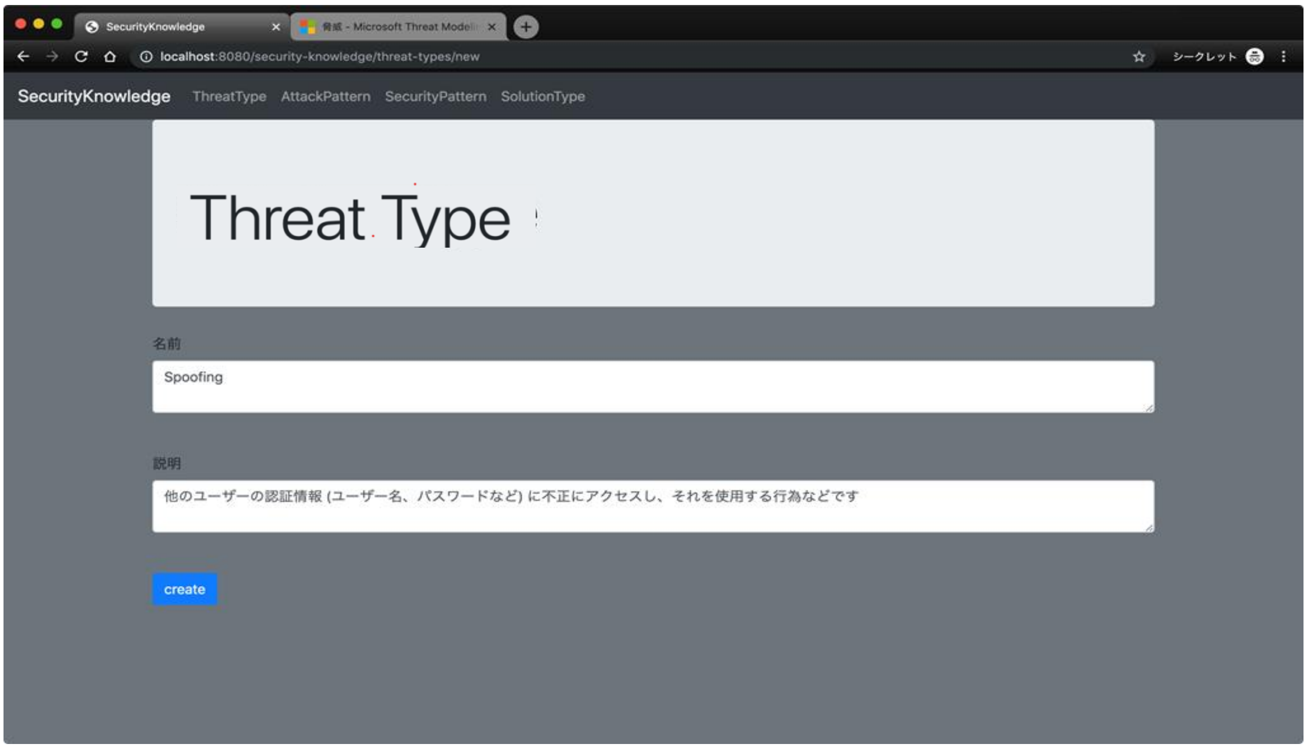
create

Threat Type List

See Threat Type List and select Knowledge.Then,you can see information of knowledge and update column of knowledge.

