A Methodological Approach for the Security Analysis of FIWARE Technology

Juan Pablo Perata
Facultad de Ingeniería, Universidad de la República,
Montevideo, Uruguay, 11300
jpperata@gmail.com

and

Gustavo Betarte
InCo, Facultad de Ingeniería, Universidad de la República,
Montevideo, Uruguay, 11300
gustun@fing.edu.uy

Abstract

This paper presents the results of a security assessment of FIWARE technology. We adopted an offensive perspective to identify vulnerabilities in deploying FIWARE components in specific architecture configurations. We identify security issues by experimenting in a locally controlled environment and propose a threat model following the OWASP methodology. We implemented attacks for three of the identified attack goals and validated our approach with an exploratory analysis of an actual working and productive FIWARE platform. This analysis helped us distinguish different types of attacks, and we ended up with recommendations for components, architecture, and access control.

Keywords: FIWARE, security assessment, threat analysis.

1 Introduction

FIWARE [1] is a platform created in 2011 through a partnership between the European Commission and European Industry. This open-source platform provides an innovation ecosystem that enables the creation of applications and services in the cloud for various purposes. This design decision makes the platform particularly useful in smart cities as it ensures interoperability between the solutions developed and allows them to be implemented in multiple cities.

The development of technology for smart cities presents various challenges in terms of security and privacy. The smart city platform plays a crucial role in protecting these elements. This research aims to answer some critical questions regarding the security of the FIWARE platform. Are there any known vulnerabilities in the FIWARE platform? What is the current security status of the FIWARE components? What security requirements does the platform impose? Is it possible to deploy an insecure FIWARE solution?

We have reviewed past research and official documentation on FIWARE to **understand and identify challenges and security issues in its components** and those related to deploying FIWARE architectures in various real scenarios described in the literature. Additionally, FIWARE provides sample architectures that describe a fictitious context reality, which we used to experiment and better understand.

One of the main results of our work is a threat model on a referential platform that includes core FIWARE components and involves several artifacts. The methodology used to model threats in the referential FIWARE platform is based on the OWASP approach to threat modeling (see [2]), which proposes a methodological framework to identify, quantify, and address associated security risks from an offensive perspective. In our work, the focus of the modeling process is on the following three steps: i) **decomposition of the application from the perspective of a potential attacker**, ii) **identification of threats with STRIDE** (see [3]), which is a methodology developed by Microsoft that helps distinguish threats by classifying them into 6 (six) categories, and iii) **exploration**

of an attacker's objectives in critical assets. Some identified objectives have also been exploited, experimenting with them in a locally controlled environment.

We validate our approach by performing an exploratory analysis of an actual working and productive FIWARE platform. That analysis led to the identification of potential attacks and a set of recommendations to improve the security of various platform components.

The structure of the rest of the article is as follows: Section 2 presents background and related work. Section 3 describes the architecture of the FIWARE platform. Section 4 exposes the main security issues in platforms containing any of the central components of FIWARE architecture. Section 5 describes the threat modeling and attack analysis performed in the referential scenario. Finally, Section 6 outlines the conclusions and reflects on future work opportunities.

This article is an extended version of [4]. The contents of Section 3 and the detailed description of the security issues discussed in Section 4 are new material.

2 Background and related work

One of the first calls for attention regarding the platform's security under analysis was informed in [5], where a race condition bug was reported in the Orion Context Broker component that caused notifications to be sent to an unintended endpoint rather than the expected one. This bug was solved in 2014 by the FIWARE team.

In [6], the authors use FIWARE as a base platform in the deployment of a remote patient monitoring solution, where their main contribution focuses on architectural and design aspects using FIWARE components called Generic Enablers (GE). that facilitate communication with different IoT devices and user interfaces. Although the focal point of the study is on functional aspects, it highlights the importance of security, particularly confidentiality. Briefly describes the implemented security layer composed of three components: Identity Management GE (IdM) that provides authentication mechanisms; Policy Enforcement Point (PEP) responsible for access control to resources on the platform; and Policy Decision Point (PDP) that interacts directly with the IdM and PEP to provide an authorization decision (allow or deny). Each of these components has its reference to FIWARE implementation.

The reference [7] presents the result of performing a FIWARE security analysis focused on access control and confidentiality. The authors state that despite being a robust platform, some components have yet to develop security mechanisms. In particular, they highlight the absence of access control in certain operations of the IDAS 5 model and the need for NGSI communications encryption support for some IoT Agent implementations. IDAS is the FIWARE reference implementation of the GE Backend Device Management component, which is generally used in most scenarios in IoT architecture.

Two groups of researchers from different European universities in [8] and [9] present the implementation of authorization components based on FIWARE, integrating AuthZForce and KeyRock. The projects focus on functional aspects and are oriented towards a particular use case, in one case, a smart electricity grid, and in another, the sharing and permissions use of data in industrial ecosystems. Both provide high-level definitions of XACML policies for role-based access control.

In [10], a project is conducted to build a smart city platform using the FIWARE component stack, showing the challenges of incorporating access control mechanisms to platform components and IoT sensors.

In [11], the authors present an analysis indicating that FIWARE only supports authentication but not authorization for IoT devices, so the credentials of a compromised sensor could be used to simulate measurements coming from other sensors. On the other hand, the work also mentions security problems in the multi-tenancy model that could allow unauthorized modifications to another tenant.

Finally, [12] and [13] present a global analysis of security threats in smart cities from the point of view of devices, systems, and communication networks. We list threats related to this work: unauthorized access, identity spoofing, man-in-the-middle (MiTM) attacks, eavesdropping, alteration and falsification of messages, and outdated software.

Based on what was investigated, it is highlighted that very few publications address a security evaluation of the FIWARE components individually or as a whole in a given architecture.

3 The architecture of the FIWARE platform

In FIWARE, the term "context" refers to the status of a physical or conceptual object existing in the real world (see [14]). For instance, an entity could represent a room, and its temperature and pressure properties may be interesting. Sensors are installed in the room to provide sensing measurements and measure these properties in real time. The values of these properties are then translated into attributes of the entity. The context of an entity, such as "Room1", is determined by the values of its properties over some time. FIWARE offers various tools to produce, obtain, publish, and consume context information.

The FIWARE architecture has several main components, as shown in Fig. 1. The core component of this architecture is the Orion Context Broker, which is responsible for managing contexts. It provides a RESTful API that enables applications, services, and other components to consume or perform context switches, make configurations, and enrich their visualizations through dashboards. For data storage, the Orion Context Broker uses a NoSQL MongoDB database.

There are two types of entities in Orion: context-producing entities and context-consuming entities. The former includes IoT sensors or external context providers (through APIs) associated with existing entities in Orion. The latter includes applications, other components, or entities that subscribe to context changes and are notified by Orion when certain configured conditions are met.

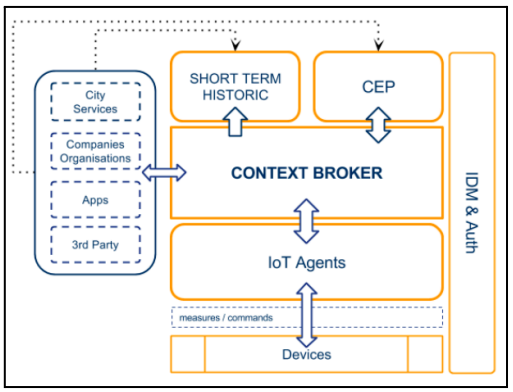

Figure 1. Main components of a FIWARE architecture (see [15])

IoT Agents constitute a means of abstraction of the communication protocols used by IoT devices, acting as a proxy in sending sensing measurements and commands between the devices and Orion. This abstraction layer as a whole is called Integrated Data Acquisition System (IDAS) (see [16]) and is the reference implementation of the GE IoT Backend Device Management. They also use a NoSQL MongoDB database for data storage.

