



**Advanced Cyber-Threat Intelligence, Detection, and Mitigation
Platform for a Trusted Internet of Things
Grant Agreement: 786698**

D7.3 CYBER-TRUST authority and publishing management

**Work Package 7: Distributed ledger technology for
enhanced accountability**

Document Dissemination Level

P	Public	<input checked="" type="checkbox"/>
CO	Confidential, only for members of the Consortium (including the Commission Services)	<input type="checkbox"/>

Document Due Date: 31/10/2019

Document Submission Date: 06/11/2019



Co-funded by the Horizon 2020 Framework Programme of the European Union



Document Information

Deliverable number:	D7.3
Deliverable title:	CYBER-TRUST authority and publishing management
Deliverable version:	1.01
Work Package number:	WP7
Work Package title:	Distributed ledger technology for enhanced accountability
Due Date of delivery:	31/10/2019
Actual date of delivery:	31/10/2019
Dissemination level:	Public
Editor(s):	Clément Pavué (SCHAIN)
Contributor(s):	
Reviewer(s):	Dimitris Kavallieros (KEMEA) Stefano Cuomo (MATHEMA)
Project name:	Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Project Acronym	Cyber-Trust
Project starting date:	1/5/2018
Project duration:	36 months
Rights:	Cyber-Trust Consortium

Version History

Version	Date	Beneficiary	Description
1.0	24/10/19	SCHAIN	Version sent to reviewers
1.01	6/11/19	SCHAIN	Updated version regarding reviewers' comments

Acronyms

ACRONYM	EXPLANATION
DLT	Distributed Ledger Technology
UI	User interface
URL	Uniform Resource Locator
NPM	Node Package Manager
CI / CD	Continuous Integration / Continuous Delivery
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface
OS	Operating System
UUID	Universal Unique Identifier
CPE	Customer-premises equipment

Table of Contents

1. Executive summary	6
2. Technical details.....	7
2.1 Changes since D7.2.....	7
2.2 The Development environment	7
2.3 DLT client implementation	8
2.4 DLT documentation	9
2.5 DLT Mock-up UI.....	9
3. Example of interaction with the UI-MockUp	11
4. Conclusion.....	19

Table of Figures

Figure 2-1: Updated Class Diagram.....	7
Figure 2-2: Example of form in the DLT UI.....	10
Figure 3-1: Creation of an organization on the DLT.....	11
Figure 3-2: The Previously created organization creates a device Class	12
Figure 3-3: A patch is created for the new DeviceClass.....	13
Figure 3-4: A citizen register his device on the platform.....	14
Figure 3-5: A patch of the device is being set	15
Figure 3-6: Network of actors	16
Figure 3-7: After the incident the device is updated to a new version of OS.....	17
Figure 3-8: Updated network of actors.....	17

1. Executive summary

The Cyber Trust DLT will bring trust between the different components developed by the different members of the consortium. It will also be used as a safe way of exchanging information for any partners inside Cyber Trust.

With this first iteration of the development of the DLT we focused on the administration of the organization and devices under the watch of Cyber Trust. At this stage, the DLT store with respect to the legal framework, the list of patch and configuration published by an organization. Devices will be able to download these patches and configurations not directly from the DLT but from a safe URL provided the DLT. It will improve the integrity of the device on the network ensuring they are running on a firmware published by their manufacturers and not one from a malicious party.

Another aspect of this iteration is the authority management through the DLT. This version can manage the acceptance and revocation of employees of an organization authority on device based on the owner of device consent. This way device flagged as malicious by the Profiling Service [A17] if the owner consent to it, will be updated to a newer version fixing a potential breach currently used to perpetrate the attack or shut down for example if there is no other to stop the attack made by the device.

In this proof of concept of what our DLT is capable of, we set up a first scenario in which two organizations are members of the Cyber-Trust and share information regarding the class of device they are selling on the market, create patches and configurations to allow devices to download them from a trusted source, administrator able to operate of device in case of an attack. All these operations create new transactions that will be validated or not by our smart contract and propagated to the other peers if validated. EU Citizen are capable of registering devices on their own and ask for the device to get the metadata related to the latest software update available from the DLT.

2. Technical details

2.1 Changes since D7.2

We made a few modifications from the class diagram we published in D7.2. These modifications are due to errors we found during the development process. The modification is that the aggregation of Organization to Patch and the one from Organization to Configuration are now between Device Class to Patch and DeviceClass to Configuration. Here is the updated version of the class diagram.