知識名	詳細情報へ移動	削除

Figure 23. Threat Type Screen


SecurityKnowledge | 脅威 - Microsoft Threat Modeli

localhost:8080/security-knowledge/threat-types/new

SecurityKnowledge ThreatType AttackPattern SecurityPattern SolutionType

Threat Type

名前

Spoofing

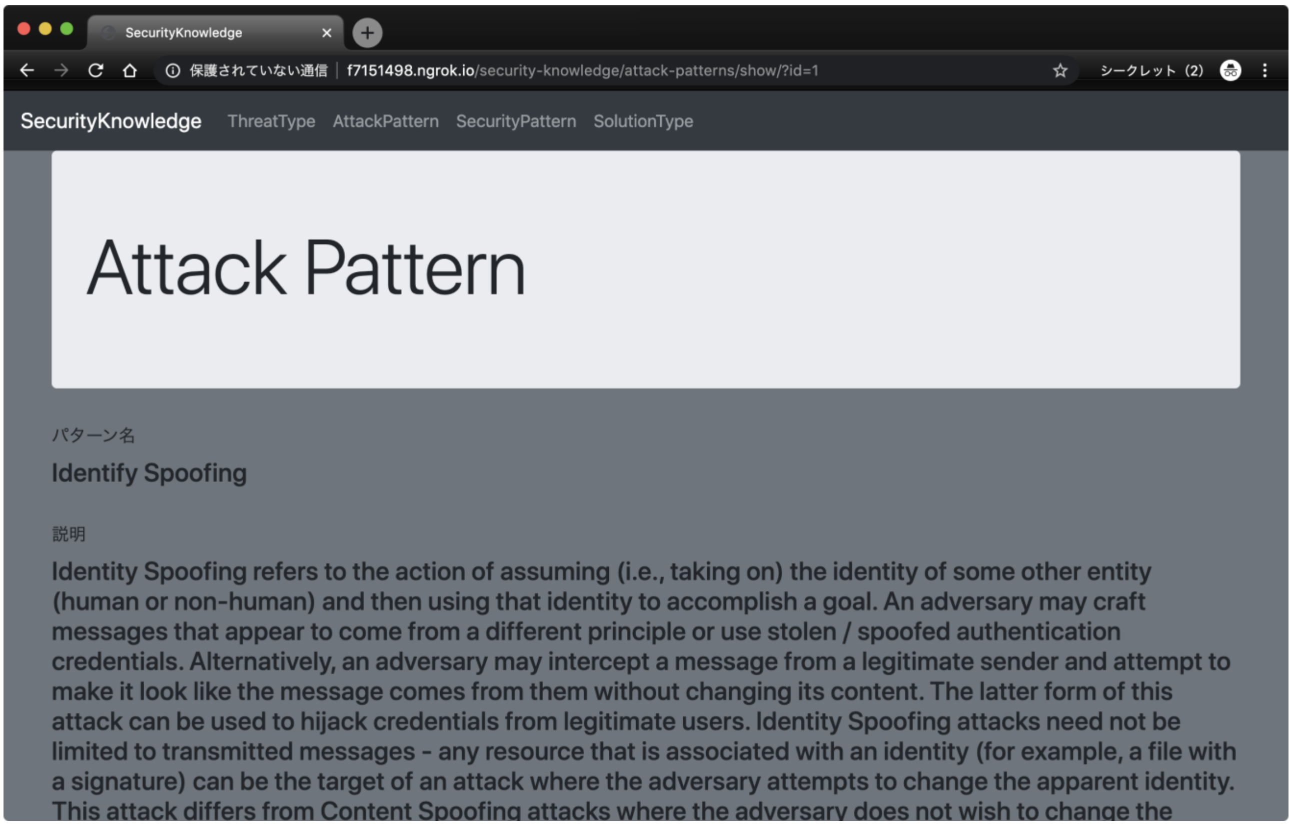
説明

他のユーザーの認証情報 (ユーザー名、パスワードなど) に不正にアクセスし、それを使用する行為などです

create

Figure 24. Threat Type registration form

3. Next, create Attack pattern knowledge in the same way. (Figure 25)



Attack Pattern

パターン名

Identify Spoofing

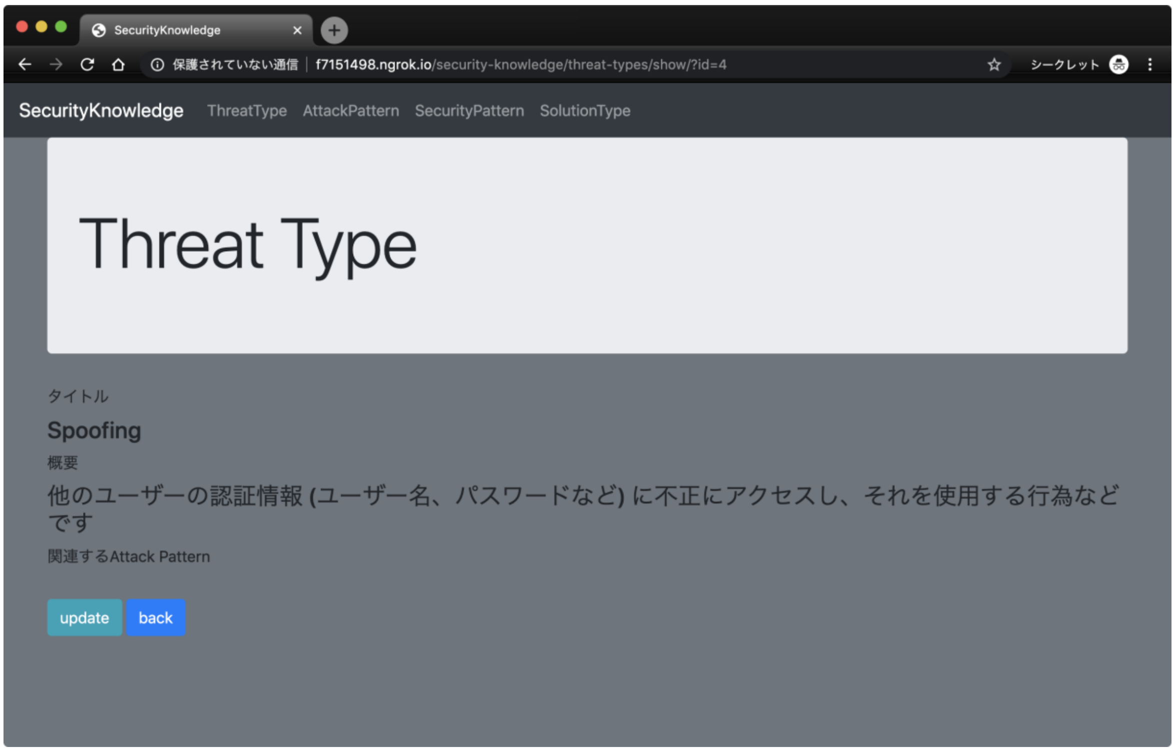
説明

Identity Spoofing refers to the action of assuming (i.e., taking on) the identity of some other entity (human or non-human) and then using that identity to accomplish a goal. An adversary may craft messages that appear to come from a different principle or use stolen / spoofed authentication credentials. Alternatively, an adversary may intercept a message from a legitimate sender and attempt to make it look like the message comes from them without changing its content. The latter form of this attack can be used to hijack credentials from legitimate users. Identity Spoofing attacks need not be limited to transmitted messages - any resource that is associated with an identity (for example, a file with a signature) can be the target of an attack where the adversary attempts to change the apparent identity. This attack differs from Content Spoofing attacks where the adversary does not wish to change the

Figure 25. Registered Identify Spoofing in Attack Pattern knowledge

4. At last, we associate Threat Type knowledge with Attack pattern knowledge.

   1) Select “Spoofing” from Threat Type List and Click “update” bottom. (Figure 26)