The IoT Agents provide two access ports called **north port** and **south port** (see [17]). The north port offers an HTTP REST API and is used to perform configurations and communication with the Context Broker. The south port is responsible for all reception of measurements from sensors.

On the other hand, the storage of historical context information is carried out through the Short Term Historical (STH) component (see [19]). In addition, it is possible to add a Complex Event Processing (CEP) module (see [18]) to analyse event data in real time and be able to configure rules that react to certain conditions.

Across all modules there is an identity management and access control layer that can be configured to establish component access protection mechanisms.

4 Security issues of core FIWARE components

Having a basic knowledge of the FIWARE components and having carried out proofs of concept interaction and observation of the behaviour of several of its modules, the next step led to a deeper analysis of potential security issues. We considered deployments of fictitious FIWARE platforms based on existing tutorials provided by FIWARE, which contain core components of any FIWARE architecture: Orion and IoT Agent.

The main security issues identified in the FIWARE platform that arise from a background review, investigation of official documentation, and interaction with components in a local controlled environment are listed below. The study focused on the core components Orion, IoT Agent and their interactions.

4.1 TLS encrypted communications not required

The use of TLS encrypted communications between components is not required, making it possible to inspect and/or alter data in transit. This includes interactions between Orion←→MongoDB, IoT Agent←→MongoDB, IoT Agent←→IoT devices and also direct interactions with Orion, IoT Agent and MongoDB. This behaviour can be observed in Section 5, subsection 5.2 where attack goals are achieved.

4.2 MongoDB authentication not required

Authentication with MongoDB is not required by default between Orion←→MongoDB and IoT Agent←→MongoDB. In a deployment of a FIWARE platform where authentication with MongoDB is not demanded, this could enable unauthorised anonymous modifications. This behaviour can be observed in Section 5, subsection 5.2 where the running instance of MongoDB is accessed.

4.3 Authorization not required by Orion and IoT Agent

By default, Orion and IoT Agent components do not offer their own access control mechanisms. Additional Fiware modules can be provided to protect their access.

4.4 Multi-tenancy model could enable unauthorised modifications in other tenants

The multi-tenancy model in Fiware is established through a logical separation of databases in MongoDB. To do this, a special header "Fiware-Service"[1] is used that allows the referenced tenant to be indicated in each HTTP request (see [20]). In the background chapter, a publication at the beginning of 2020 (see [11]) made visible security issues in the FIWARE multi-tenancy model that could allow alterations to be made in other tenants, even in non-existent tenants. The investigation emphasized that in a platform where the Keyrock component was added for authorization capabilities, establishing access control mechanisms would not be possible based on the "Fiware-Service" header. Some proof of concept tests were carried out to validate these behaviours, reaching the following preliminary results.

4.4.1 Arbitrary tenant creation by specifying "Fiware-Service" HTTP header

Assuming a scenario with no access control mechanisms deployed, if an HTTP request contains the header "Fiware-Service" with a value "X" and the tenant does not exist, Orion creates a database "orion-X". Below are two examples of an entity created through Orion in a legitimate tenant with the name "tenant_1" in the first instance. Then another entity is created in a non-existent tenant with the name "nonexistent".

```
curl -iX POST \
  '<http|https>://<orion-host>:<orion-port>/v2/entities' \
  -H 'Content-Type: application/json' \
  -H 'Fiware-Service: tenant_1' \
  -H 'Fiware-Servicepath: /' \
  -d '
{
    "id": "urn:ngsi-ld:Store:008",
    "type": "Store",
    [...],
    "name": {
        "type": "Text",
        "value": "Bösebrücke Einkauf"
    }
}'
```

It can be seen in the MongoDB database engine that the newly created entity is stored in the database associated with **tenant_1**:

```
> show databases;
orion           0.000GB
```

[1] https://fiware-orion.readthedocs.io/en/latest/devel/httpHeaders.html#17-fiware-service

```
orion-tenant_1  0.000GB
> use orion-tenant_1
> db['entities'].find()
{ "_id" : { "id" : "urn:ngsi-ld:Store:008", "type" : "Store", "servicePath" : "/" }, "attrNames" : [
"address", "location", "name" ], [...], "name" : { "type" : "Text", "creDate" : 1644242378.4697275,
"modDate" : 1644242378.4697275, "value" : "Bösebrücke Einkauf", "mdNames" : [ ] } }, [...] }
```

Similarly, the creation of another entity but for a non-existent tenant:

```
curl -iX POST \
  '<http|https>://<orion-host>:<orion-port>/v2/entities' \
  -H 'Content-Type: application/json' \
  -H 'Fiware-Service: nonexistent' \
  -H 'Fiware-Servicepath: /' \
  -d '
{
    "id": "urn:ngsi-ld:Store:045",
    "type": "Store",
    [...]
    "name": {
        "type": "Text",
        "value": "Store Y"
    }
}'

> show databases;
orion           0.000GB
orion-tenant_1  0.000GB
orion-nonexistent  0.000GB
> use orion-nonexistent
> db['entities'].find()
{ "_id" : { "id" : "urn:ngsi-ld:Store:045", "type" : "Store", "servicePath" : "/" }, "attrNames" : [
"address", "location", "name" ], "attrs" : [...] "name" : { "type" : "Text", "creDate" :
1644243022.322857, "modDate" : 1644243022.322857, "value" : "Store Y", "mdNames" : [ ] } }, [...] }
```

4.4.2 "Fiware-Service" header support in Fiware authorization rules

The Keyrock component adds the ability to incorporate the "Fiware-Service" header in authorization rules in August 2020. This would indicate that versions before 7.9.2 do not have this feature.

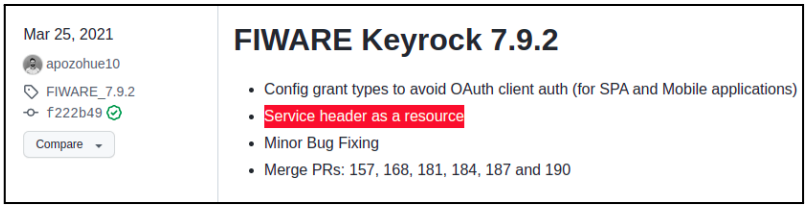

Figure 2. Details of the release version of FIWARE IdM Keyrock 7.9.2[2]

Figure 3. Support of Fiware-Service header in permissions in FIWARE IdM Keyrock 7.9.2[3]

[2] https://github.com/ging/fiware-idm/releases/tag/FIWARE_7.9.2
[3] https://github.com/ging/fiware-idm/blob/20420b045f11dde29673a6d1fff8c88ece05c524/controllers/oauth2/oauth2.js

4.5 IoT Agents receive measurements from unregistered devices

The huge impact of this issue is availability: sending large measurements in bulk could cause malfunctions in the IoT Agent or Orion, unavailability of services, or exhaustion of storage resources in MongoDB. We conducted a little investigation that supports this finding.

