École Polytechnique Fédérale de Lausanne

IoT Platform for Smart City - Security challenges of Digital Twins

by Dorian Laforest

Master Thesis

Prof. Edouard Bugnion
Thesis Supervisor

Florent Martin
External Expert

EPFL IC DCSL
INN 237 (Bâtiment INN)
Station 14
CH-1015 Lausanne

August 19, 2022

Abstract

To face the Digital Twin security challenges and furthermore the access control to Digital Twins, we aimed at defining a security concept that covers complex digital twin relations, and that is applicable for millions of digital twins and devices.

In the context of this internship, we started with an evaluation of digital twins models and identified their limitations in term of security. We decided to focus on access control and evaluated some concepts. Finally, we moved on the implementation of the Zanzibar global authorization system from Google. In fact, it is designed to manage authorization for complex entities eco-systems including relations between these entities. It is also efficient in managing large sets of entities and thus well adapted to our IoT and digital twin context.

We implemented a proof of concept based on Open-Source IoT platform extended with SpiceDB authorization component (Zanzibar Open-Source implementation). We applied it to a real-life scenario, i.e., access to building rooms, sensors, and actuators.

The approach seems promising, but rise the challenge of creating and maintaining multiples models, i.e., one for the digital twin and a second for the access control policies. This challenge should be studied in future works.

Contents

Chapter 1

Introduction

For the past decades, the use of *Internet of Things* IoT devices has risen exponentially. In a few words, IoT devices are physical objects, augmented with sensors and processing power, which can connect to a network and exchange data. Although they are mostly known for their application in smart home with devices such as smart lights, smart fridges, or smart video security systems, they are also use in other domains such as healthcare, cars, manufacturing and even public services.

Smart Cities are also a good example of the use of IoT devices. It is a city where various public services can be connected. All the data collected can be used to analyze, manage, and make predictions, thus enabling a better and more sustainable governance of the city. Having all data from different public services together allow to break the vertical data silos and enable interaction between the services. An example is a system that could detect where an ambulance must go and control the traffic light such that the ambulance has a safe path towards its destination.

This project comes from the need of defining a concept for a Digital Twin platform representing a Smart City. In this context, a Digital Twin can be seen as a digital representation of element of the city that can be used to control and to connect with others. Such platform comes with specific requirements such as high performance for interacting and controlling the various connected devices, and overall security considering that there is a link between the physical and digital devices. Misuse of the Digital Twin could directly impact the physical devices or worst, it could endanger people lives.

The IoT world is growing exponentially but its security is not mature and lets place to a lot of improvements. Therefore, the goal of this project is to propose a concept of security framework for Digital Twin. As the security of such a platform concerns a wide range of aspects such as the security of the digital device itself or the security of the API proposed to the consumer of the Digital Twin, this work will focus on one aspect: the access control of the Digital Twin. That is how can we make a fast, reliable, and high performant fine-grained access control for such a platform.

focus: the access control of the digital twin

In order to design that platform, we need to understand what a Digital Twin is, how it is designed and how it works. Then we can focus on security challenges and on the existing platform. The two major actors in this domain are Microsoft with its Azure Digital Twins platform and FIWARE with its Open-Source framework of components. Both platforms are a good step toward a digital twin platform but they have some shortcomings on their access control system. The access control mechanism of Microsoft's platform is limited and does not allow for fine grained access control. The FIWARE framework has some components for security and allows more fine-grained control but there are some limitations as the access control uses XACML that can be very verbose and complex to manipulate, and the security components have some limitations in term of configuration and integration.

As the existing solutions does not meet the requirements we set for a Digital Twin platform, we chose to enhance the access control of the FIWARE framework and implement a proof of concept that allows a fine grained, consistent, and high performing access control system.

Chapter 2

Background and related work

2.1 Digital Twins

A Digital Twin can be seen as a real-time digital representation of a real-world object, process, service or even a person. This representation is defined by diverse attributes that can be properties (e.g., temperature, speed, status) or relationships, linking Digital Twins together. These attributes do not need to be measurable data, they can be metadata or even computed data (e.g., predictions from some machine learning algorithm). [4, 16]

A Digital Twin is updated using real-time data so that it depicts an accurate view of the object it represents. Through it, one can also control the real entity when applicable, meaning that the link between the digital twin and its counterpart can be bi-directional, synchronous and in real-time.

It can be used for simulation, prediction, command and control, optimization or simply to view and understand its real-world counterpart current situation.

A Digital Twin is not a simple running digital service or a microservice, it is in fact, an interface that exposes commands and data at current time and the infrastructure used to power it can be quite complex. This infrastructure requires multiple components such as a database to keep the Digital Twins models and live data, a database to keep historical data of twins, a component to communicate with IoT devices and make the translation between native IoT communication protocols and higher-level protocols, and more. Additionally, all this complex infrastructure needs to be secured.

2.1.1 Models

Building Smart Cities requires to have interoperability between the various systems in order to easily use all the data in a meaningful way. There currently exists many models to represent Digital Twins, making the task difficult. To improve that, actors in industries as well as *Standards Developing Organizations* (SDO) have tried to create models that could enable the interoperability among various domains and stakeholders [11, 23] but there is presently no consensus on which model to use. Nevertheless, some of them are more prominent.

Ontology

An ontology is a formal way of representing the most general features and relations of a subject. It can be further divided in three parts: the meta-ontology, the core ontology, and the application ontology.

The meta-ontology is the ontology of the global key concepts. It defines the building blocks of the ontology. The core ontology is the one built on top of the meta-ontology and that specifies the core of the system that we want to represent. The application ontology is refining the core ontology to a specific domain of application. Figure 2.1 shows the relationships between the ontologies of two of the main actors in the smart cities' domain: Microsoft and *European Telecommunications Standardization Institute* (ETSI) [12]. We can see that these ontologies are based on JSON-LD.

Using their respective ontologies, Microsoft and ETSI have defined their model to describe Digital Twins.