SecurityKnowledge

保護されていない通信 | f7151498.ngrok.io/security-knowledge/threat-types/show/?id=4

SecurityKnowledge ThreatType AttackPattern SecurityPattern SolutionType

Threat Type

タイトル

Spoofing

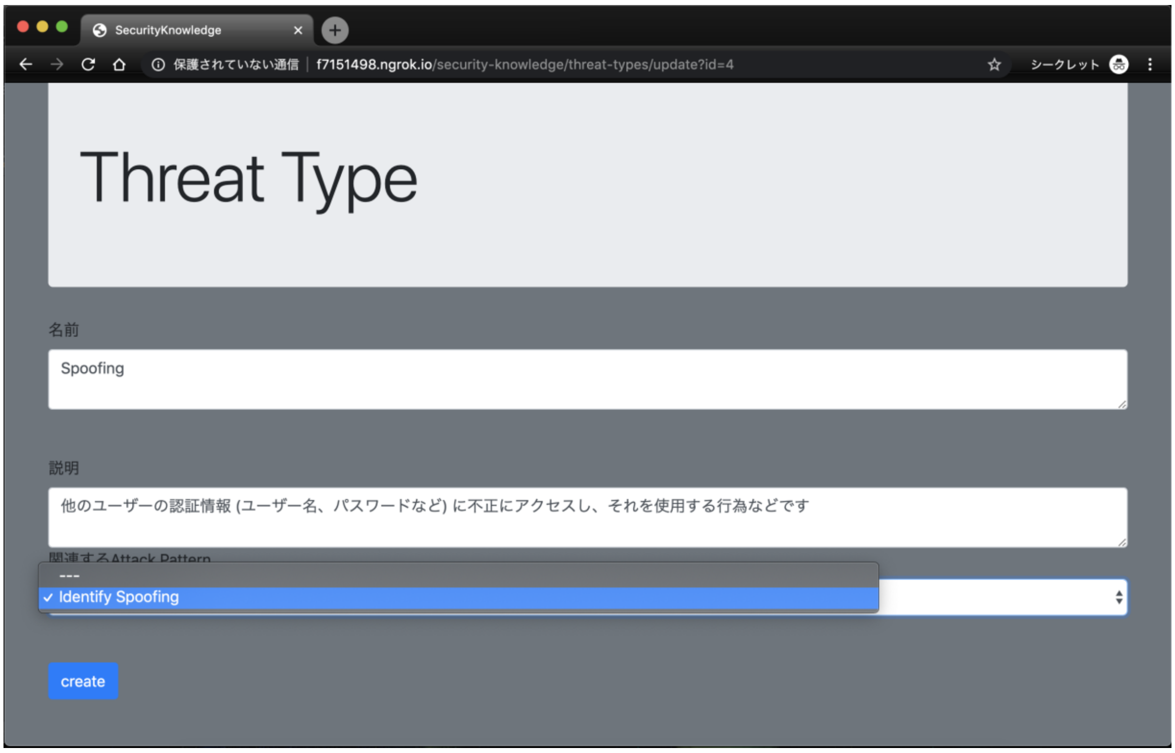
概要

他のユーザーの認証情報 (ユーザー名、パスワードなど) に不正にアクセスし、それを使用する行為などです

関連するAttack Pattern

Figure 26. Spoofing information in Threat Type Screen

2) Chose "Identify Spoofing" from pull-down menu. (Figure 27)


SecurityKnowledge

保護されていない通信 | f7151498.ngrok.io/security-knowledge/threat-types/update?id=4

Threat Type

名前

Spoofing

説明

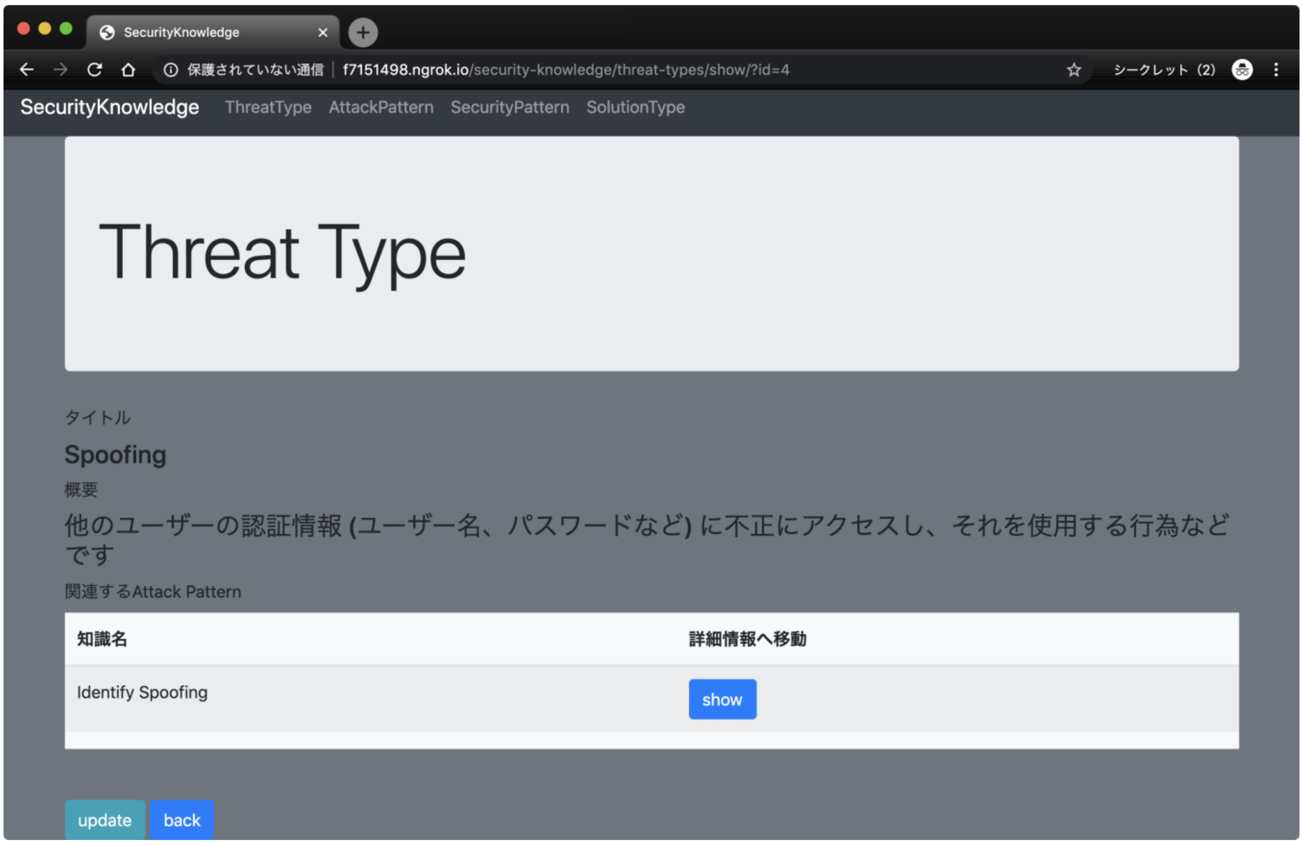
他のユーザーの認証情報 (ユーザー名、パスワードなど) に不正にアクセスし、それを使用する行為などです