For IoT devices to communicate their sensing measurements to IoT Agents, they must first configure a group of devices and then register each device individually. Simplifying, below there is an example of the creation of a service provisioning group, interacting directly with the north port of the IoT Agent with the Ultralight protocol (a simple text-based protocol), where the HTTP body is a JSON object specifying the <API-KEY> that IoT devices will use when sending their measurements to the following endpoint <http|https>://<iot-agent-host>:<iot-agent-**south**-port>**/iot/d**. It also indicates the URL of the context broker that will receive these measurements (cbroker parameter), and the type of entity for unrecognised devices (entity_type parameter), which in this case is "Thing".

```
curl -iX POST \
  '<http|https>://<iot-agent-host>:<iot-agent-north-port>/iot/services' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "services": [
   {
     "apikey":      "<API-KEY>",
     "cbroker":     "<http|https>://<orion-host>:<orion-port>",
     "entity_type": "Thing",
     "resource":    "/iot/d"
   }
 ]
}'
```

Then, the registration of a new sensor is carried out, interacting in a similar way with the north port of the IoT Agent:

```
 curl -iX POST \
  '<http|https>://<iot-agent-host>:<iot-agent-north-port>/iot/devices' \
  -H 'Content-Type: application/json' \
  -H 'fiware-service: openiot' \
  -H 'fiware-servicepath: /' \
  -d '{
 "devices": [
   {
     "device_id":   "motion001",
     "entity_name": "urn:ngsi-ld:Motion:001",
     "entity_type": "Motion",
     "timezone":    "Europe/Berlin",
     "attributes": [
       { "object_id": "c", "name": "count", "type": "Integer" }
     ],
     "static_attributes": [
       { "name":"refStore", "type": "Relationship", "value": "urn:ngsi-ld:Store:001"}
     ]
   }
 ]
}
'
```

The recently created sensor will communicate its measurement by sending a POST request to the endpoint: "<http|https>://<iot-agent-host>:<iot-agent-sourth-port>/iot/d?i=motion001&k=<API-KEY>". For example:

```
curl -iX POST \
'<http|https>://<iot-agent-host>:<iot-agent-sourth-port>/iot/d?k=<API-KEY>&i=motion001' \
  -H 'Content-Type: text/plain' \
  -d 'c|1'
```

Simulating measurements from sensors previously not registered or non-existent on the platform is possible, being the only required thing a valid previously registered <API-KEY> (parameter k on the request). The i parameter (denoted by the device_id) can be an arbitrary string and the measurement received will be sent to the context broker and tagged as being sent by an entity of type “Thing”. A scenario where an attacker can intercept communications between IoT devices and IoT Agents with no TLS in place (see subsection 4.1) would be able to capture a valid <API-KEY> and send false measurements from registered sensors or non-existing ones. A large-scale measurement submission could cause the IoT Agent or Orion to malfunction, cause a denial of service, or cause storage resource exhaustion in MongoDB.

It is worth mentioning this behavior was visible with IoT Agent Ultralight version 1.16.2; probably earlier versions and those of IoT Agents handling other native protocols are affected too. Starting with version 1.17.0, a validation of the device identifier is in place, resulting in an HTTP 422 Unprocessable Entity error when the device has not been previously registered. However, it was observed that the entity is also created in the Orion database but without the sensing metric and there is also a reference to the non-existent device in the IoT Agent Ultralight database.

4.6 Subscribing to changes in context may allow context data to be disclosed if no access control mechanisms are settle up

Orion allows applications to subscribe to context changes and be informed when variations occur. Subscription creation supports an arbitrary URL and conditions that have to be met (for example in terms of attribute values and entity types) for the notification event to fire. When all conditions for a subscription are satisfied, Orion creates a synchronous HTTP POST request to the specified URL updating the data in JSON format.

In this sense, the operations of creating, updating and listing subscriptions may involve disclosure of sensitive information (context data and existing subscribed applications), if their access is not controlled or the access control mechanisms are permissive.

Assuming a scenario in which there are no access control mechanisms in the creation of subscriptions, an attacker can interact directly with Orion by creating a subscription specifying a URL of their domain to be notified when any entity changes (pattern .* will match everything):

```
curl -iX POST \
  --url '<http|https>://<orion-host>:<orion-port>/v2/subscriptions' \
  --header 'content-type: application/json' \
  --data '{
  "description": "Notify me of all entity changes",
  "subject": {
    "entities": [
      {
        "idPattern": ".*"
      }
    ]
  },
  "notification": {
    "http": {
      "url": "<attacker-controlled-url>"
    }
  }
}'
```

For example, assuming an attacker created a subscription to be notified of any context change, if an attribute of an entity is updated as below, a POST request will be sent by Orion to the attacker controlled server:

```
curl -iX PUT \
  --url '<http|https>://<orion-host>:<orion-port>/v2/entities/urn:ngsi-ld:Product:001/attrs/price/value' \
  --header 'Content-Type: text/plain' \
  --data 89
```

A notification is received at the attacker server with the contents of the modified entity:

```
POST / HTTP/1.1
Host: <attacker-controlled-url>
User-Agent: orion/3.3.1 libcurl/7.61.1
```

```
Fiware-Servicepath: /
Accept: application/json
Content-Length: 257
Content-Type: application/json; charset=utf-8
Fiware-Correlator: 7280eac8-86bc-11ec-9d24-0242ac120103; cbnotif=1
Ngsiv2-AttrsFormat: normalized

{"subscriptionId":"61fed38a99f48b3f741c055e","data":[{"id":"urn:ngsi-ld:Product:001","type":"Product","nam
e":{"type":"Text","value":"Beer","metadata":{}},"price":{"type":"Integer","value":78,"metadata":{}},"size"
:{"type":"Text","value":"S","metadata":{}}}]}
```

4.7 Versions of the iotagent-node-lib library lower than 2.12.0 do not support authentication with MongoDB

iotagent-node-lib is a Node.js library provided by FIWARE to serve as a base for custom IoT Agent implementations. Reviewing the commit history in github, it could be seen that MongoDB authentication support was added in April 2020.

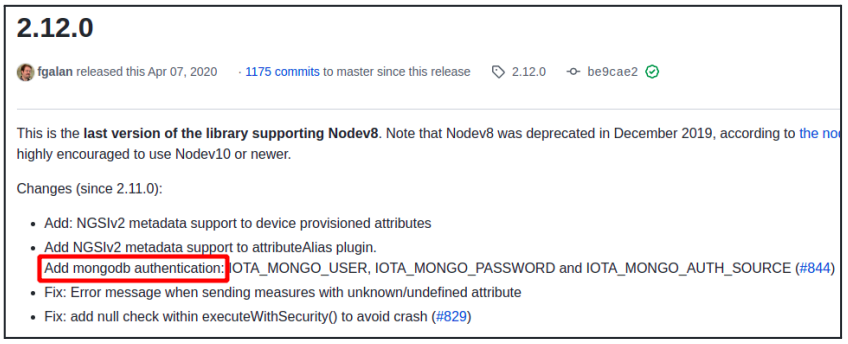

Figure 4. Details of the release version of 2.12.0 of iotagent-node-lib[4]

4.8 Versions of the iotagent-node-lib library lower than 2.13.0 do not support TLS communications with MongoDB

Reviewing the commit history in github, it could be seen that TLS support with MongoDB was added in version in September 2020.

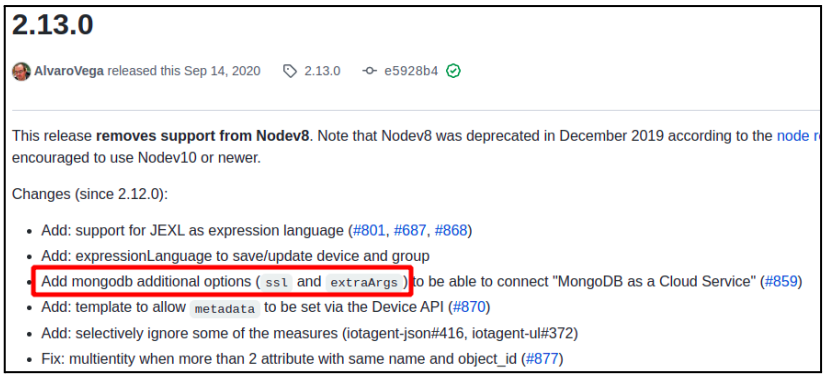

Figure 5. Details of the release version of 2.13.0 of iotagent-node-lib[5]

[4] https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.12.0
[5] https://github.com/telefonicaid/iotagent-node-lib/releases/tag/2.13.0

4.9 Orion versions lower than 2.3.0 do not support TLS with MongoDB.

Reviewing the commit history in github, it could be seen that TLS support with MongoDB was added in November 2019.