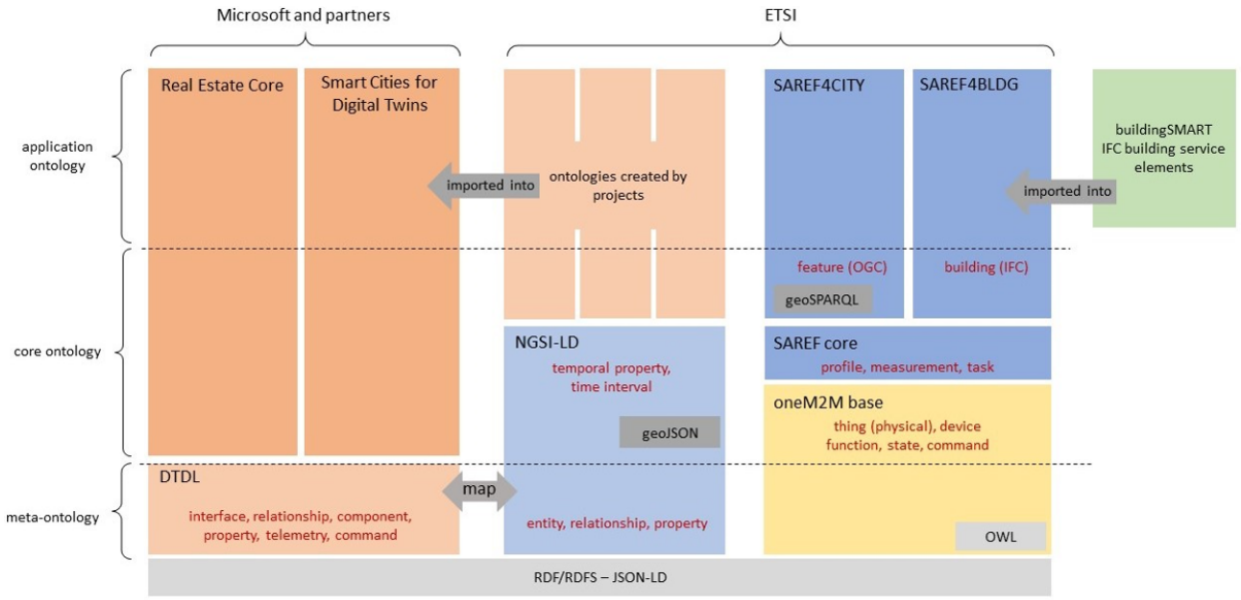

Figure 2.1: Summary of Digital Twins ontologies. Source: [1]

2.1.2 JSON-LD

JSON for Linked Data (JSON-LD) [25] is a linked data format based on JSON. Its purpose is to encode contextualized meaning into regular, meaningless, JSON documents. To do so, it uses *Internationalized Resource Identifiers* (IRIs) [10] as key for the JSON data. That way, there is an unambiguous identification of terms and not only it avoids developers to use the same key for different purpose, it also allows machine and human to get a definition of what the term means by following its IRI.

As this notation can lead to exceedingly verbose documents, JSON-LD introduces the *@context* object that will contains the mapping between IRIs and shortcut terms. See Figure 2.2 for an example of a JSON-LD document representing a person with and without the uses of *@context*.

Figure 2.2: Example of a JSON-LD document representing a person. With and without the use of *@context*

2.2 Security challenges of Digital Twin

On top of the common security challenges such as confidentiality, integrity and availability, Digital Twins also highly increase the exposition surface as seen in Holmes et al. [19]. In their paper, they describe the main challenges and risks created by the use of Digital Twins. Even if some of the challenges are described for specific types of Digital Twins, they can still apply to all. Indeed, the Digital Twins infrastructure uses an immense number of connected devices that comes from various vendors and each of them is not secured at the same degree. Synchronous connection data exchange in real-time also highly increases the cybersecurity risks. For example, a malicious update to a Digital Twin may be directly reflected in its counterpart and endanger people's life. Digital Twins can mimic physical systems with high accuracy, thus gaining access to them could easily allow an attacker to bypass the security that is applied to the base system. Additionally, as the Digital Twins are an entry point to the system they can potentially allow access to other twins, to data that can contain *Personal Identifiable Information* (PII) or *Intellectual Property* (IP). As the Digital Twins are driven by data, this can create new challenges with regard to compliance and data regulations.

Another important security challenge is privacy. Braun et al. [5] describes the privacy threats that Smart Cities can induce and how some of the challenges can be solved with methods such as the hardening of systems and cloud security. They also describe the cascading effect of privacy and security issues as everything is interconnected and state that layered security approaches as well as transparent standards for privacy are crucial.

Various devices need to be connected and potentially old existing system need to be incorporated into the new platform, some usual approaches on security may not be applicable. As

Hearn and Rix [18] point out, original security models with hardware security or air-gapped systems may not be applicable to Digital Twins. Existing critical systems using these techniques need to be updated and handled in a new way. A novel approach to security for Digital Twins needs to be taken as the popularity of IoT makes it an easy target for hackers.

As there is a lot of security challenges on a vast number of domains going from IoT devices trust and authentication to end user control and data privacy, this thesis will focus on one aspect that is crucial: access control.

2.3 Access control concepts

Access control is a primary security mechanism. Its goal is to guarantee that all access on the system respects the security policies that were defined.

There exist several paradigms for access control, the primary ones are shown in Table 2.1.

	Access Control List (ACL)	**Role Based Access Control (RBAC)**	**Attribute Based Access Control (ABAC)**
Concept	Lists associating permissions to objects. Permissions consist of action allowed for a subject (e.g., read, write, execute).	Permissions are assigned to roles and roles to subjects	Attributes of objects, subjects or environment are used
Example	Resource Y: (User X, read/write), (User Z, execute)	Resource Y is accessible by users with the role building manager	Resource Y of type building is accessible by users with the role building manager on weekday from 8am to 8pm

Table 2.1: Primary access control paradigm

In massive systems, using ACLs is not feasible as there can be a boundless number of users and resources. Maintaining such lists is not possible in an efficient way as it is hard to know who can access a specific object. RBAC can help reduce the number of rules and improve maintainability by grouping users by roles but it does not allow fine-grained access control. ABAC solves this problem and allows the access control system to model relationships and complex policies. ABAC can also be called *Policy-based Access Control* (PBAC) as it uses policies to express what is allowed or not.

eXtensible Access Control Markup Language (XACML) [14] is a standardized implementation of ABAC and offers a policy language as well as an architecture and a processing model for these policies. The main goal of the architecture is to separate the client and the access decision. That way the policies can be updated and applied immediately to all clients. Two major components of this architecture are the *Policy Enforcement Point* (PEP) and the *Policy Decision Point* (PDP).

simple ACLs cant feasibly define all required accesses to a boundless amount of actors and un unknown amount of resources

eXtensible Access Control Markup Language

policy decision point

point enforcement policy

The role of the PEP is to catch clients' access requests to a resource, build a decision request that will be forwarded to the PDP and enforce the PDP response by either allowing or denying the access to the resource. The PDP uses the authorization policies to evaluate the decision requests it receives.

XACML not scalable and hard to write in

Even if XACML is around for several years and is widely used, the evaluation time of a PDP does not scale well with a high number of policy rules [8]. Additionally, XACML is very verbose, and it is not that easy to write rules in it. Therefore, it is not really suitable for a Digital Twins perspective as there will be a high number of rules. There is also a need for low latency and high scalability.

Zanzibar

Zanzibar [27] is Google's consistent, global authorization system used in its services such as YouTube, Calendar, and Drive. It is a *Relation-Based Access Control* (ReBAC) system that offers a simple and flexible data model to represents ACLs. In addition to its ability to scale to trillion of ACLs and millions of authorization requests per second, Zanzibar also provides consistency of access control decision even with constant changes to ACLs and object contents. The system therefore achieves correctness with low latency and high availability.