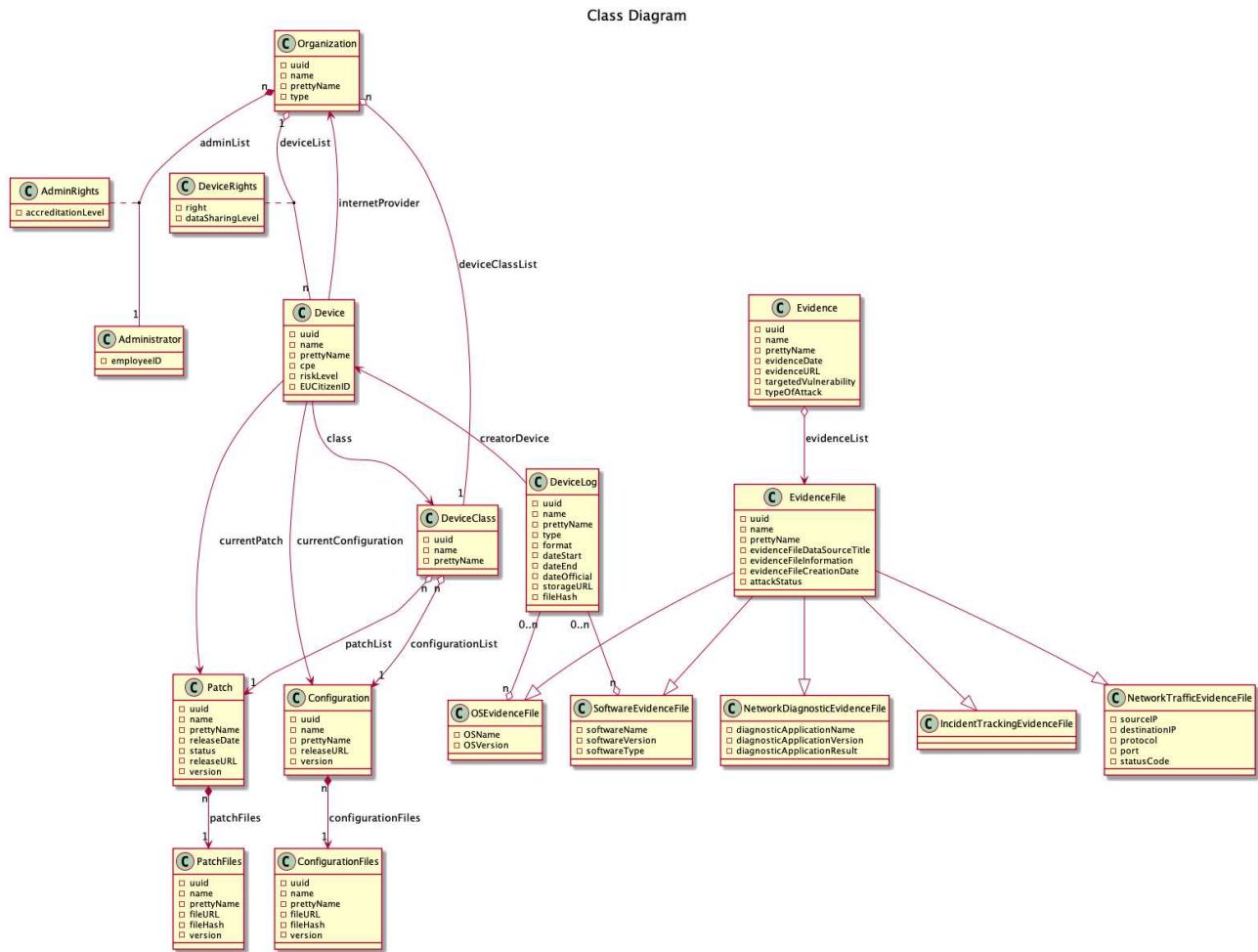


Figure 2-1: Updated Class Diagram

2.2 The Development environment

To develop the solution, we have used **Node.js** 8.12.0 which is the last stable version at the beginning of the development process. We installed a few dependencies via **NPM** to develop some of the feature of the project. To implement our API, we used **Loopback 4**. As stated in D7.2 we have used **HyperLedger** in its 1.4.0 version to use the private data that was introduced in this release. For the front-end we have use **React**. All data used during the development phase was mocked and no personal data was used.

To host our code, we first use Scorechain’s **Gitlab** to fasten up the configuration of the project such as configuration for Gitlab CI / CD as it was already pre-configured because of a previous project

realized by Scorechain. The final version of the code was however published to the Gitlab of the consortium. As we described in the previous deliverable of D7.2 we use **Docker** and **Kubernetes** to deploy our mock-up and we will continue to employ the same technologies later for deploying the DLT to partners premises. Currently due for our mockup we deploy four different images. The Orderer peer, the two organization peers and their APIs to interact with them. The last one is a **postGres** database that register logs of the transactions created on the platform for debugging and demonstration purpose. This functionality won't be pushed to production.

We worked in an agile environment. We create user-stories in the Gitlab to specify how the API should behave for each endpoint we aim to implement. Once the user story implemented, it was integrated into the test plan. If the developer thinks the implementation is correct regarding the user story, the status of the user story become User Acceptance Test. This means the creator of the user story test the implementation propose by the developer. If it is satisfying the user story is validated. Otherwise the developer makes changes to fulfil the user story.

Beside this document, the deliverable comes in different part. The first one is the implementation of the smart contract and an API to interact with them. This is the part some of the components developed by the Cyber Trust consortium will interact with. The second part is the documentation of the code and the API. This will be a reference for the integration of the DLT inside Cyber Trust. The last one is a front end used as a demonstration material for our DLT.