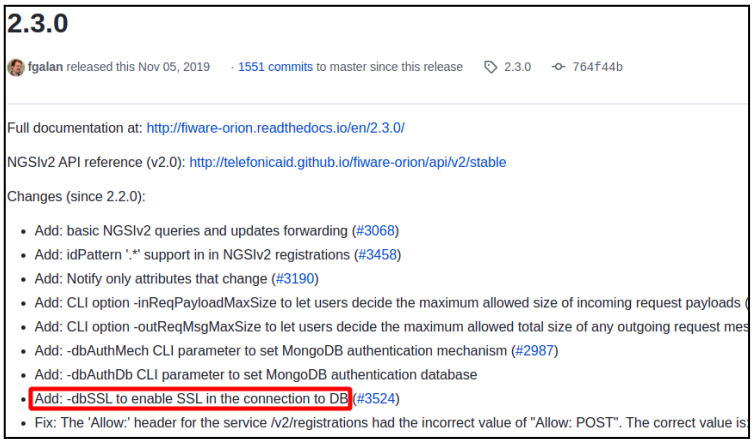

Figure 6. Details of the release version of 2.3.0 of Orion[6]

4.10 Orion does not support TLS certificate verification with MongoDB

To date, Orion does not support verification of the path certificate in TLS communications with MongoDB in any of its versions. This means that Orion's implementation using the mongodb connector does not provide support for validating the certificate offered by the MongoDB server if encrypted communications are used.

It was noted that in version 3.0.0 of April 2021, a change was introduced that accepts the use of invalid certificates through the option **tlsAllowInvalidCertificates=true** in the mongodb connector used by Orion. As indicated in the official MongoDB documentation, it is strongly recommended to avoid using this property. This could allow MiTM attacks between Orion and MongoDB to go undetected.

Figure 7. Details of the release version of 3.0.0 of Orion[7]

[6] https://github.com/telefonicaid/fiware-orion/releases/tag/2.3.0
[7] https://github.com/telefonicaid/fiware-orion/releases/tag/3.0.0

Figure 8. Option tlsAllowInvalidCertificates=true in MongoDB connection[8]

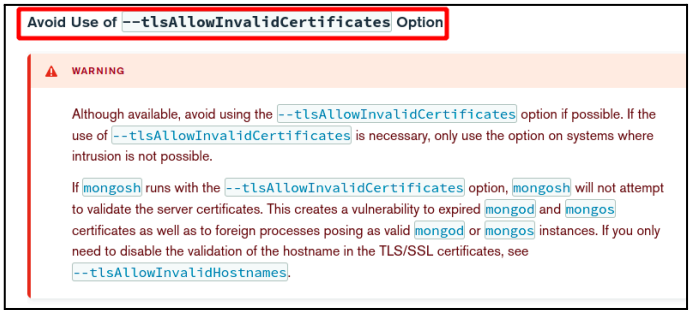

Figure 9. MongoDB recommendation in the use of the property tlsAllowInvalidCertificates[9]

4.11 Sensor accounts created through the FIWARE IdM Keyrock component and used by IoT devices, only provide support for authentication but not authorisation

An attack scenario is proposed in which if the credentials of a sensor were compromised, they could be used as seen in subsection 4.5 to simulate measurements coming from other sensors, even generate measurements from non-existent sensors.

To summarise, the security issues detailed allowed us to establish a primary general security state of the central components of FIWARE, studied at an architectural level. This provided elements of interest for the beginning of the threat modelling stage in Section 5.

5 Threat modelling and attack analysis of FIWARE technology

FIWARE provides guidelines for possible reference architectures to be considered Powered-By-Fiware. However, it does not require adhering to fixed deployment structures, the only exception being using the Orion component. FIWARE tutorials model a fictional reality, providing a basis for experimentation and learning purposes. The documentation explicitly mentions that they are not prepared to be used “as is” in production installations, as they present several security flaws, such as clear text credentials, unencrypted communications, and lack of database authentication. On the other hand, it was challenging to find official documentation related to hardening guidelines regarding FIWARE components in production environments.

This section describes **the approach to model threats on a reference FIWARE platform**. The work presents a methodological approach based on the OWASP Threat Modelling process as already mentioned in Section 1. OWASP is a non-profit organization whose purpose is to improve application security by providing resources, tools, education and training, supported by a large community.

The methodology describes a series of activities for improving security by following these objectives from an offensive perspective: **identification, quantification and addressing of security threats** in these referential phases:

---

8 https://github.com/telefonicaid/fiware-orion/blob/4fbebd331e4bd6e27353b9660c29352d4ffc91a5/src/lib/mongoDriver/mongoConnectionPool.cpp#L392

9 https://docs.mongodb.com/manual/tutorial/configure-ssl-clients/#avoid-use-of---tlsallowinvalidcertificates-option

- **Phase 1: decompose the application to model the target in order to get a better understanding**, drawing data flow diagrams, identifying assets, recognizing entry points where a potential attacker could interact with the system and listing permissions associated with external actors.
- **Phase 2: identify threats using STRIDE categorization**, providing a map of potential threats and assumptions that can be checked.
- **Phase 3: define attacker objectives**, by analysing one or more threats taking place **affecting assets that pose a critical position** in the system.
- **Phase 4: design attack vectors** for all or for a subset of the identified attacker goals showing how these goals could be achieved by an attacker.
- **Phase 5: determine countermeasures and mitigation**, by defining a strategy for addressing the threats identified based on the business impact they present. Usually the evaluation results in accepting, eliminating, mitigating or transferring each risk.

Phases 1, 2, 3 and 4 are carried out in Subsections 5.1, 5.2 and 5.3 in a referential platform. Phase 5 is described as recommendations and guidance in Subsection 5.4 when doing the exploratory analysis of a real FIWARE platform, and takes advantage of all the work made in the referential platform.

5.1 Referential platform

The experimentation scenario shown in Fig. 10 is based on an existing FIWARE tutorial to understand the platform; this is a simple scenario that describes a stock management application for a supermarket chain. It consists of a simple deployment of microservices present in any FIWARE platform, composed of several modules: i) a central unit called **Orion Context Broker** responsible for the context life cycle of the information of an imaginary reality; provides a RESTful API through which applications, services, and other components can use to consume, perform context switches or perform configurations, ii) an artifact called **IoT Agent** in charge of allowing devices with their native protocols to communicate with Orion, iii) a NoSQL **MongoDB** engine in charge of the databases linked to these two previous elements, and iv) a set of **IoT devices** simulated by software, such as sensors (send measurements) and actuators (receive commands).

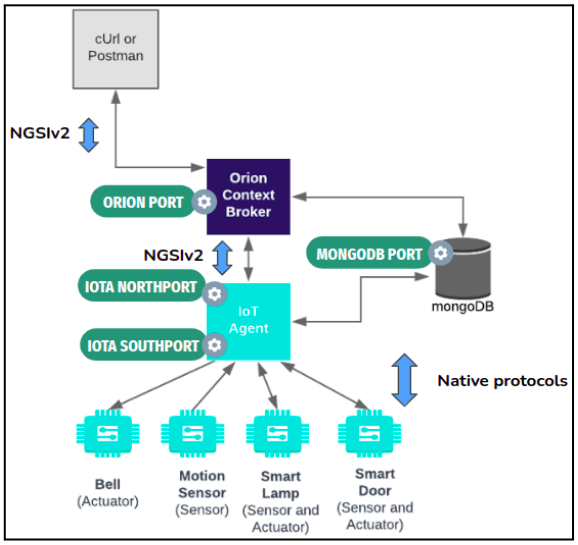

Figure 10: FIWARE referential platform [21]

Orion is a C++ implementation of the NGSIv2 REST API [22], based on the OMA-NGSI 9/10 specification [23], which provides a context data model around the concept of entities. Through HTTP requests, it is possible to create, view, modify, and delete entities and attributes related to said entities. Additionally, it provides capabilities to list, register, update, and delete externally sourced context providers and list, create, update, and delete subscriptions upon context changes. Clients can access the API through an HTTP port, which defaults to 1026.