It provides a powerful configuration language that can define any relations between users and objects, and that can model group memberships as well as nested group. Furthermore, the language allows complex access control policies to be made by supporting set operators such as intersection, union and difference.

ACLs are defined as lists of relation tuples that resemble "user U has relation R to object O" or "set of users S has relation R to object O", where S can be another object-relation pair. The object O can also represent a group to model group memberships. Similarly, ACL checks have the form "does user U have relation R to object O?"

In addition to ACL checks, Zanzibar's API allows clients to read and write relation tuples, to watch tuple updates that happened since a specific timestamp and to inspect the active ACLs.

One problem that can arise in a distributed permission system is, as called in the paper, the *new enemy* problem. It occurs when the system either fails to respect the order in which ACL updates have happened or uses old ACLs on new content. This problem can be caused by a replication delay that would make the system to not respect the causal ordering of ACL updates. Table 2.2 shows examples of this problem.

Neglecting ACL update order	**Misapplying old ACL to new content**
1. Alice removes Bob from the ACL of a folder.	1. Alice removes Bob from the ACL of a document.
2. Alice then asks Charlie to move new documents to the folder, where document ACLs inherit from folder ACLs.	2. Alice then asks Charlie to add new contents to the document.
3. Bob should not be able to see the new documents but may do so if the ACL check neglects the ordering between the two ACL changes.	3. Bob should not be able to see the new contents but may do so if the ACL check is evaluated with a stale ACL from before Bob's removal.

Table 2.2: Examples of the new problem taken from Zanzibar's paper[27].

To solve this problem, Zanzibar uses Spanner [7], Google's global database system, to store ACLs. Spanner's TrueTime mechanism is used to assign each transaction a timestamp that respects a global ordering and ACLs checks are evaluated at a specific timestamp in all its databases to ensure that they all have the same view. Although using the latest timestamp during ACL checks would ensure having all the ACL updates, it can drastically increase the latency. Zanzibar therefore uses a timestamp that could result in using slightly stale data but with low latency.

As some scenario requires to use the most recent ACLs version, Zanzibar allows clients to set a lower bound on the timestamp that will be used for ACL checks. To do so Zanzibar defines a *Zookie* which is an opaque token representing a specific global timestamp. When clients do ACL updates, they receive a Zookie back and can then use it to do ACL check that encompasses all updates up to and including the one specified by the Zookie.

A global authorization system allows a consistent user experience across applications, it also allows applications to coordinate access control when objects come from different applications. Furthermore, such a system requires to solve engineering challenges concerning data consistency as well as scalability. Dealing with these challenges for all applications at once helps reducing the resources needed.

As no code or proper API definition is given, some initiatives have developed their own implementation of Zanzibar. Authzed [20] is one of them and is currently one of the most mature version. It is a multi-tenant permissions system as a service based on Zanzibar. It also provides

an Open-Source version that is called SpiceDB [29].

2.4 Main Digital Twin implementations

2.4.1 Azure Digital Twins

Azure Digital Twins (ADT) [3] is Microsoft's solution for a Digital Twin platform. It is currently in active development, and it had new features added as well as functional changes during the execution of this thesis.

ADT can be used to create twin graphs based on digital models that can represents building, factories, or even entire cities. Figure 2.3 shows an example of a twin graph and model graph. ADT can also be connected to Azure IoT Hub to connect the Digital Twins to actual IoT devices and update the twins with live data. Additionally, other Azure services can be connected to ADT in order to build advance solutions. An example of such service is *3D Scenes Studio* (currently in preview mode) that can represent an environment in 3D and enables the monitoring and controlling of the physical twins through 3D scenes. It is therefore easy to deploy a Digital Twin infrastructure with advance features using Azure services.

ADT is using its self-defined modeling language, *Digital Twin Definition Language* (DTDL) [9], to define Digital Twins models. DTDL is based on JSON-LD and is also in active development.

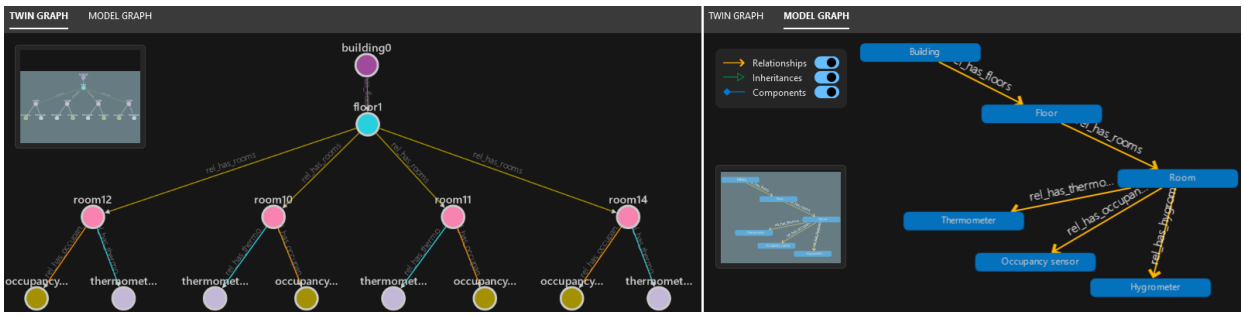

Figure 2.3: Example of a Twin graph and a Model graph in ADT Explorer.

For security, ADT uses Azure Active Directory as well as Azure RBAC to manage access control, authentication, and authorization. It provides encryption of data at rest and in-transit using Microsoft-managed encryption keys.

2.4.2 FIWARE

FIWARE [15] is a framework of Open-Source components used to build and develop smart solutions and Digital Twins.

FIWARE is using *Next Generation Service Interfaces - Linked Data* (NGSI-LD) [13] as an information model and API for interacting with context information. It is standardized by ETSI and is also based on JSON-LD. It comes from research projects and is used in several other research projects as well as in some smart cities' initiatives. [24, 26]

The FIWARE framework is composed of multiple components that can be assembled together and with third-party components to ease the creation of smart solutions platform. These components are separated in categories as showed in Figure 2.4. Note that although not explicitly shown, components handling security exists and are in the "Data/API Management, Publication and Monetization" category.

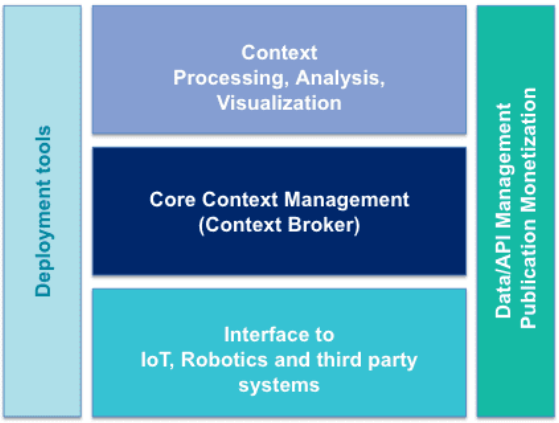

Figure 2.4: FIWARE components. Source: [15]

The main component at the core of the FIWARE architecture is the context broker. Its role is to enable the management of context information. Orion-LD is the NGSI-LD context broker proposed by FIWARE. It exposes an HTTP REST API following the NGSI-LD API specification.