2.3 DLT client implementation

The first step of developing the client of the DLT, the one that will run on partners backend, was to create the smart contracts. We used the Go language to create them. Smart contract is the core of the deliverable and of the global solution that Scorechain will propose at the end of the project, so it was important to begin with that task. The purpose of a smart contract is to verify that the data received are correct and can be stored in the DLT and propagated to the peers authorized to access the newly added data. This is the most critical part of the DLT because DLT is used as a trusted source of information and smart contract is used as a validator for the data. In order to have trust in smart contracts we have created a set of unit tests for them. Each Go file is tested in a file with the same name followed by test under the same folder. We do integrity checks such as what happens when we try to create a device that is linked to a non-existing organization. We ensure that smart contracts are responding correctly to any cases we could imagine. Once this step was over, we build the API on top of our smart contracts to interact with them.

Another important part of our effort to develop the client was to make it easy to deploy remotely. Indeed, several components had to be deployed in the context of this mock-up. The Orderer, the central part of the DLT, the two peers, one for each organization in this mock-up environment. The API on top of each peer of the DLT has to be deployed separately as well. All of this component has to be synchronized on start up as they communicate together to make the mock-up functional. The DLT is deployed on a single VM on the OTE testbed. All partners can access it. Of course, each unit test created in our process of development is executed after each commit to ensure the stability of the platform.

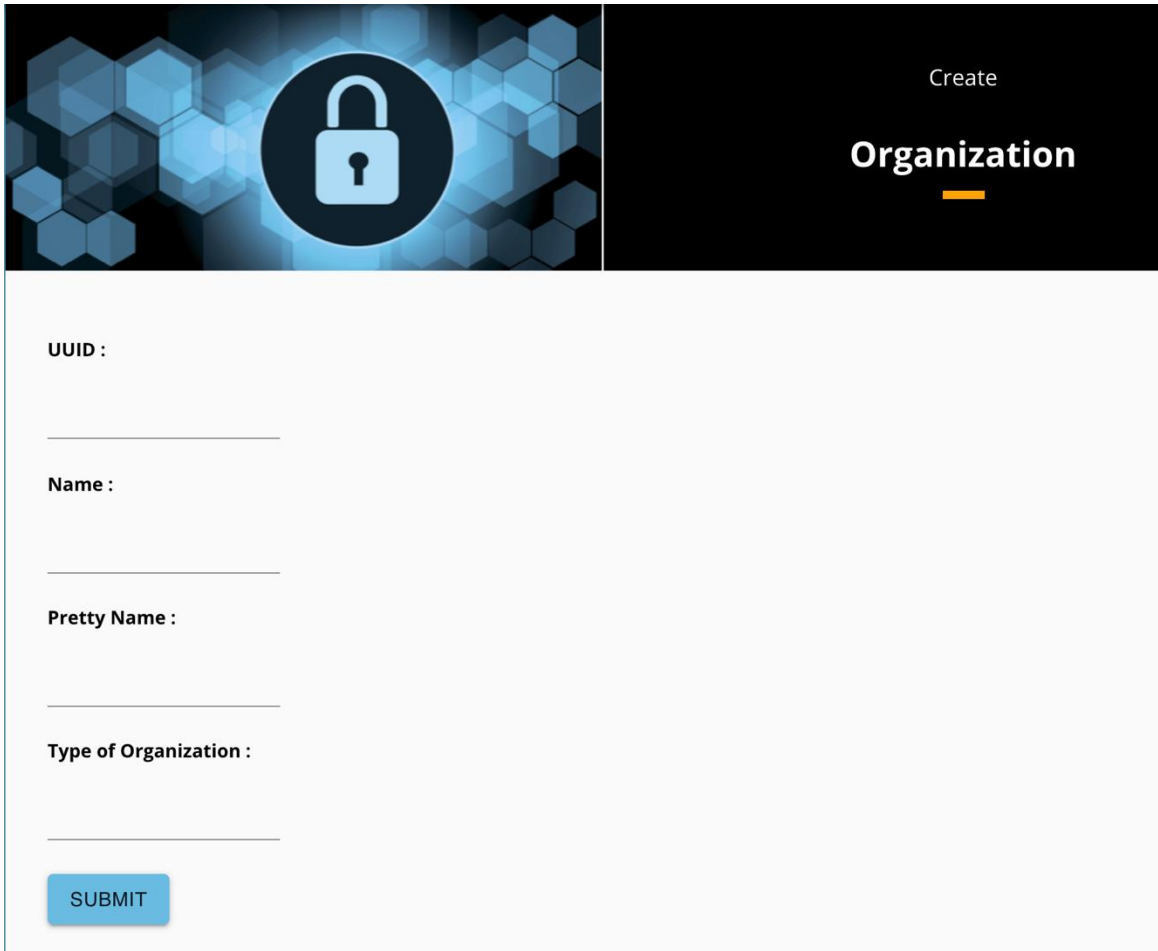
Regarding the consensus mechanism used by the Orderer, we used Solo. As stated in D7.2 we keep the simplest consensus mechanism for the moment and we will choose the final consensus during the pilot testing phase. We can afford that choice due to the flexibility of HyperLedger Fabric. Indeed, consensus mechanism are plug and play, so we can change to one another easily.

