GOEASY
GalileO-based trustEd Applications for health and SustainabilitY

D3.3 – Initial Local Trust Features for Devices, SDKs and Applications

Deliverable ID	**D3.3**
Deliverable Title	**Local Trust Features for Devices, SDKs and Applications**
Work Package	**WP3**
Dissemination Level	**PUBLIC**
Version	**1.0**
Date	**2019-06-20**
Status	**Peer review**
Lead Editor	**BQ**
Main Contributors	**ISMB, CNET, GRENAPES**

Published by the GOEASY Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 776261.

Document History

Version	Date	Author(s)	Description
0.1	2019-05-24	BQ	First Draft with TOC
1.0	2019-06-19	BQ	Ready for peer review
1.0	2019-06-20	BQ	Final version of the document
1.1	2019-12-04	BQ	Final version reviewed

Table of Contents

Introduction

As stated, Qualcomm software stack for current platforms in the market does not support location RAW data handling and parsing. As a solution agreed with the rest of participants on GoEasy project, BQ has redefined their own roles in order to validate location provided by standard Android APIs as long as the trustability and integrity of the device being used.
For that purpose, BQ has developed the following three APIs for APPs development purpose, being their share through a java library to the rest of participants of the project:

(API1) Trusted devices. This API provides the needed functions to check the device has not been compromised. The next security checks are performed:

1. Rooted system
2. Fake position option enabled inside developer settings
3. Officially released fingerprint.
4. Unlocked bootloader

(API2) Cell Tower. This API ensures the location provided by the device is located within the action radio of the network antenna the device is connected to.

As agreed jointly with all GoEasy players in order to give support for the rest of developers, in D3.3 review, it is being described one **API3** to get location RAW data. As it has been confirmed in the beginning of this introduction this API3 will not be able to be used at this moment since there is no Qualcomm device in the market supporting RAW data.
This document describes the purpose and use of each one of these APIs, as long as the technical details of its solution definition and implementation.
Due to problems into the Qualcomm architecture, they don't give support to be able to get the RAW data from location chip. It doesn't exist right now any device from Qualcomm with RAW data available since this code part has not been published. From BQ, we try to contact with Qualcomm but it has been impossible to get support to use the RAW data.
For this reason, in a meeting held with the other members of GoEasy, the BQ role has been redefined in order to develop the necessary tools to validate the location data and ensure they are correct. To do this BQ has developed the APIs described in the following.
In the D3.3 review, it is described one API3 to get the RAW data, but as it has been argued in the beginning of this introduction this tool is not will be a viable option due to Qualcomm does not provide the necessary code, thus it was decided jointly with all the GoEasy players to BQ develop a new API for the rests of developers. This tool is the request by the keycloak server to get one verification token that the app will use to send data to secure servers.

1.1 Scope

The objective of this Task 3.3 is to provide the *AsthmaWatch* and *Apes Mobility* applications with the necessary tools to verify and guarantee that the terminals meet integrity requirements needed for the correct interpretation of location data given by the device, i.e. that the device is perfectly located or that it has not been intentionally manipulated (rooted), and thus, be sure (especially for the case of the Apes Mobility app) that the user does not receive or enjoy benefits by misusing the terminal and, therefore, the application.
In the specific case of *Asthma Watch* application, the citizens can obtain benefits by using this application daily during their usual journeys, avoiding the most polluted areas of the city. To do this, relying on the GALILEO signal, the app adjusts the established route according to the points of greatest contamination, making possible a smooth journey.
On the other hand, the *Apes Mobility* application uses the signal of GALILEO with the purpose of using it to certify and guarantee that the user really is at the point he claims to be, avoiding signal manipulation to benefit inappropriately from the advantages offered by the application.