IoT Agents are components responsible to interact with devices using IoT protocols (such as LWM2M or LoRaWAN) and transfer the devices request into NGSI-LD before sending them to the context broker.

To implement secure access to components, there is three components that can be used:

- Keyrock is an Identity management component and provides OAuth2-based authentication of users and devices.
- AuthZForce is an authorization *Policy Decision Point* (PDP) and provides an API to get authorization decisions based on authorization policies defined in XACML v3.0 (eXtensible Access Control Markup Language).
- Wilma is a *Policy Enforcement Point* (PEP) Proxy. It can be used together with Keyrock and AuthZForce to secure and enforce access control to other applications such as Orion-LD.

An example of architecture using these security components is shown in Figure 2.5. An application that wants to send a request to the context broker will go through the PEP Proxy. The PEP will check that the request contains a valid access token and will retrieve user information such as the user role from Keyrock. It will then use these information in addition to the information from the HTTP request to create an XACML decision request that will be send to AuthZForce. The PDP will check with its policies if the user is allowed to do the request to Orion-LD and it will send back to the PEP its decision. The PEP will then either deny the request or forward the request to Orion-LD and send back the answer to the application.

There is no mention of encryption for data at rest, it is to the developer to enable and correctly configure the various components. Likewise, TLS is not activated by default and need to be setup in each component.

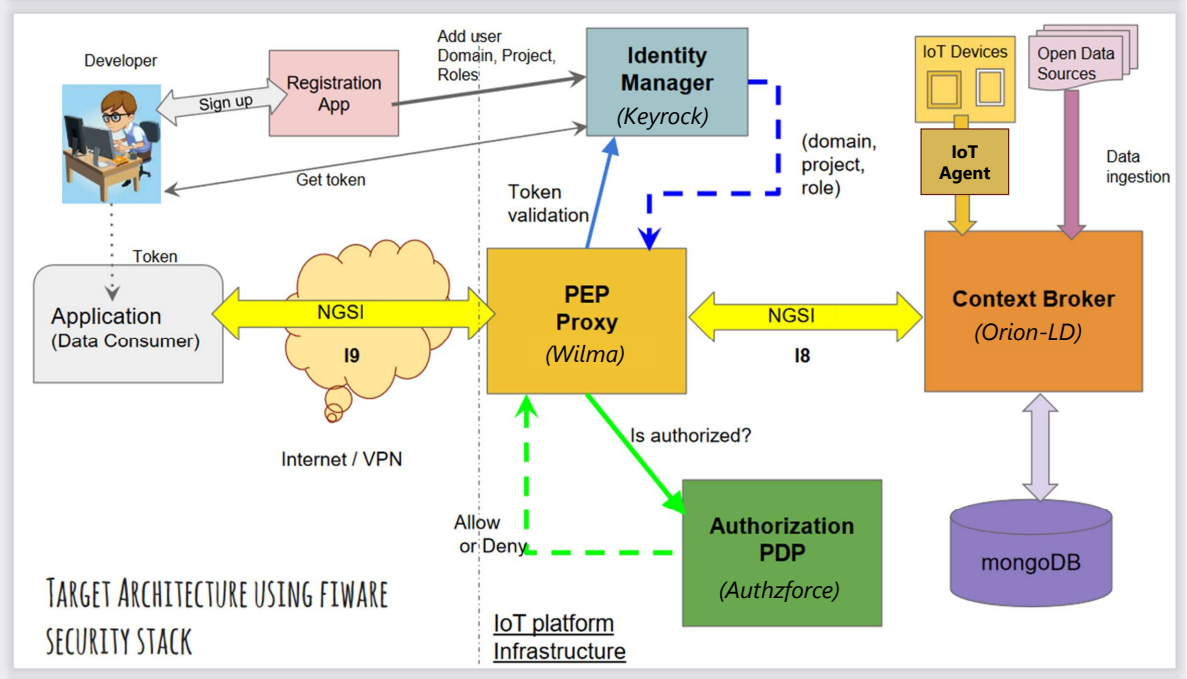

Figure 2.5: FIWARE architecture with security components. Source: [6]

2.4.3 Conclusion and limitations

azure digital twin

ADT seems to be a good platform for Digital Twins as it allows to deploy rapidly the infrastructure and services. However, it is currently still in development and although it can connect to multiple

Azure services it requires to write Azure functions to connect the different part and propagates the data. Additionally, from a security point of view, ADT uses Azure RBAC and Azure Active Directory for authentication and authorization. The permission scopes are limited and in the current version can only be set at global level, i.e., models, twins, relationships, and queries but not to specific twins, models, relationships or attributes.

FIWARE also seems to be a good platform for Digital Twins. With its Open Source and modular architecture it allows to develop all kind of application and infrastructure. However, Salhofer [28] shows that the proposed components and structure for security has limitations in term of configuration and proper integration and the current documentation may lack important or updated information. Furthermore, the policies are in XACML which can be excessively tedious to write, maintain and update.

Chapter 3

Security challenge: access control

multiple actors, with different data management protocols

In a smart city design, there are multiple actors coming from different businesses, domains, and companies. Many of these actors need to interact on objects (Digital Twins) that can belongs to them or to other actors and perform CRUD operation on them.

need to interact with any given number of existing data

An example of such application would be a city where data from waste management, such as position of garbage trucks and filling status of garbage bins, could be used together with data from the smart traffic domain, such as air pollution and traffic jam. With these joint information, garbage trucks could optimize their path in real-time to avoid traffic jams and places where the garbage bins are empty. This could help reduce pollution and could enhance traffic fluidity.

need for high performance system that can handle hundreds of devices/actors with low latency

For such a system to work, there are additional constraints that need to be solved. As one can control the physical objects through their Digital Twins or need information in real-time from these twins, there is a need for a high performance system that can handle hundreds to thousands of devices and actors with low latency and high availability. Moreover, the system should be able to handle complex models with relations and inheritance between objects. It should also be able to allow fine-grained, configurable access control to these models and objects. It is therefore a paramount requirement to have a security concept that can handles all these constraints.

allow fine tuning of resource access at any time (allow reconfiguration)

As seen with Azure Digital Twins and FIWARE, both actors in the smart city domain acknowledge the need for security and implement some approach for it. Nevertheless, there is some shortcomings in their solutions. ADT uses an RBAC system to handle authorization, but it does not currently allow fine grained permissions at an attribute or specific Digital Twin level. This means that a user has access to either all Digital Twins or none. The FIWARE ecosystem offers components for security, but they are not easy to use nor configure as their documentations is not always up to date or even existing. Furthermore, the use of advanced policies for access control can be tedious as it requires writing complex XACML files and their maintenance is complicated. Using a system that is complex to configure and use can easily leads to misconfigurations or errors that can have a substantial impact on the security.