2.4 DLT documentation

The documentation of this deliverable comes in two part. The first one is the documentation related to our API in order for the other members of the consortium to know how to interact with our API as the DLT is a central component of the project it was important for us to provide as much details on how to interact with it. Thus, we used Swagger to create the documentation. Swagger is an open source software framework that helps developer to create RESTful applications. We designed the API specifications with Swagger before the beginning of the development phase. This way of working is called API-first development. Using this approach gave us a clear vision of the data flow we wanted to implement in D7.2. We then applied our documentation to a scenario we created to ensure the solution we aim to deliver match the project requirement we established in previous deliverable of Cyber Trust such as D2.3 and D7.1. This scenario will be depicted in more details in section 3 of this document.

2.5 DLT Mock-up UI

During the development phase described previously, we feel the need to test our work not only with APIs call but in a more visual way. During meetings with the different partners and in workshop, it was hard to keep the attention of the audience with only HTTP calls result, especially for non-technical person. We then decide to create a minimalistic UI. Each endpoints of our API are implemented in this mock-up. Each POST call has a page with a form to fill in for each field necessary to ensure the good validation of the data via our smart contract. For instance, this the form to create an entity on our DLT.



The screenshot shows a web interface for creating an organization. At the top left is a header image with a blue padlock icon on a hexagonal pattern. To the right, the text 'Create Organization' is displayed in white on a black background, with 'Organization' underlined. Below this is a form with the following fields: 'UUID :', 'Name :', 'Pretty Name :', and 'Type of Organization :'. Each field has a horizontal input line. At the bottom left of the form is a blue 'SUBMIT' button.

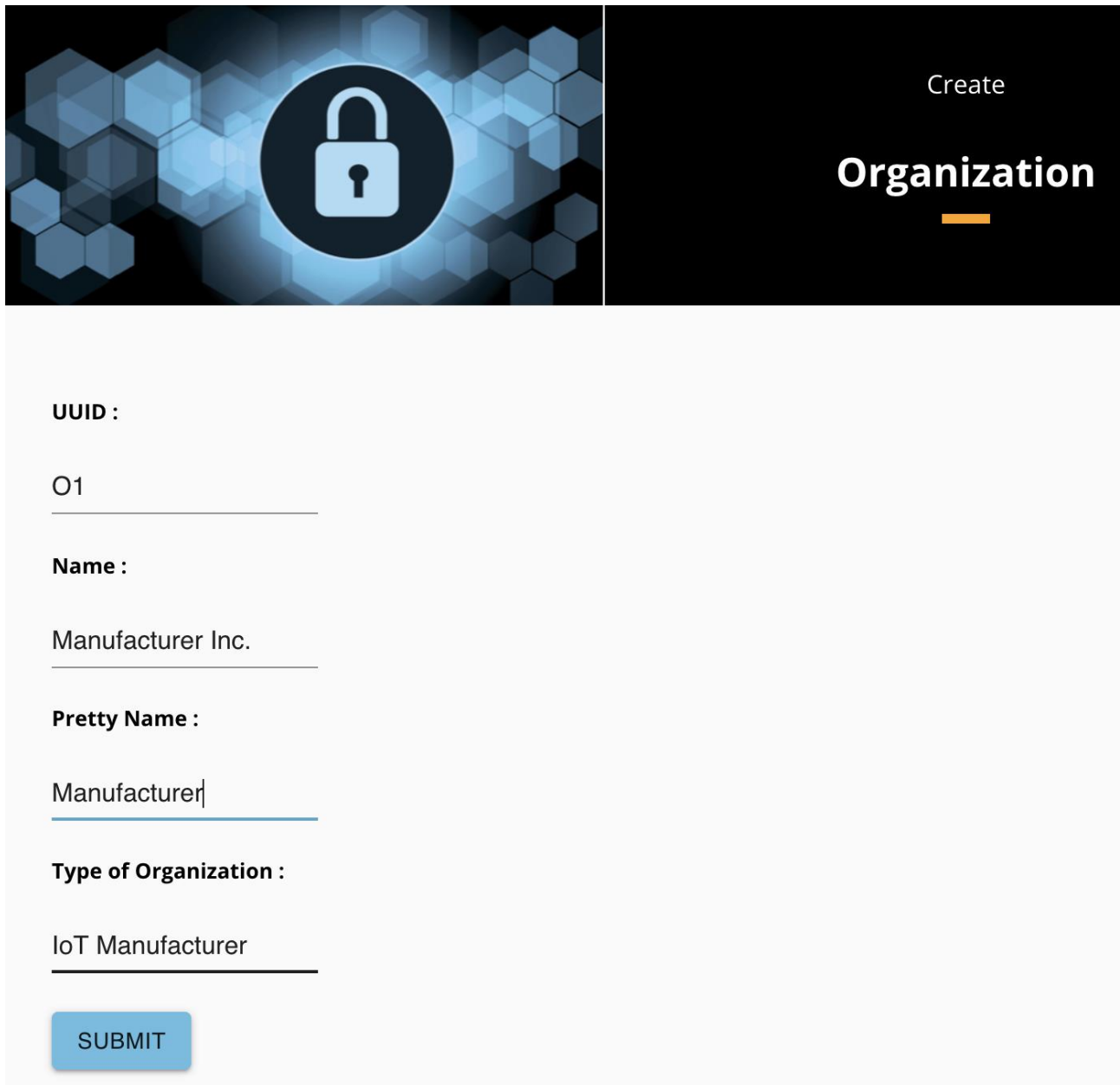
Figure 2-2: Example of form in the DLT UI