The IoT Agent component exposes two ports, called “northport”, which will be used for configurations and communications received from the Context Broker, and “southport” to receive measurements from IoT devices, whose default values are 4041 and 7896, respectively.

Using the OWASP Threat Dragon tool [24], a visual representation of the platform in the form of a DFD is shown in Fig. 11, where the following components can be found: i) **Internal and external actors**; anonymous users or applications that interact directly with Orion, and administrators capable of making configurations in all components, ii) **Subsystems**; specified by Orion, IoT Agent and IoT devices, iii) **Data Stores**; illustrated by the MongoDB engine where information is stored in databases, iv) **Entry and exit points**; places where potential attackers can interact with the system and where data leaves it respectively: Orion HTTP port, north and south ports of the IoT agent, MongoDB port, interactive shells with Orion and the IoT agent, v) **Trust boundaries**; denoted by the interfaces defined through entry and exit points, vi) **Assets**; recognized as valuable resources for a potential attacker and that require protection: Orion, context data, IoT Agent, MongoDB engine and databases, configuration settings for each component, docker subsystem, identification and authentication credentials of administrative personnel, vii) **External dependencies**; identified by the MongoDB engine and the Docker subsystem, and viii) **Data flows**.

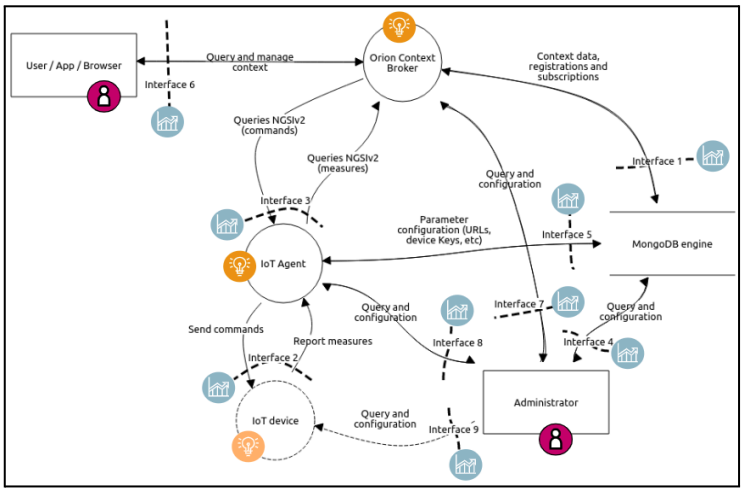

Figure 11: DFD of FIWARE referential platform (own elaboration)

5.2 Threat analysis with STRIDE

STRIDE defines a classification of threats in these six categories: i) **Spoofing** is the act of impersonating or performing actions on behalf of an actor or process; affects the security property of authentication, ii) **Tampering** refers to an intentional and unauthorized act of making modifications to the platform; affects the integrity, iii) **Repudiation** refers to denying responsibility for an action; affects the non-repudiation property, iv) **Information disclosure** involves the exposure of information to an unauthorized actor or process; affects confidentiality, v) **Denial of service** refers to causing a service to be unavailable; affects availability, and vi) **Elevation of privilege** indicates the fact of acquiring capabilities for a process or actor without proper authorization; affects the security property authorization.

Considering that the study focuses on the FIWARE platform, IoT devices and data flows from the Administrator are outside the scope of the threat analysis. We have identified more than 30 (thirty) threats in [25], as shown in Table 1, considering potential weaknesses that can be exploited according to the DFD obtained in Subsection 5.1. It is worth clarifying that each threat has its category indicated in the column Type with a letter from the set {S,T,R,I,D,E}; each represents the first letter of the 6 STRIDE categories mentioned. We will briefly explain some of the identified threats relevant to the attack analysis presented in Subsection 5.3.

Table 1: Classification of threats using STRIDE

ID	Threat	Type	Interface in DFD
T1	**IoT Agent malicious or fake**	S	I2, I3, I5
T2	**Orion malicious or fake**	S	I1, I3, I6

ID	Threat	Type	Interface in DFD
T3	Compromised MongoDB database	S	I1, I5
T4	Attacker spoofing identity of an administrator	S	I4, I7, I8
T5	User spoofing identity of an application	S	I6
T6	Manipulation of context data and configuration settings during scheduled maintenance	T	I4, I7, I8
T7	Manipulation of docker images during scheduled maintenance	T	I4, I7, I8
T8	Alteration of data in transit between Orion and IoT Agent	T	I3
T9	**Alteration of data in transit between Orion and MongoDB**	T	I1
T10	**Alteration of data in transit between IoT Agent and IoT devices**	T	I2
T11	Alteration of data in transit between IoT Agent and MongoDB	T	I5
T12	Alteration of data in transit between user/app/browser and Orion	T	I6
T13	**Impostor Orion taking the place of a genuine instance**	R	I1, I3, I6
T14	Impostor IoT Agent taking the place of a genuine instance	R	I2, I3, I5
T15	MongoDB database compromised spoofing identity of a legitimate one	R	I1, I5
T16	Anonymous update of configuration settings for docker images, Orion, IoT Agent and MongoDB	R	I4, I7, I8
T17	Anonymous update of context data	R	I4, I6, I7
T18	Anonymous update of configuration settings for context data	R	I4, I6, I7
T19	**Eavesdropping on communications between Orion and MongoDB**	I	I1
T20	Eavesdropping on communications between Orion and IoT Agent	I	I3
T21	Eavesdropping on communications between IoT Agent and MongoDB	I	I6
T22	Eavesdropping on communications between user/app/browser and Orion	I	I6
T23	Eavesdropping on communications between IoT Agent and IoT device	I	I2
T24	**Leakage of MongoDB database records**	I	I1, I4, I5
T25	Leakage of docker images settings	I	I4, I7, I8
T26	**Leakage of context data and configuration settings**	I	I4, I5, I6
T27	**Orion unavailable or not responding**	D	I3, I6
T28	IoT Agent unavailable or not responding	D	I2, I3
T29	MongoDB database unavailable, not responding or resource exhaustion	D	I1, I5
T30	**Unauthorised access to Orion**	E	I6, I7
T31	Unauthorised access to IoT Agent	E	I8
T32	Unauthorised access to MongoDB engine	E	I4
T33	Docker sandbox escape to access host	E	I4, I7, I8

Threats T1 and T2 are related to identity impersonation activities. T1 involves the action of replacing a legitimate Orion instance with a modified version that acts like the real Orion but has intended malicious behaviour. T2 threat, on the other hand, represents the act of performing tasks on the platform with the capabilities of an administrator user.

Threats T9 and T10 show cases where a potential attacker can inspect and modify traffic in transit between components in an unauthorised manner. T13 threat constitutes a repudiation threat in which a fake instance of Orion has taken the place of the genuine one, and it is difficult or even impossible to claim responsibility for its actions, since legitimate versus malicious activity is not as easily distinguishable or malicious tasks remain almost unnoticed.

Threats T19, T24 and T26 refer to information disclosure actions: T19 identifies the presence of a MiTM in the unencrypted traffic between Orion and MongoDB that can capture all data stored in and retrieved from the database; T24 makes visible the fact of a possible leak of context data from configuration settings directly from the database, while T26 reflects the point of information disclosure through the REST APIs endpoints exposed by the components.

T27 threat involves an unavailable Orion instance caused by a malicious action. Finally, the T30 threat reflects unauthorised access to Orion to take control of its actions or make other lateral movements.

5.3 Attack analysis

In a first step of the exploration, we decided to carry out the identification of attack vectors for the platform assets that we consider play a central role in the FIWARE technology: Orion and context data. We derive, using the threats obtained in Subsection 5.2, the attack objectives that are summarised in Table 2.