either cant define fine tuned resource access.

or dont offer an easy configurable policies for access protocol, if the system is hard to use --> misconfigurations or errors that may have huge security impacts

For these reasons, this work is focused on designing a security concept for Digital Twin that has a flexible and efficient access control mechanism at its center.

this is the approach defined by the author

To do so, we will start from the FIWARE architecture because it is a good base to start. Indeed, FIWARE is Open Source and have a modular architecture, allowing us to focus our work on specific components. That way we can upgrade and add security components while keeping the others responsible for Digital Twins. We will create a new Policy Decision Point based on the SpiceDB Open Source implementation of Zanzibar and we will update the existing Policy Execution Point Wilma to connect to and use the new PDP. We will also update the Identity Provider to Keycloak as it is a more accessible, configurable and mature IdP. In the following sections we will design and implement a proof of concept for this system.

what will be done to achieve the goal of the project

Chapter 4

Proposed solution

Security is a major necessity for a Digital Twin platform because it enables actors to manipulate data that can contain Personal Identifiable Information or Intellectual Property. Additionally, it allows the control of physical devices in real time. These physical devices can be critical to the correct functioning of systems, therefore, their access should be restricted and controlled. As the security of such a platform is manifold we will focus on the access control of the Digital Twins.

There are crucial requirements to this access control system. Since it will be at the front of all actions and controls done to the digital twins, the access control system should be scalable and flexible while maintaining high availability and low latency.

Access control system is scalable and flexible while maintaining high availability and low latency

In order to design a concept for the access control system, we will update an existing Digital Twin platform. Then, we will define a proof of concept based on that design and detail a working implementation of it.

4.1 Design

We will base the Digital Twin platform on the FIWARE platform as it is currently the most mature and used platform in research projects and smart cities initiatives. Furthermore, it is an Open-Source framework of modular components allowing us to easily update or add components to achieve our requirements. We will use a small scenario to help reasoning and modeling our concept.

4.1.1 Scenario

In a smart city design, there are multiples actors coming from different businesses, domains, and companies. Many of these actors need to interact on objects, i.e., Digital Twins by doing CRUD actions. The context can be illustrated by the following use case:

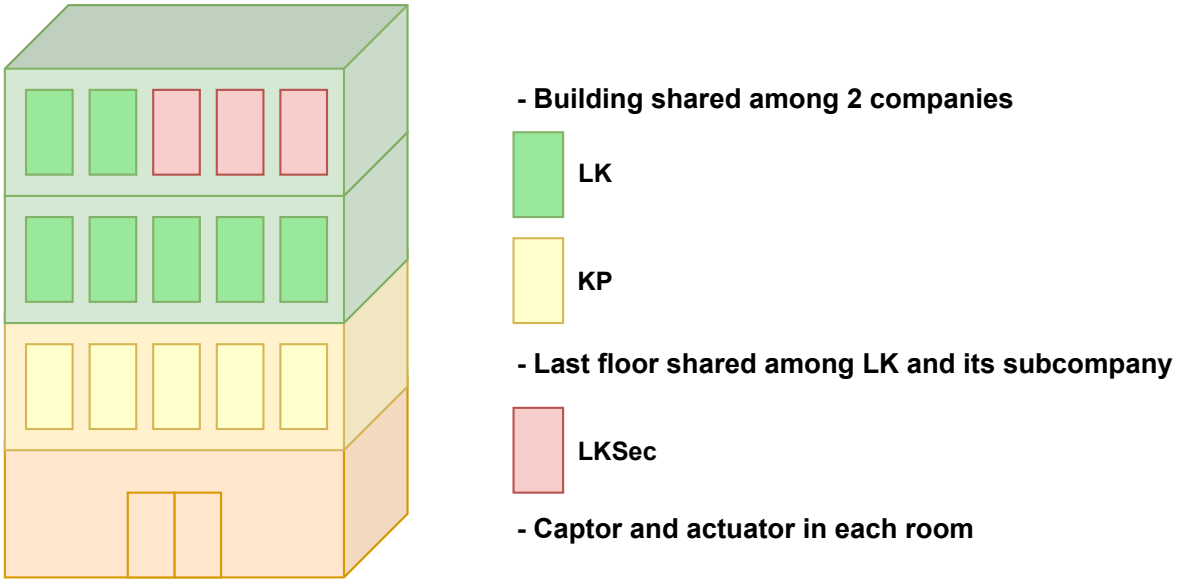

Figure 4.1: Building use-case scenario

Users from one company should not be allowed to access the captors and actuators from other companies. Furthermore, users from a subcompany should be allowed to access the captors and actuators from its parent company but not necessarily the other way around.

Note that although in a real smart city scenario we would have many more different objects, this simple example is perfect to mimic the most important aspect we want to show, i.e., access control. Furthermore, by choosing the right components, the access control can be done at a more fine-grained level, e.g., it could use the type of a Digital Twin or even some of its attributes.

4.1.2 Architecture and main components

A high-level architecture of the proposed Digital Twin platform is shown in Figure 4.2.

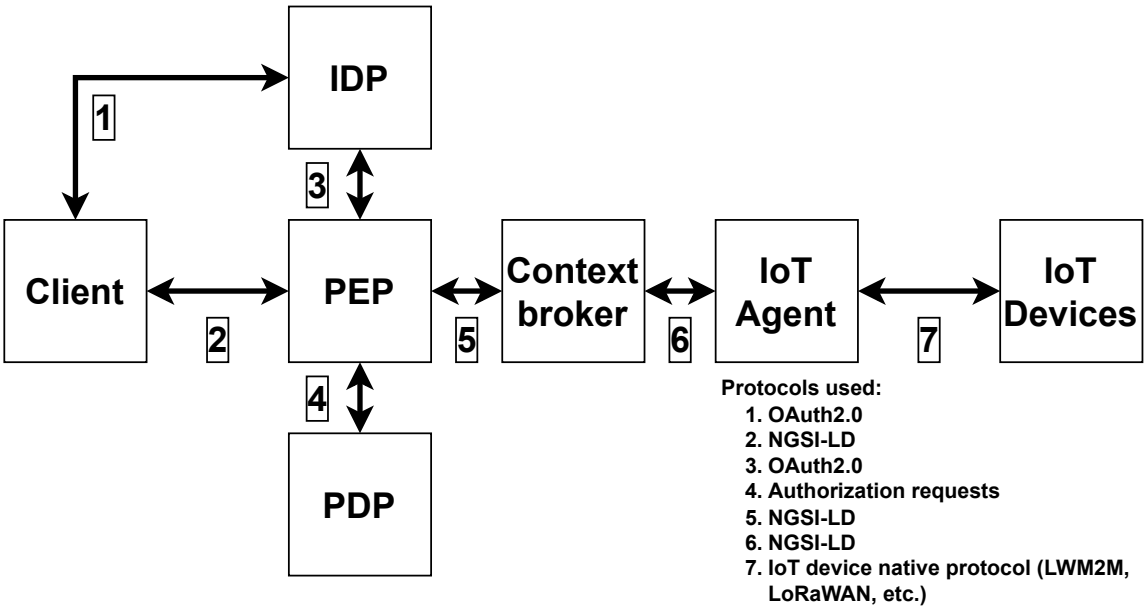

Figure 4.2: High-level architecture of the proposed Digital Twin platform

We will now detail the design of all components and explain how they interact with each other.