By filling this form the end user will create a new transaction that will be sent to the central node, the Orderer, on the Cyber-Trust backend and propagated to the other organizations running a peer of the DLT.

3. Example of interaction with the UI-MockUp

To help to understand the power of our solution and how it is easy to use, we design a small scenario on how our DLT can be used to monitor the version of OS a device is using and ensuring when it update its firmware, it download the new version from the official source provided by the manufacturer of the device itself.

First of all, we will create the network of actors for this scenario. Thus, we begin the IoT device manufacturer named Manufacturer Inc.



Create

Organization

UUID :

O1

Name :

Manufacturer Inc.

Pretty Name :

Manufacturer

Type of Organization :

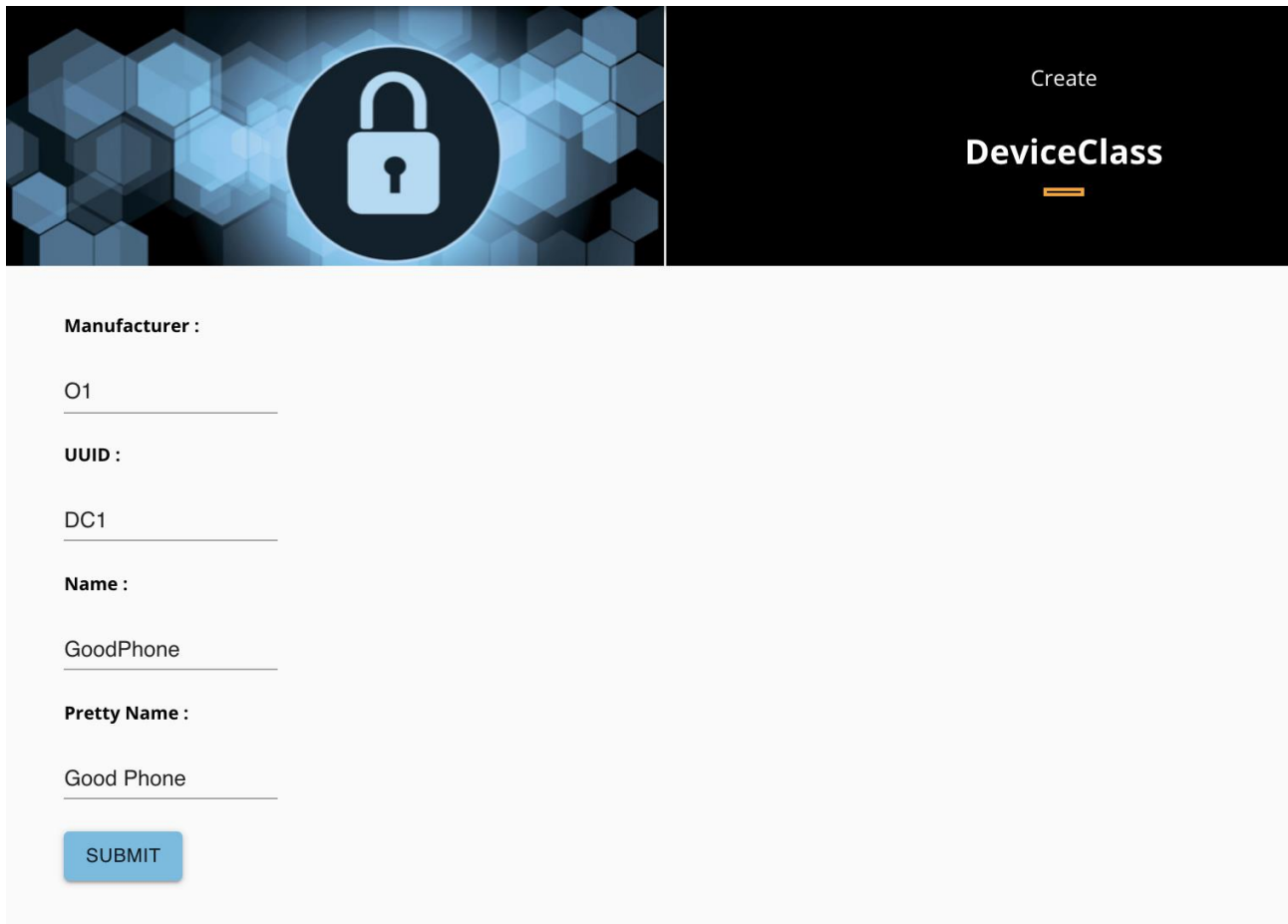
IoT Manufacturer

SUBMIT

Figure 3-1: Creation of an organization on the DLT

At the beginning an organization just need to refer its name on the platform and type to the platform. Its Uuid will be attributed automatically by the Cyber Trust backend but we keep it visually in our UI as it's not the DLT task to do deduce it. Here the newly created organization has been attributed the uuid O1.