1.2 Related documents

ID	Title	Reference	Version	Date
[RD.1]	D3.2 Initial algorithms for Multi-constellation and Authentication			
[RD.2]	D3.6 Final Local Trust for features Devices, SDKs and Applications			

2 Trusted Device Library

The library developed by BQ has the objective of providing any developer of mobile applications related to the GOEASY platform the necessary tools and knowledge about the location and its integrity and reliability. This library can be imported in any Android Application and exports three APIs that developers can use as long with standard Android API.

2.1 API 1: TRUSTED DEVICE TOOL

```
/**
 * Returns if the device is trusted:
 *  - is not a root device
 *  - is not enable fake location
 *  - is not the bootloader unlocked
 *  - the fingerprint has release-key
 *
 * @return True if the device is trusted
 */
fun isTrustedDevice() : Boolean {
    return (!isDeviceRooted() && !isFakeLocationEnabled() && !isBootloaderUnlocked()
            && isFingerprintRelease())
}
```

Returns	
boolean	True if the device is trusted

The first API (API I), named **Trusted Device**, is the part of the library which checks if the application is running over a secure device. It performs the next integrity and security validations:

- **Is the bootloader locked?**
  - The bootloader is the manager of Android boot (or the manager of another operating system). It is the part of the software which is in charge of performing several checks before the system starts. Is also the region of the software that is responsible for performing various tests before the boot of the operating system can start, as well as giving instructions to the OS so that the boot is carried out without problems. It is also the software stack holding fastboot protocol, the usb interface to allow flashing the device. It means, if the bootloader is unlocked, there are high possibilities that the operating system has been manipulated.
- **Is the device rooted?**
  - The routing of Android devices is the process through which a malware gets granted with superuser permissions and privilege access.
  - This API ensures there is no application installed in the system with superuser permission granted. It also ensures *sudo* and *su* commands are not available through android console.
- **Is the Fake Location developer option disabled?**
  - Android Developer Settings option that allows applications to modify the location provided by Android Framework on the Android API, so they can "cheat" the rest of applications. These

applications require special permissions from Android ***(ACCESS_MOCK_LOCATION).***

This APIs ensures *fake location* settings is not enabled. It also ensures there is no application installed in the system with ***ACCESS_MOCK_LOCATION*** **granted.**

- **Has the device a officially released fingerprint?**
  - Android build fingerprint is a unique identifier for each compiled firmware. It contains several information like the date of building, compilation type (user, userdebug, eng), and so more.

    This APIs ensures device's fingerprint is a release one.

2.2 API 2: CellTowerValidation Tool

fun validateCellTower (location : Location, callback: CellTowerCallback) {...}

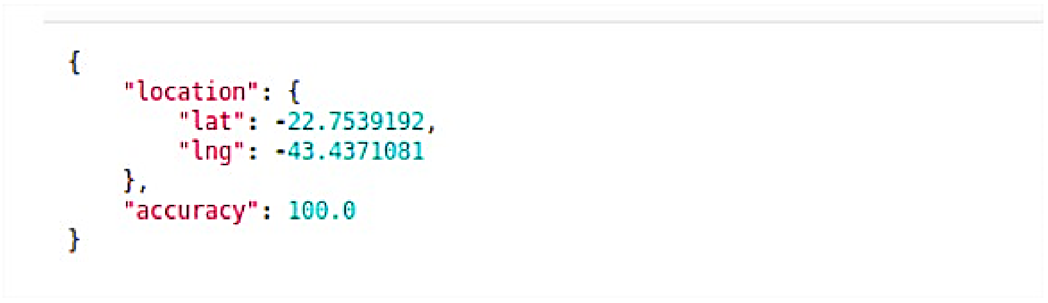
Parameters	
Location location	GNSS location provided by the device
CellTowerCallback callback	Callback use for asynchronous returning of the result