IoT Agent and IoT Devices

The IoT Agent is the component responsible to make easier the integration of IoT devices, robots, and other third-party systems. This component makes a bridge between IoT protocols such as LWM2M, LoRaWAN or HTTP/MQTT messaging and NGSI-LD. It will therefore be able to propagate IoT devices telemetry to the context broker and propagate commands from the context broker to the IoT devices.

Context broker

context broker is the storage, and is the information relayer(?) through the REST API

The context broker is the main component of the digital twin platform. It contains all the Digital Twin representation and data, and it exposes a REST API to query, manipulate and control these Digital Twins with NGSI-LD. Users go through the Policy Enforcement Point to do NGSI-LD request and IoT Devices go through IoT Agent. In FIWARE, the main implementation of the context broker is Orion-LD therefore it is the one that will be used.

Client client is an interface, who will do queries on behalf of the user

The client represents the component that will be used by users to make NGSI-LD requests to the context broker. These requests are used to query, manipulate and control the Digital Twins. The client will first retrieve an OAuth2.0 User Access Token from the IDP and then it will make the request to the Policy Enforcement Point with the token attached. As both interaction from the client use HTTP, we will use curl to make the requests and simulate a real application.

As seen with Figure 2.5 in subsection 2.4.2, the access control system of FIWARE uses three components: an Identity Provider, a Policy Decision Point and a Policy Enforcement Point. These are the three components that will be changed or updated.

IdP

Keyrock is the FIWARE component used as Identity Provider and Management. It offers the main functionalities one can expect from an IdP such as the support of OAuth2.0, user profile and attributes, and Single Sign-On functionalities. However, its configuration and usage are quite limited and its interface is not user friendly and cumbersome to use. For these reasons this component is replaced by Keycloak which is a mature and well-known Open-Source and extensible Identity and Access Management service that support the major standard protocols such as OpenID Connect and OAuth 2.0.

Keycloak will be used to manage the users of our system and to authenticate the users when they do requests to the context broker API.

PDP

Authzforce PDP is the FIWARE component responsible for authorization. It uses XACML 3.0 policies to do authorization decision. This PDP is not easy to configure, maintain and updates, and it uses XACML which makes writing, updating and maintaining policies tedious and hard. Thus, we replace this component with another authorization system. From the requirements, the new system needs to offer high availability, low latency, and enough flexibility to handle complex use case. Zanzibar is therefore a suitable candidate to replace the curent PDP.

read previous sections to understand zanzibar

As Zanzibar is not Open Source, we will use SpiceDB which is an Open-Source implementation of Zanzibar. SpiceDB is the best candidate among the Open-Source implementations of Zanzibar because it is currently the most mature and offers all the main functionalities of Zanzibar. It is a Relation-Based Access control with a powerful and flexible configuration language to model all kind of use-case. It has an API allowing users to read and write relationships, check ACL permissions and watch updates that happened since a specific timestamp and although it

making sure the correct snapshot is being looked at

does not reach the same performance number as Zanzibar, SpiceDB offers high availability and low latency.

As the source code or some specific implementation's details were not disclosed or only vaguely described in the Zanzibar paper, SpiceDB cannot be exactly the same as Zanzibar. Indeed, Zanzibar is using Spanner as its backend to prevent the new enemy problem and it is using Google's ID management system GAIA to get unique and global user ID.

On the other hand, SpiceDB uses CockroachDB as one of its proposed backend. CockroachDB allows SpiceDB to scale horizontally and provides global distribution and availability required by a permission system. Even though CockroachDB does not have an API similar to Spanner's TrueTime API, the authors of SpiceDB have found a way to configure and add mitigations to CockroachDB in order to avoid the new enemy problem[21]. Furthermore, SpiceDB does not represent the user in a numerical format but let the developer model the user in order to allow complex use-case. It also allows optional parameters to enforce boundaries between multiple tenants that are in the same deployment, and it offers an additional API endpoint that lists resources to which a subject has a specific permission.

As SpiceDB will be used to do ACL check to know if a user has a specific permission to a digital twin or if a user can create a new digital twin, we need to model the relation and permission that will be used by the scenario defined in subsection 4.1.1. We will use a digital twin object to represents the different entities (buildings, floors, rooms, captors, and actuators). These digital twins have a company as owner and can be related to another digital twin, e.g., a floor has a building as a parent. We also need to represent the users that can read, update, and delete the digital twin.

ACL checks in SpiceDB have the form "Does user U have relation R to object O?". Therefore, we cannot put the create permission in the digital twin object as it will not exist when issuing the check. To solve that issue, the create permission will be set in the company object, that way user of the company that have the creator relation will be able to create new digital twins. To take advantage of the relations, we can also define groups of users that can read, update, and delete digital twins at company level. Additionally, as in the scenario some companies should not be able to access their subcompanies digital twin, we need a way to cut the inheritance of digital twins in some case. We can solve this issue by using the fact that permissions are sets in spiceDB and we can use set operations on them.

The schema in Figure 4.3 is the result of the above objects, relations and permissions in the SpiceDB format. This schema represents the structure that SpiceDB will manage. The definition are the objects we want to store, the relations are how these objects relate to each other and the permissions are sets of users that will be computed from the actual relationships that will be added to SpiceDB during operation. For example, to declare that user X123 is member of company LK the following relationship should be issued to SpiceDB: "user:X123, member, company:LK".

Similarly, to check if user X123 can read the digital twin Actuator123 the following ACL check should be issued to SpiceDB: “user:x123, reader, digital_twin:Actuator123”. And to check if user X123 can create a Digital Twin, we need to check if that user has that permission at his company level. The ACL check would therefore be “user:X123, create_digital_twin, company:LK”.