Then, our device manufacturer registers the information relative to the type of device they're selling on the market. Right now, they're just producing one type of device, the Good Phone.



The screenshot shows a web interface for creating a device class. The top header is black with the text 'Create DeviceClass' in white. Below the header is a form with the following fields:

- Manufacturer :** O1
- UUID :** DC1
- Name :** GoodPhone
- Pretty Name :** Good Phone

A blue 'SUBMIT' button is located at the bottom of the form.

Figure 3-2: The Previously created organization creates a device Class

As you can see the manufacturer of the class of device declare itself as the producer of the device and in a first step only declare the name of the device class. In order to ensure the integrity of their device, IoT manufacturer have to publish the patch a device can download and run on. The uuid attributed to the new device class is DC1. This is the next step O1 has to do to be complete its Cyber Trust compliance.



UUID:

P1

Name:

path1

Pretty Name:

path1

Release Date:

date1

Status:

current

Release URL:

releaseUrl1

Version:

v111

DeviceClassUuid:

DC1

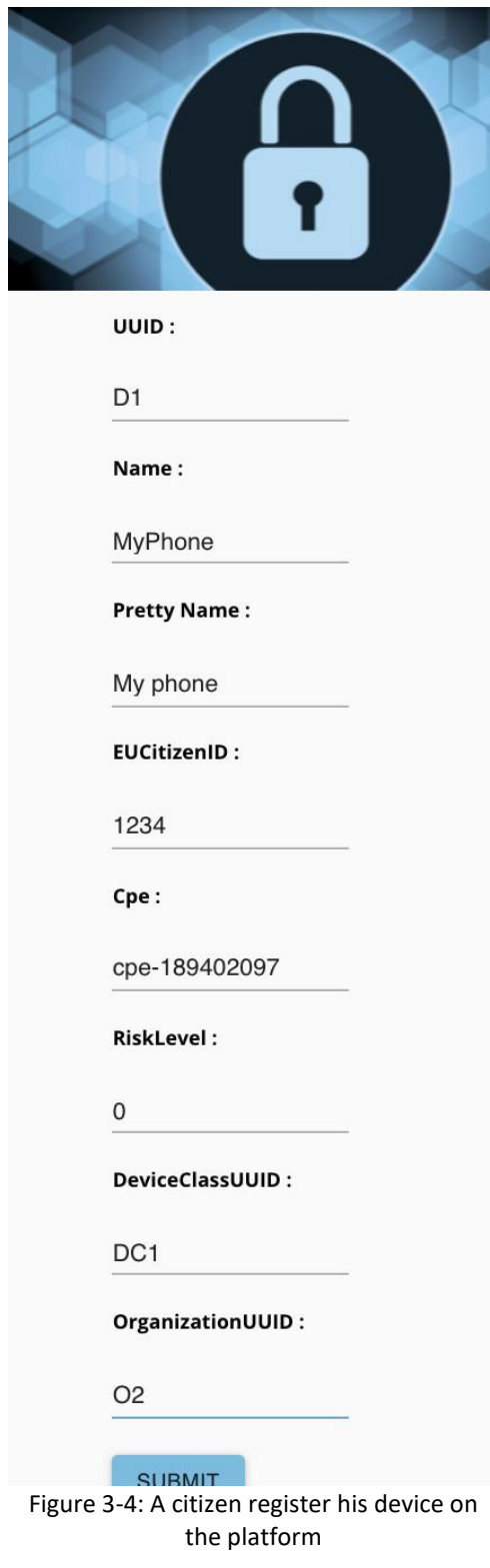
SUBMIT

Figure 3-3: A patch is created for the new DeviceClass

To register a patch for its device class O1 has to declare its name, the date of release, the status of the patch (it's an enumeration with three possibilities current, vulnerable and obsolete). The next is

probably the most interesting of this form. It's the safe location where the device can find the patch to download. The remaining field are the identifier of the version and the uuid of the publisher of the patch, in our case it's the manufacturer of the device itself. The patch created has the uuid P1.

Our manufacturer has given the platform enough information for the moment. The platform now starts to accept registration of device own by citizens. A Citizen already using the Cyber Trust



UUID :
D1

Name :
MyPhone

Pretty Name :
My phone

EUCitizenID :
1234

Cpe :
cpe-189402097

RiskLevel :
0

DeviceClassUUID :
DC1

OrganizationUUID :
O2

SUBMIT

Figure 3-4: A citizen register his device on the platform

platform is interested in having its Good Phone monitored. So, he/she connects to the platform and click on add a new device. He/she fills a form.