Table 2: Attack analysis - Identification of attack goals for critical assets Orion and context data

ID	Asset	Attacker goal	Affected security properties
O1	**Orion**	**Spoof Orion identity with a modified version**	authentication, non-repudiation, integrity, confidentiality
O2	**Orion**	**Unauthorised access**	authorization
O3	Orion	Escape docker context and access host	authorization
O4	Orion	Degrade service or make operation unavailable	availability
O5	Context data	Unauthorised modification	integrity, non-repudiation
O6	**Context data**	**Data leakage**	confidentiality

Subsequently, we proceeded to experiment with offensive strategies that would allow us to affect the security of the identified attack goals. In particular, we target objectives O1, O2 and O6. The idea was to experiment with remote or network adjacent attack vectors of low complexity. We present a brief sum-up of the attack strategies and their corresponding attack goals in Table 3.

We used the following main techniques and procedures to develop the attacks:

- Knowledge of building docker images.
- Creating bash scripts.
- Generation of reverse shells in Linux.
- Knowledge of SUID technique for privilege escalation in Linux.
- Using tcpdump, mongodb client.

Table 3: Attack analysis - Experimentation with attack objectives

ID	Attacker goal	Attack goals
A1	Modified image of Orion with a background process providing a remote backdoor persistence access. Once the image container has been initiated and it is up and running, a reverse shell process periodically tries to connect back to an attacker listener.	O1, O2
A2	Modified image of Orion providing a SUID binary that allows to gain root privileges, having a shell access. The original /bin/bash binary is copied to a known location and added the SUID flag.	O1, O2
A3	Modified image of Orion providing a mongodb client that allows unauthenticated connection to MongoDB engine.	O6
A4	Use of "network containers" docker feature to allow a container to share the network stack of another container. Context data leakage evidenced using a tcpdump container image sharing the network stack of Orion and used to sniff traffic between Orion and MongoDB.	O6

Attack strategies A1, A2 and A3 are based on a possible identity spoofing of Orion by modifying its base image. To begin with, we studied possible techniques that can be used to replace a running Orion instance by leveraging the generation of a malicious docker image and making it available in a trusted docker registry. We identified 4 (four) potential scenarios where an Orion spoof could occur: (i) a modified and distributed version of the image via the

official FIWARE account on Docker Hub, (ii) a modified and distributed version of the image created by an organisation and distributed to Docker Hub, (iii) a modified and distributed version of the image created by an organisation in a CI/CD pipeline, and (iv) a modified version of the image created directly on the host server or previously distributed to a local registry.

The experimentation process followed tactic (iv) in a locally controlled environment. If we succeed in spoofing Orion, we demonstrate that the use of the altered image could allow:

- Unauthorised remote access to the running instance (see Fig. 12).
- Privilege escalation (see Fig. 13).
- Access to the MongoDB engine and its databases without credentials taking advantage of the absence of access control mechanisms between Orion and MongoDB (see Fig. 14).
- Full control of its operation.

Finally, strategy A4 provides a practical technique to disclose context data because the traffic between Orion and MongoDB is not encrypted (see Fig. 15).

We achieved the attack objectives O1, O2 and O6.

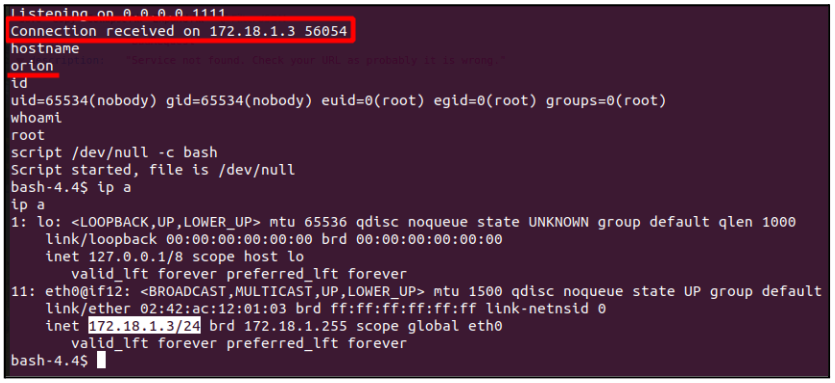

Figure 12: Reception of *reverse shell* to attacker listener

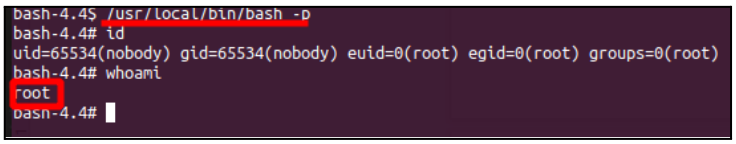

Figure 13: Privilege escalation vía SUID binary

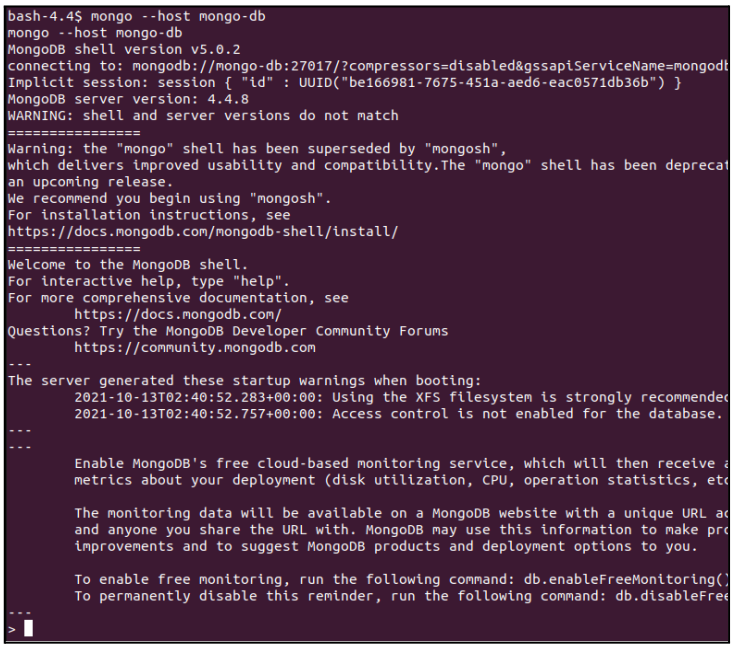

Figure 14: Checking access to MongoDB instance without authentication

Figure 15: Eavesdropping unencrypted traffic between Orion and MongoDB

5.4 An exploratory security analysis of a real FIWARE platform

Close work with the IT and Security departments of the Municipality of Montevideo (government of the city of Montevideo, capital of Uruguay), allowed the design of an exploratory analysis on its FIWARE platform. As the result of a guided questionnaire, Fig. 16 provides a representative diagram of the platform architecture.