From the above examples, it follows that the information needed for the ACL checks are the user ID, the permission that we want to check and the digital twin ID or the user’s company.

```
definition user {}

definition company {
    relation member: user | company#member
    relation dt_creator: user
    relation dt_updater: user
    relation dt_deleter: user

    permission create_digital_twin = (dt_creator & member)
    permission read_digital_twin = member
    permission update_digital_twin = (dt_updater & member)
    permission delete_digital_twin = (dt_deleter & member)
}

definition digital_twin {
    relation owner: company
    relation parent: digital_twin

    relation reader: user|company#member
    relation updater: user|company#member
    relation deleter: user|company#member

    // relation that will contain the same digital_twin as the parent if we want to not inherit permission from the parent
    relation not_inherit_parent: digital_twin

    permission read = owner->read_digital_twin + reader + (parent->read - not_inherit_parent->read)
    permission update = owner->update_digital_twin + updater + (parent->update - not_inherit_parent->update)
    permission delete = owner->delete_digital_twin + deleter + (parent->delete - not_inherit_parent->delete)
}
```

Figure 4.3: SpiceDB schema of the PoC scenario

PEP

Wilma PEP is the FIWARE element that is used to proxy the components that need to be secured. To do so, it checks incoming requests for authentication and authorization with the help of the IdP and PDP. As the IdP and PDP has been changed, this PEP proxy needs to be updated to interact with the new components.

The authentication and authorization flow of a request can be seen in Figure 4.4. It consists of three steps. (1) All requests to the context broker need to go through the PEP Proxy and they need to contain a User Access Token. The PEP will check if the token is valid and if so, it will retrieve the user information from Keycloak. (2) The request will be validated by using the OpenAPI 3.0

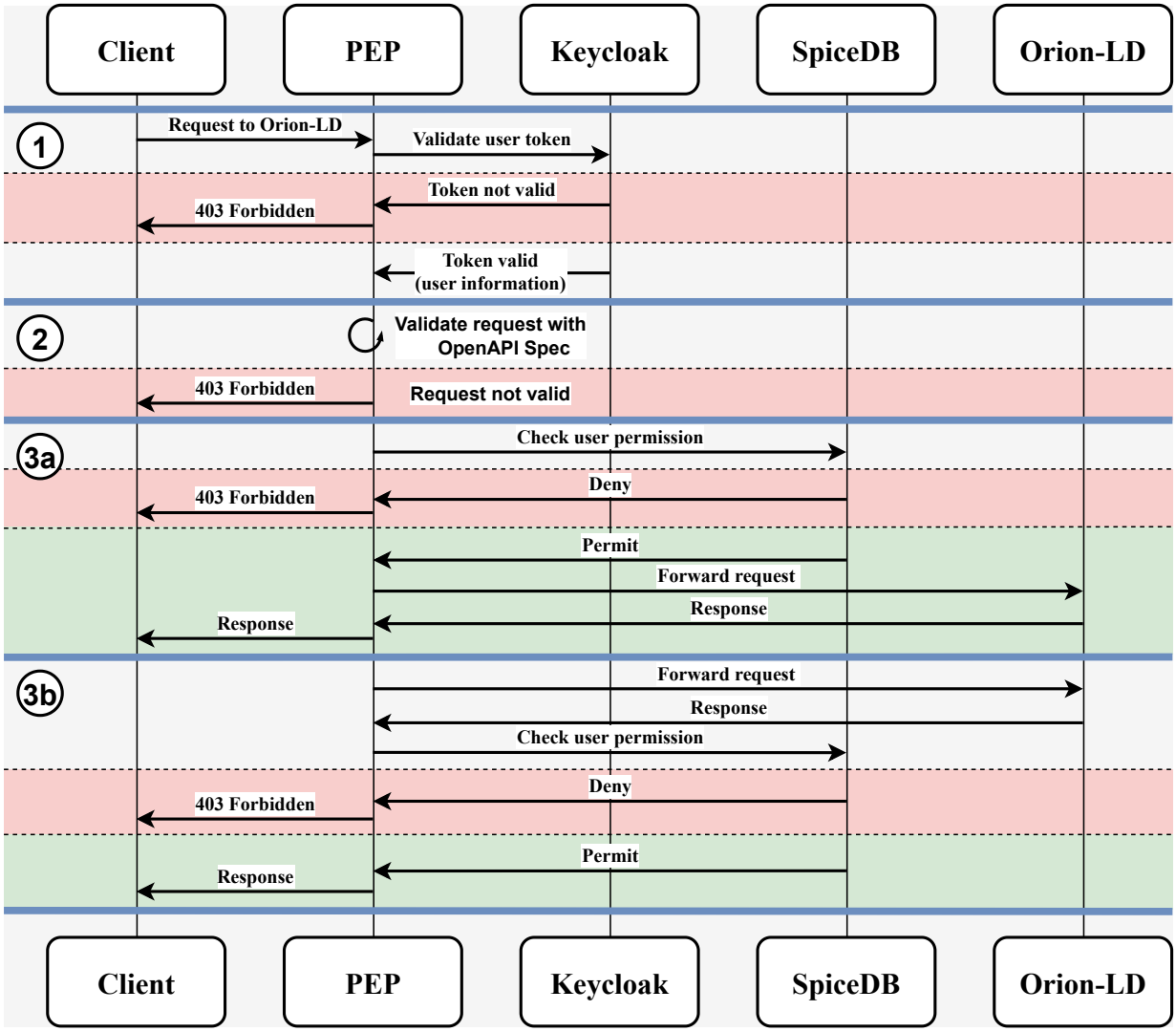

Figure 4.4: Authentication and authorization flow

specification of the NGSI-LD API. (3) The incoming request HTTP verb (POST, GET, PUT, PATCH and DELETE), its path and contents, and the user information from Keycloak will be used to derive the necessary information required to perform authorization checks against SpiceDB. As in some case not all required information are present, this gives two scenarios for this ACL check:

(3a) The request contains all the necessary information to perform an ACL check. The PEP will generate the check request, send it to the PDP and if the action is authorized, it will forward the original request to Orion-LD and send back the result to the client.

(3b) The request does not contain sufficient information, e.g., a request for listing all elements of a specific type does not contain the digital twin ID. As the PEP cannot generate the check request, it will first forward the original request to the context broker, then it will issue ACL check

requests to the PDP and filter the result. Finally, it will send the filtered version of the result to the client.

In NGSI-LD, commands are represented as attributes on digital twin. To execute a command that will be dispatched to an IoT devices, the NGSI-LD request is an update of that attributes with a random value. It is therefore not possible to easily differentiate between the updates of a digital twin's attributes and an actual command on the digital twin. For this reason, the choice was made to not differentiate the commands and the attributes updates. A solution to that could be to update the NGSI-LD API to support an additional information on attributes that contain command, or to provide a specific path to differentiate between the two. As the components used are heavily dependent of the NGSI-LD API, this modification is not done in the scope of this project.

4.2 Implementation

In a context of *Proof of Concept* (PoC), each component of the solution is deployed as a Docker container, in an Ubuntu 20.04 Virtual Machine. The deployment is done using Docker compose and Dockerfile, but it remains manual as no orchestrator was used. Figure 4.5 depicts the architecture of the solution.

The main benefits of using containers for the deployment of the components are:

- Portability: each component can be deployed anywhere (VM, Cloud provider, etc.)
- High availability and scalability: using orchestrators, we can run multiple replicas of containers, including master nodes and load balance the traffic.