関連するAttack Pattern

✓ Identify Spoofing

Figure 27. Edit form in Threat Type screen

3) “Spoofing” in Threat Type knowledge is associated with “Identity Spoofing” in Attack pattern knowledge. (Figure 28)


SecurityKnowledge
保護されていない通信 | f7151498.ngrok.io/security-knowledge/threat-types/show/?id=4
シークレット (2)
SecurityKnowledge ThreatType AttackPattern SecurityPattern SolutionType
update
back

Threat Type

タイトル

Spoofing

概要

他のユーザーの認証情報 (ユーザー名、パスワードなど) に不正にアクセスし、それを使用する行為などです

関連するAttack Pattern

知識名	詳細情報へ移動
Identify Spoofing	show

Figure 28. Association of Treat Type knowledge “Spoofing” and Attack pattern knowledge “Identity Spoofing”

3.2 Package Information & Installation Instructions

The following section provides a list of the tools and dependencies for the correct operation of the Security analysis tool.

Required Tools and dependencies

The following tools and dependencies are required to use Security analysis tool:

The security analysis tool is under development and subject to change the following tools and dependencies.

- Docker
- MySQL

Install Docker

The installation can be found in the Docker's webpage https://docs.docker.com/v17.09/engine/installation/, but in the section "Install Docker" in the "Demonstration 1 – Crypto Companion Database (CCDB) " there is a list of the main steps and commands for Windows and Ubuntu.

Install MySQL

The installation can be found very detailed in MySQL's webpage.

For Windows: https://dev.mysql.com/doc/mysql-installation-excerpt/5.5/en/windows-installation.html

For Linux: https://dev.mysql.com/doc/mysql-installation-excerpt/5.5/en/linux-installation.html

Download and Run Demonstrator

The security analysis tool is under development and it is not currently available to download and run.

A step by step guide will be provided after the Security analysis tool is completed.

User Manual

The security analysis tool is under development and some of its functionalities may change in the near future, so a user manual is not provided yet.

The user manual will be provided after the Security analysis tool is completed.

Licensing (if applicable)

Licensing for all the components/software used:

- Docker is under Apache License 2.0 (https://www.apache.org/licenses/LICENSE-2.0) form more detail go to https://www.docker.com/legal/components-licenses.
- MySQL (GPL) is under GNU General Public License version 2.0 (https://opensource.org/licenses/GPL-2.0)

Reference use case

The requirements of each M-Sec pilot are described in each use case diagram. As a use case is a representation of a normal user's interaction with the system, it's difficult to elicit security threats and requirements by only the use case.

On the oter hand, a misuse case is a represetion of actions that can be performed by attackers to harm the systems. By using misuse cases, we can identify and mitigate the threat.

We provide Security analysis tool for a misuse case diagram that enables the association of security knowledge with elements that constitute the diagram.