The form here abstract a few things. In the final version end user won't have to refer uuids directly but more likely he/she will have to choose the name of the company and type of device out of a list in a html form. RiskLevel and Customer-premises equipment (CPE) will be deduced from the device information provided by Smart Device Agent (A03m, A05m, A08m, A12, A14). Basically he/she will just have to refer a name for the device to be able to dissociate it from its other device especially if he/she own multiple device of the same class. The device has the uuid D1.

Then the Smart Device Agent (A03m, A05m, A08m, A12, A14) installed on the user device sent the current version of Operating System the device is running on. The Smart Device Agent is doing the task to find the current version of operating system as it has access to the system and this process is less error prone if it is done by a software rather than a human.

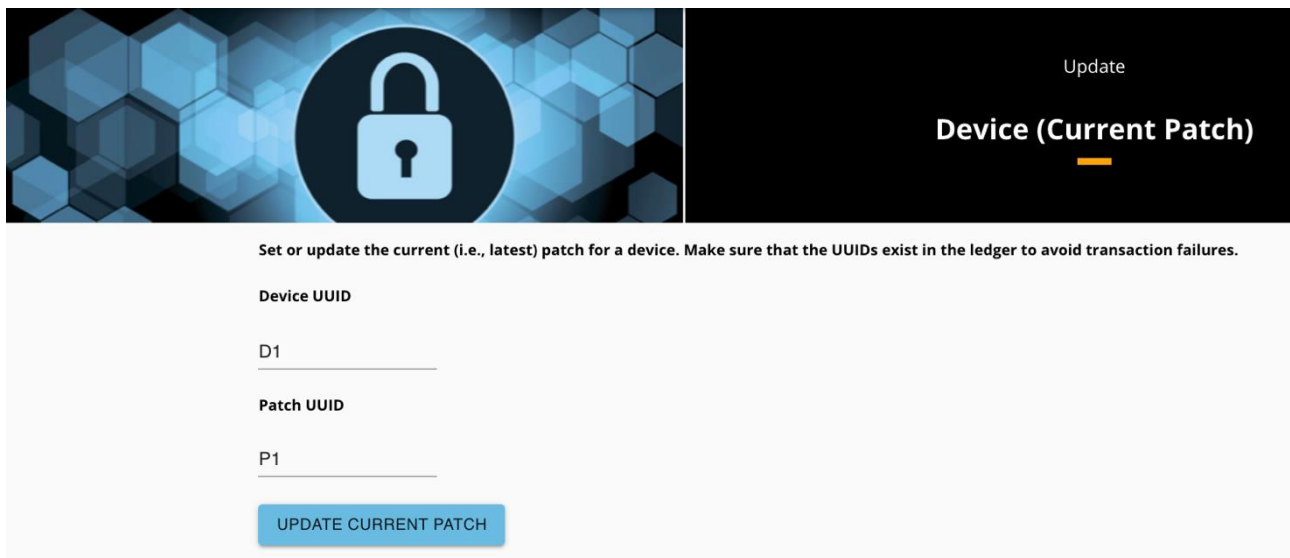


Figure 3-5: A patch of the device is being set

The Device D1 is running on the only patch published by its manufacturer which is P1. We have successfully created a network of the actors of our environment. Here is a schema of our environment.

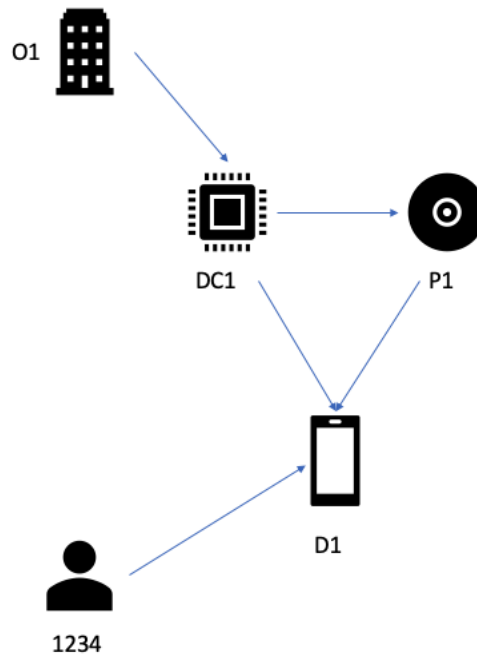


Figure 3-6: Network of actors

As you may see every actor we have created are connected together in the above schema. This is why we can say that we have created a network of actors.

Now for the purpose of the scenario let's imagine that the patch previously created is found malicious by the Crawling service (A10). A10 updates the patch through an API call and P1 status is now set to vulnerable. O1 is then alerted that one of its patches is vulnerable and EU Citizen 1234 is alerted there is a vulnerability on its device.

O1 publish a new patch P2 to solve the security breach. Smart Device Agent (A03m, A05m, A08m, A12, A14) alert the user via a notification that a new update is available. The user consent to perform the update. Once the update completed the DLT is updated via a last call to our API for this scenario.