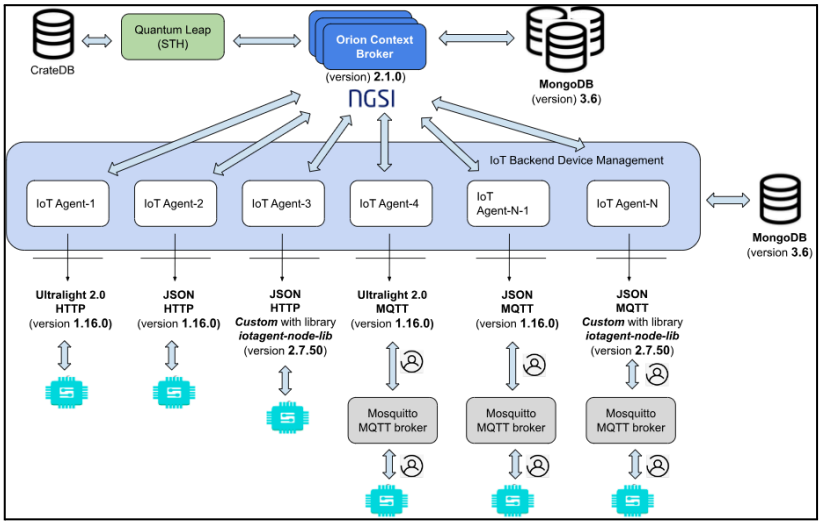

Figure 16: Diagram of smart city platform of the Municipality of Montevideo (own elaboration)

This platform has many architectural similarities, in terms of its design and deployment, with the reference platform depicted in Fig. 10: it is a microservice platform based on an official and personalized on-premises IoT agents with JSON and Ultralight protocols and transports HTTP and MQTT. Furthermore, Orion and IoT Agent components do not authenticate with MongoDB, and communications between the components are not encrypted. The firewall rules for incoming traffic received from IoT devices on the south ports do not consider reverse proxy servers or API gateways. In what follows, we describe the outcomes of our exploratory security analysis.

First, the smart city platform uses Orion version 2.1.0, which does not support TLS-encrypted communications with MongoDB. In terms of the attack strategies described in Table 3, the A4 attack could potentially be used to leak context data. It is important to mention that TLS support was added in version 2.3.0. At the time of this investigation, Orion did not support client-side certificate validation with MongoDB when TLS was enabled. This lack of validation procedure could allow MiTM attacks between Orion and MongoDB to go undetected. Additionally, the iotagent-node-lib library used for custom IoT agents was at version 2.7.50, which does not support authentication or TLS communications with MongoDB. Those capabilities were added in later versions 2.12.0 and 2.13.0. A similar attack strategy to A4 could be used to leak the configuration settings of IoT devices. Moreover, all IoT agents share the same MongoDB engine; that means that a custom IoT Agent or an anonymous user with access to the MongoDB engine could potentially view or modify the configuration settings of an arbitrary database of any IoT Agent in an unauthorized way if we combine this element with two facts: (a) authentication with MongoDB is not configured, and (b) the databases default name is used (if not explicitly set via configuration). Finally, the official versions of IoT Agent 1.16.0 used on the platform allow the registration of measurements from unregistered sensors. Suppose an attacker can intercept communication between IoT devices and agents and capture valid API keys (used to authenticate IoT devices). In that case, they can provide false measurements of real devices or unregistered devices. Sending large-scale measurements could cause the IoT Agent or Orion to malfunction, causing a denial of service or exhausting storage resources in MongoDB.

After performing the exploratory security analysis and identifying possible attack vectors in the Municipality of Montevideo smart city platform, we prepared recommendations and suggestions regarding the components and their interactions. In addition, we proposed access control architectures using FIWARE security components:

- Update Orion to a version greater or equal to 2.3.0 with TLS support with MongoDB.
- Monitor incoming communications to MongoDB to identify abnormal behavior by analyzing log messages. These actions would allow us to be alert of possible MiTM attacks between Orion and MongoDB.
- Update iotagent-node-lib library to a version greater or equal to 2.13.0 with support for authentication and TLS communications with MongoDB.
- Enable authentication with MongoDB in communications between Orion and MongoDB, and IoT Agents and MongoDB.
- Enable encrypted communications between Orion and IoT Agents, Orion and MongoDB, IoT Agents and MongoDB.
- Protect APIs exposed to IoT devices through reverse proxies or API gateways.

Next, we worked on creating some access control alternatives according to the analysis scenario in Fig. 16. In the first place, the addition of a proxy component placed in front of Orion is proposed, which determines whether it allows or denies HTTP requests received (see Fig. 17). It is observed that a PEP Proxy component is added, whose reference implementation in FIWARE is Wilma[10]. This component requires all requests to present a valid access token obtained through the IdM, which in this case is the Keyrock[11] reference implementation. If components have been configured to accept roles and permissions, the PEP Proxy escalates the access decision to Keyrock PDP.

[10] PEP Proxy Wilma: https://github.com/ging/fiware-pep-proxy
[11] Fiware IdM Keyrock: https://github.com/ging/fiware-idm

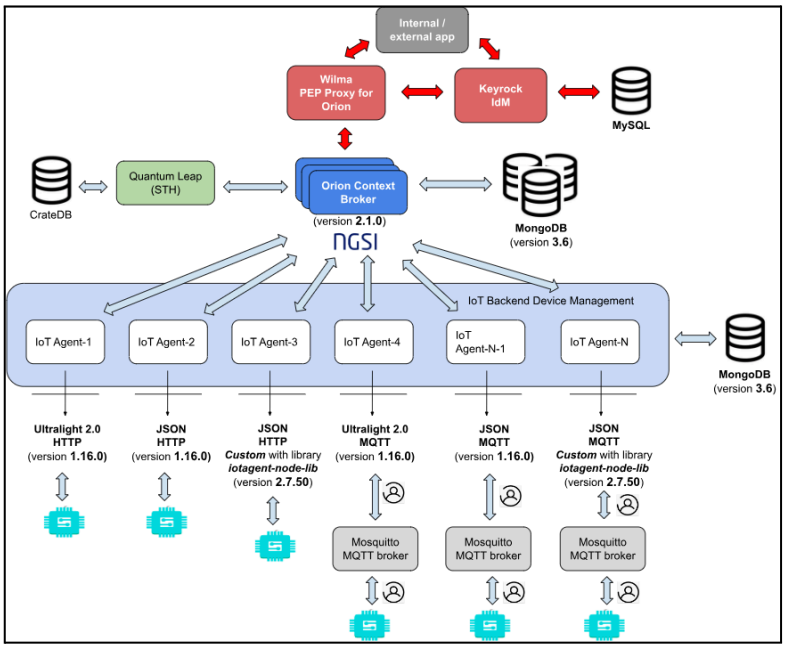

Figure 17: Securing access to Orion Context Broker from applications (own elaboration)

As a second access control proposal, it is recommended to secure access to the south port of the IoT Agent, where a proxy component is placed between the IoT Agent and the sensors as shown in Fig. 18 and determines whether to allow or deny the HTTP requests received from the sensors to the IoT Agent ("northbound" communication). As in the previous case, the presence of the IdM and PEP Proxy is displayed. Following practices from the official documentation, the sensor accounts created through the IdM only intervene in the authentication process but not authorization. From another perspective, it would be possible to add authorization if regular user accounts are created for the sensors in the IdM; to do this it is necessary to establish a logical separation between common user accounts and sensor user accounts, for example using certain terminology in the name registry.

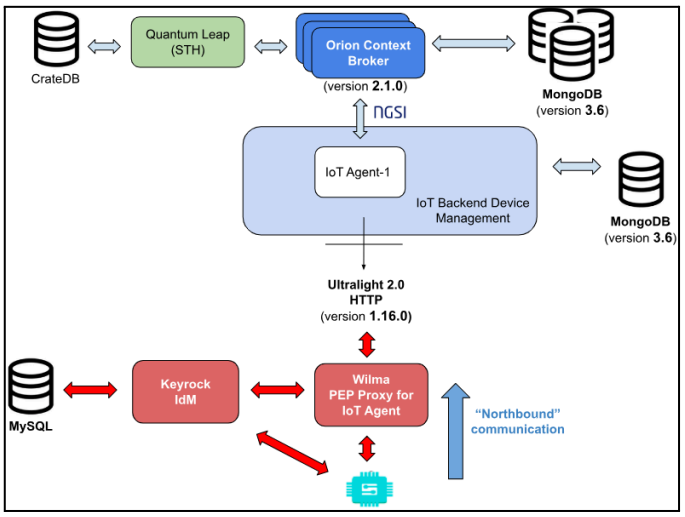

Figure 18: Securing access to Orion Context Broker from Iot Agent (own elaboration)

Finally, Fig. 19 recommends securing access to Orion from the IoT Agent. The idea is to create a specific user account for each IoT Agent.