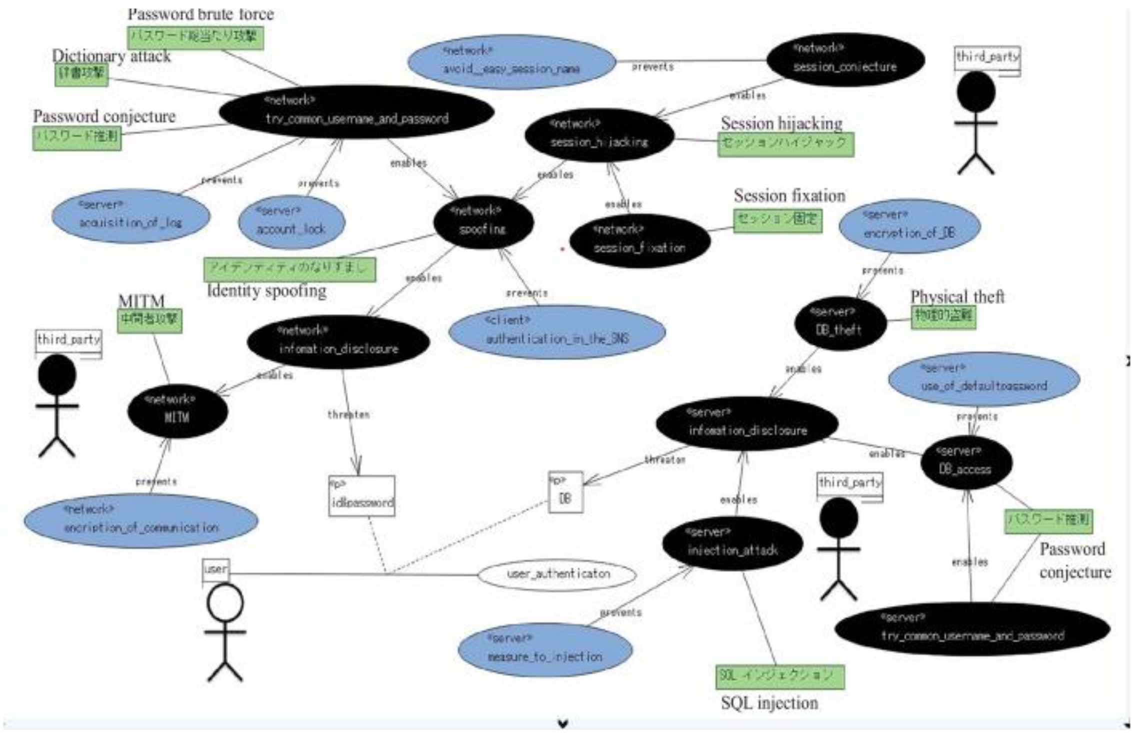


Figure 29. Example use and misuse cases.

Summary

In this document, we describe the Security analysis tool with the list of its components. As the Security analysis tool is still under development, some complementary information such as the Installation guide and software needed to run, the user manual will be provided in the second version of the deliverable.

Next Steps

The next steps during next year will be to enhance the Software security knowledge base by adding various types of security knowledge and to make the existing and the new easier and friendlier to use.

In the next steps, we need to set various types of security knowledge in the software security knowledge base. For example, there are CLAPS (Comprehensive, Lightweight Application Security Process), OWASP (The Open Web Application Security Project) Cheat Sheet, CAPEC (Common Attack Pattern Enumeration and Classification) and so on.

At the same time, we'll develop Security Requirement Modeling Support System in order to create misuse case diagram based on security knowledge in the software security knowledge base.

And Security analysis tool will enable both the reference to the knowledge base and the creation of a diagram by embedding the elements of the knowledge base into the diagram.

The association of misuse case diagrams with security knowledge results in the recording of design rationale.

The security analysis tool is expected to provide users with useful information by considering the reuse of misuse case diagrams. In addition, the association of elements of a misuse case diagram with knowledge in the knowledge base may enable us to support traceability.

4. Conclusion

The main focus of Task 4.4 is to establish engineering foundations to support the development of secure smart city applications of the M-Sec system.

In this document, we have presented the description of two prototypes, the Crypto Companion Database and the Security analysis tool. The Crypto Companion Database provides sensitive data security that cannot be stored on Blockchain. The Security analysis tool provides security requirements that the use case diagram cannot elicit. They contribute to establishing engineering foundations to support the development of secure applications of the M-Sec system.

The next step for the M18-M30 period will be to enhance the functionalities and to make the existing and the new more easy and friendly to use. And the final version of this deliverable will be delivered in month M30.