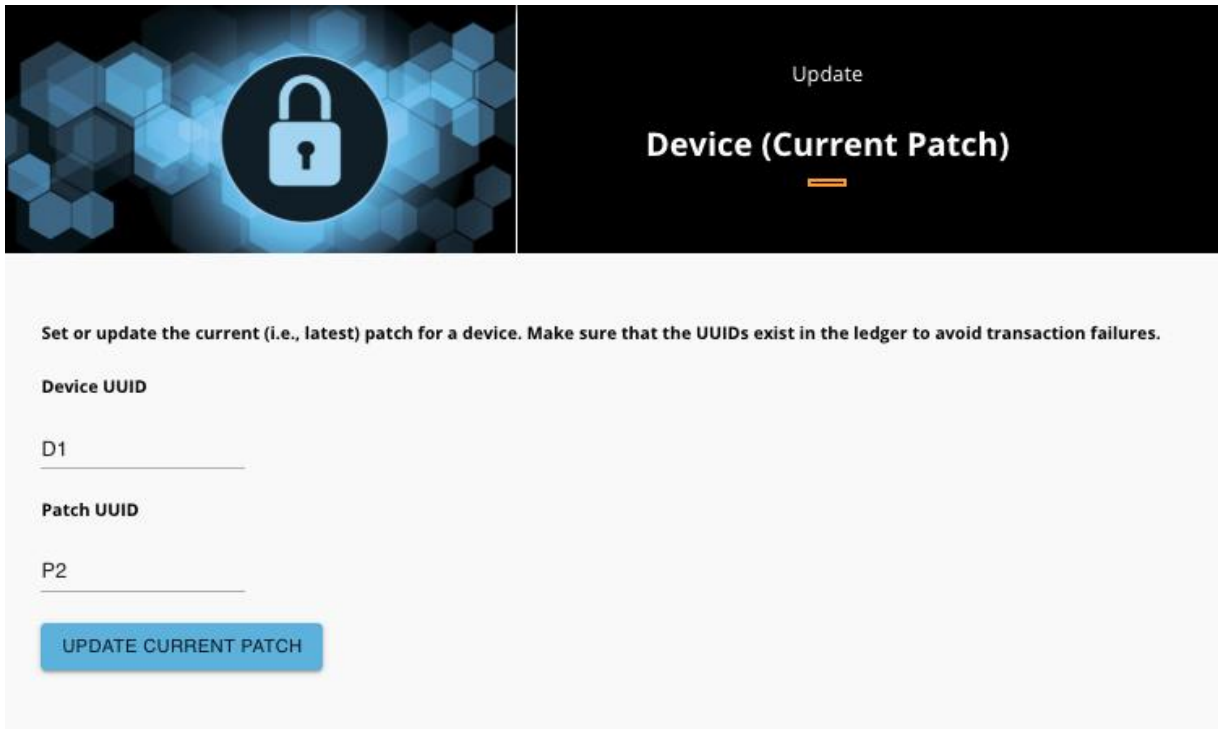


Figure 3-7: After the incident the device is updated to a new version of OS

D1 is now running on P2. Now let's have a look again at our network of actors.

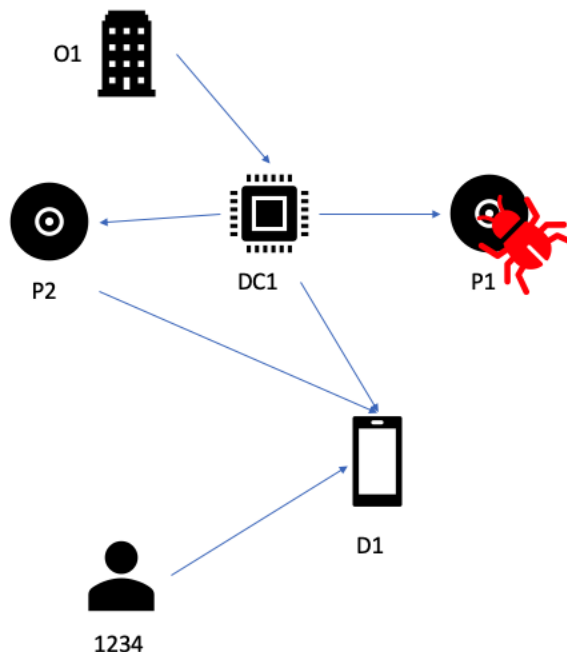


Figure 3-8: Updated network of actors

P1 is set at vulnerable and a new patch P2 is available for the device. The has updated itself as the user was alerted of the availability of a new version of the firmware for its device.

In this scenario we saw how the Cyber Trust components and the DLT will solve the problem of integrity of the device by detecting vulnerabilities in firmware

4. Conclusion

This document concludes the first iteration of the DLT development according to the architecture we previously depicted in D7.2. With this iteration, we are able to create the network of actors in the IoT environment, ensuring their integrity through a trusted filesystem upgrade. Our next step to fulfill the rest of the end user requirements for the DLT is to work now on the forensic evidence aspect. This step is dedicated to the LEA to help them during their investigation.