Returns	
boolean	True if GNSS location is inside cell tower action ratio

The second API, API 2 or **Cell Tower Validation** is the part of the library which is qualified for matching the position obtained by the application from the device with the network antenna to which the device is connected. For that purpose, we use the **Mozilla BD Project** API *(Mozilla Location Service) https://location.services.mozilla.com/api*, which contains information regarding carrier operators antenna all over the world, including their ID and location coordinates (reference)
Mozilla Location Service is an Open-Source project that provide network antenna information. It provides information about cell sites and WiFi access point. In February 2019 MLS had collected more than 44.43 million unique cell network.
Geolocation requests are sent using one post request with a JSON body.

```
class CellTowerInfo(
    var radioType : String?,
    var mobileCountryCode : Int?,
    var mobileNetworkCode : Int?,
    var locationAreaCode : Int?,
    var cellId : Int?,
    var psc : Int?
)
```

An example of Mozilla API returning the information related to the network antenna.



```
{
    "location": {
        "lat": -22.7539192,
        "lng": -43.4371081
    },
    "accuracy": 100.0
}
```

For location accuracy this API uses accuracy provided by Mozilla database (some dozens of meters in city antenna; a few kilometers in field antennas). The distance to the antenna is calculated following a straight line.

Ongoing improvement. Android API provides not just the carrier antenna the device is connected to, it also lists every antenna the radio of the device is able to talk with. BQ is extending current API to allow the use of multiple (3) network towers in order to get a better location triangulation.

Future improvements. For better accuracy, this API may be extended with a new parameter that allows the developer to pre-set the maximum accepted ratio. In case the distance between the GNSS location provided by the device and the antenna's location exceeds either the accuracy fixed by the APP or the accuracy returned by Mozilla API, this function will return *false*.

2.3 API 3: Keycloak Android Tool

Returns	
string token	Token the APP uses to authenticate itself against Serengeti server

Previously known as GNSSRawData API, this API returns to the application the information about the raw data and the constellation. The access to location RAW data relied entirely on the device being used. For that reason, the Consortium decided that the BQ Library should have a new tool that to help different developers who will use the Goeasy platform. This tool is a request for the Keycloak server (a Goeasy server which generates validation token to preserve the security of the platform) to get a token that the app will use to send the information to Serengeti (see deliverable D4.2).

```
fun obtainToken (keycloakCallback: KeycloakCallback) {
    ApiServiceInterfaceKeycloak.create().getKeycloakToken(Constants.CLIENT_ID,
            Constants.USERNAME,  Constants.PASSWORD, Constants.GRANT_TYPE, Constants.CLIENT_SECRET)
            .enqueue(object : Callback<KeycloakResponse> {
        override fun onFailure(call: Call<KeycloakResponse>, t: Throwable) {
            LogcatProxy.e("[onFailure] -> ${t.localizedMessage}")
            keycloakCallback.onFailure(CustomGoeasyException(CustomGoeasyException.KEYCLOAK_EXCEPTION))
        }

        override fun onResponse(call: Call<KeycloakResponse>, response: Response<KeycloakResponse>) {
            if(response.isSuccessful){
                LogcatProxy.d("[OnResponse] ${response.body()?.access_token}")
                response.body()?.access_token?.let { keycloakCallback.onKeycloakTokenResponse(it) }
            } else {
                LogcatProxy.d("[OnResponse] ${response.code()}")
                LogcatProxy.d("[OnResponse] ${response.errorBody()!!.source()}")
                keycloakCallback.onFailure(CustomGoeasyException(CustomGoeasyException.KEYCLOAK_EXCEPTION))
            }
        }
    })
}
```

The application just needs to call to this method of the BQLibrary, and a token will be returned.

Conclusions

In order to assure the correct behaviour of the devices GoEasy applications are running onto, and not to reward the users with bonuses they do not deserve, the GOEASY platform has developed a series of APIs and features that will ensure the integrity of the positioning system to authenticate the signal processed by the Android OS through the apps.
The updated version of this deliverable (D3.6) will include the results of the tests performed on the latest version of the apps and APIs.

Acronyms

Acronym	Explanation
GNSS	Global Navigation Satellite System
API	Application Programming Interface
SDK	Software Developer Kit
OS	Operating System
JSON	JavaScript Object Notation

List of figures