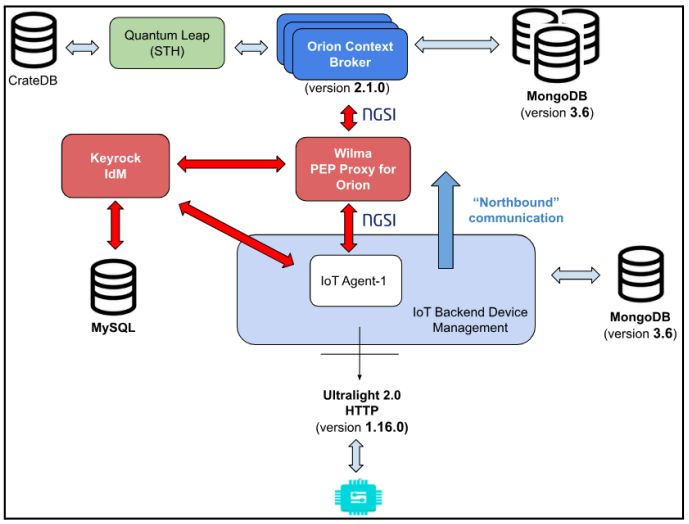

Figure 19: Securing access to Orion Context Broker from Iot Agent (own elaboration)

6 Conclusion and further work

Our research provides a methodological basis to carry out a security analysis of the FIWARE technology and a baseline identification of existing security issues in the FIWARE core components Orion and IoT Agent. The STRIDE-based threat modeling in Section 5, Subsection 5.2 draws a map of different threats, which could serve as a reference for similar or more complex platforms. It is worth highlighting each artifact resulting from the modeling process, such as a granular decomposition of the scenario, its representation in a DFD, and the enumeration of attack objectives for key assets Orion and context data. We achieved the (three) attack objectives proposed in the referential platform in a controlled environment; creating a set of recipes or reproducible steps demonstrating their feasibility was possible. The preliminary analysis of the real platform allowed us to validate and contrast the security analysis performed on the referential platform described in Fig. 1.

Multiple paths are open for future investigation that may deepen and enrich the current work. The security analysis presented only covered Orion and IoT Agent in a potential deployment from an architectural perspective. It would be interesting to deep dive into the low-level implementations, pursuing, for instance, a security evaluation of the implementation of the NGSIv2 REST API in Orion to find security vulnerabilities and a source code analysis of Orion with a specific focus on data flow analysis that may guide possible flaws in code, such as injections and overflows, among others. Also, the same approach discussed could be carried out for the IoT Agent component. Other reference modules within FIWARE solve specific aspects such as storage of historical information, access control, data visualization, etc.; such components could also be considered in a security analysis.

References

[1] Fiware. The FIWARE platform. 2022. [Online]. Available: https://www.fiware.org/

[2] OWASP. OWASP Threat Modeling Process. 2022. [Online]. Available: https://owasp.org/www-community/Threat_Modeling_Process

[3] Microsoft. The STRIDE security Model. 2005. [Online]. Available: https://docs.microsoft.com/

[4] J. Perata and G. Betarte. "A Security Analysis of a Referential Architecture of the FIWARE Platform". 2023 XLIX Latin American Computer Conference (CLEI). La Paz, Bolivia. 2023. pp. 1-9. https://ieeexplore.ieee.org/document/10346164

[5] Santander hackathon. Notification sent to wrong reference endpoint. 2013. [Online]. Available: https://github.com/telefonicaid/fiware-orion/issues/13

[6] M. Fazio, A. Celesti, F. Márquez, A. Glikson, M. Villari. "Exploiting the FIWARE cloud platform to develop a remote patient monitoring system". 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7405526

[7] C. Oliveira et al. “Improving Security on IoT Applications Based on the FIWARE Platform”. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8432306

[8] G. Suciu et al. “FIWARE authorization in a Smart Grid scenario". In 2020 Global Internet of Things Summit (GIoTS). 2020. [Online]. Available: https://doi.org/10.1109/GIOTS49054.2020.9119589

[9] A. Munoz-Arcentales et al. “Data Usage and Access Control in Industrial Data Spaces: Implementation Using FIWARE”. Sustainability pp. 12, 9. 2020. [Online]. Available: https://doi.org/10.3390/su12093885

[10] P. Salhofer, J. Buchsbaum, M. Janusch. “Building a FIWARE Smart City Platform”. pp. 3-5. 2019. [Online]. Available: https://scholarspace.manoa.hawaii.edu/handle/10125/60175

[11] P. Detzner and P. Salhofer. “Analysing FIWAREs Platform - Potential Improvements”. In the 53rd Hawaii International Conference on System Sciences (HICCS). pp. 1-7. 2020. [Online]. Available: https://scholarspace.manoa.hawaii.edu/handle/10125/64551

[12] S. Ijaz, M. Shah, A. Khan, M. Ahmed. Smart Cities: *A Survey on Security Concerns*. pp. 5-11. 2016. [Online]. Available: https://pdfs.semanticscholar.org/8cc4/e546be34d2c62f987b40c99df4f5e5bd969b.pdf

[13] J. Lee, J. Kim, J. Seo. “Cyber attack scenarios on smart city and their ripple effects”. pp. 2-5. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8669431

[14] J. Fox. FIWARE Wednesday Webinars. Core Context Management. March 2019. pp. 4-9. [Online]. Available: https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-core-context-management

[15] H. Esquivel,K. Nájera. El ecosistema FIWARE. 2018. pp. 28. [Online]. Available: https://docplayer.es/75797737-El-ecosistema-fiware-hugo-estrada-esquivel-karen-mariel-najera-hernandez-co-funded-by-the-horizon-2020-framework-programme-of-the-european-union.html

[16] Fimac Project. Backend Device Management IDAS. 2016. [Online]. Available: http://web.archive.org/web/20210615130701/https://fimac.m-iti.org/5a.php

[17] J. Fox. FIWARE Wednesday Webinars - IoT Agents. April 3, 2019. pp. 5-11. [Online]. Available: https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-iot-agends

[18] J. Fox. FIWARE Wednesday Webinars - Short Term History within Smart Systems. April 2, 2020. pp. 5-10, 14. [Online]. Available: https://www.slideshare.net/FI-WARE/fiware-wednesday-webinars-short-term-history-within-smart-systems

[19] M. González. FIWARE Complex Event Processing. 2015. pp. 2-9. [Online]. Available: https://www.slideshare.net/finodexproject/finodex-d32v2-annex2fiwarecep

[20] F. Márquez, K. Zangelin. Managing Context Information at large scale (Advanced topics). 2016. pp. 67-70. [Online]. Available: https://www.fiware.org/wp-content/uploads/2016/12/2_FIWARE-NGSI-Managing-Context-Information-at-large-scale.pdf

[21] Fiware. Fiware Tutorial IoT Agent. FIWARE 202: Provisioning the Ultralight IoT Agent. 2022. [Online]. Available: https://github.com/FIWARE/tutorials.IoT-Agent

[22] Fiware. NGSI-v2 Fiware specification. 2018. Available: http://fiware.github.io/specifications/ngsiv2/stable

[23] Open Mobile Alliance. NGSI Context Management v1. 2012. [Online]. Available: http://www.openmobilealliance.org/release/ngsi/v1_0-20120529-a/oma-ts-ngsi_Context_management-v1_0-20120529-a.pdf

[24] OWASP. OWASP Threat Dragon. 2022. [Online]. Available: https://owasp.org/www-project-threat-dragon/

[25] J. Perata. “Análisis de seguridad de la plataforma de ciudades inteligentes Fiware”. Magister thesis degree. Universidad de la República (Uruguay). Facultad de Ingeniería. 2022. pp. 68-72. [Online]. Available: https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/33680