It is worth noting that a deployment of the solution in AWS, avoiding coupling with SaaS services, is in progress within another project. That deployment, among others, focuses on deployment automation and containers orchestration.

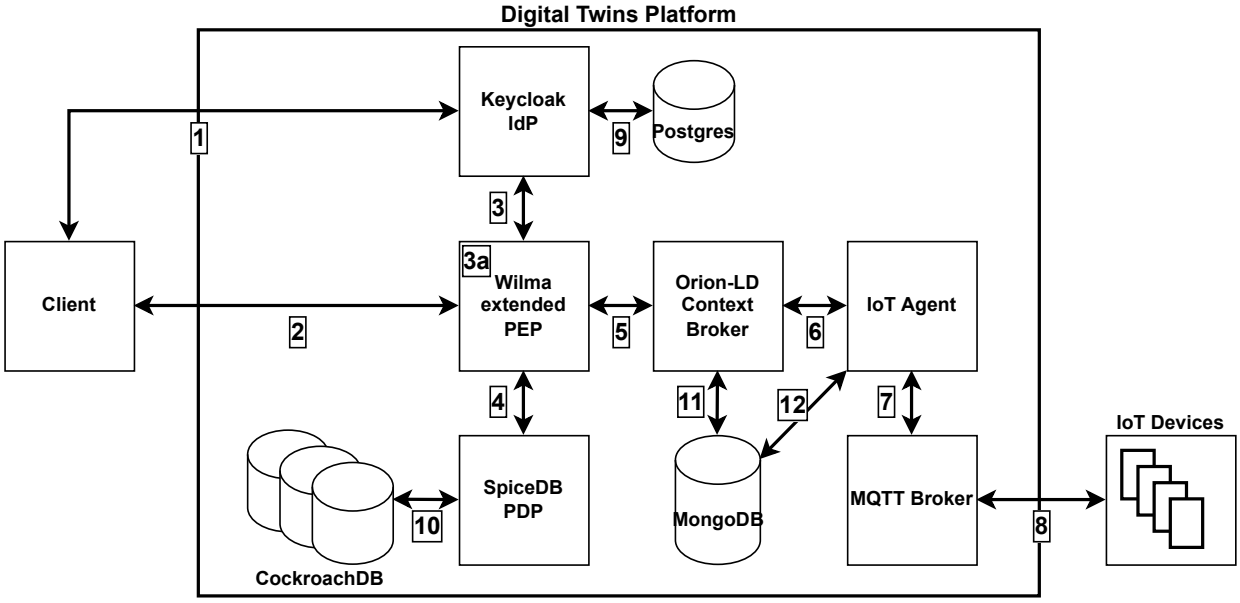

Figure 4.5: PoC architecture

For security purpose, each communication channel should be encrypted with TLS. For the purpose of the PoC, the ability to use TLS was checked but not necessarily enforce everywhere. This was done to remove the burden to handle TLS certificates and to avoid spending excessive time in configuration of the components. In a real deployment, TLS will be enforced everywhere.

The client is an Open-Source, cross-platform API client for GraphQL, REST, and gRPC named Insomnia. It is used to request User Access Token to the IdP and to interact with the context broker through the PEP.

The IoT devices are simulated by a simple NodeJS application. This application publishes MQTT messages towards the IoT Agent.

The IoT Agent is a component from FIWARE, it uses an MQTT broker to receive messages from IoT devices, and communicate with Orion-LD to update the corresponding Digital Twins. It maintains the device data in a MongoDB database.

Orion-LD is the main context broker from FIWARE, it offers an HTTP REST API to create, update, delete and control the Digital Twins. It receives requests from client through the PEP and from the IoT Agent. It maintains the Digital Twins data in a MongoDB database.

Keycloak is the IdP deployed. It uses a Postgres database to store its data. It was configured to use OAuth 2.0, and to authenticate client and deliver User Access Token. The OAuth 2.0

Password Grant type is currently used to easily get User Access Token from the client. As the client application collect the user's password and send it to the authorization server, it is not recommended for production. This grant type is deprecated and should be replaced with the Authorization Code grant type with the PKCE extension.

SpiceDB is the PDP deployed. It uses a CockroachDB database to store its data. It offers a gRPC endpoint to maintain the ACL and to do ACL checks and it requires a Bearer token to identify the components communicating with it.

The PEP is a NodeJS middleware based on the FIWARE Wilma PEP component. It was updated to use Keycloak as an IdP and SpiceDB as a PDP. Furthermore, its internal flow was updated to generate the ACL request towards SpiceDB, and additional checks to validate the request has been implemented. It uses the *express-openapi-validator* library to validate the NGSI-LD request against the NGSI-LD OpenAPI Specification.

The standard flow is (using numbers from Figure 4.5):

1. The client requests a token to the IdP using an OAuth2.0 flow.
2. Once the client gets the token, it can send a NGSI-LD request to the PEP with the token attached in the authorization header.

```
➜  ~ curl --request DELETE \
  --url pep/ngsi-ld/v1/entities/urn:ngsi-ld:building:TourTest \
  --header 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOi
NuU3dGdUlDOEtjV3U2akF1OE5QYmNZb0NLczN1N0tJIn0.eyJleHAiOjE2NjA4OT
oiMTQzNmFjMWUtNDI2OS00M2MyLWJhN2ItOWUxNjhiOTI3YmRkIiwiaXNzIjoiaH
.....................................lI8fqueqSzn7FDfd08LItMtU2P3SkK-
```

Figure 4.6: Example of an NGSI-LD request with the User Access token in its Authorization header

3. If the token is valid, the PEP reaches to the User info endpoint of Keycloak to get user details including the user ID and the user company. Else the request is rejected.

```
➜  ~ curl --request GET \
  --url http://localhost:8082/realms/master/protocol/openid-connect/userinfo \
  --header 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkI
TQzNmFjMWUtNDI2OS00M2MyLWJhN2ItOWUxNjhiOTI3YmRkIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9z
IiwidHlwIjoiQmVhcmVyIiwiYXpwIjoicG9jLXBlcCIsInNlc3Npb25fc3RhdGUiOiI5MDBmZWYxYS0
7InJvbGVzIjpbImRlZmF1bHQtcm9sZXMtbWFzdGVyIiwib2ZmbGluZV9hY2Nlc3MiLCJ1bWFfYXV0aC
ZpZXctcHJvZmlsZSJdfX0sInNjb3BlIjoiZW1haWwgcHJvZmlsZSIsInNpZCI6IjkwMGZlZjFhLTQ5Z
mVkX3VzZXJuYW1lIjoiYWxpY2UiLCJnaXZlbl9uYW1lIjoiQWxpY2UiLCJmYW1pbHlfbmFtZSI6IkVj
drfNaVohlQ2Oha625feIYGm9YfdizHu8uJWfPyZvJ06lWaIoHmhsNy8PJtCclZ_2XnkszH7u6RAF91E
FDfd08LItMtU2P3SkK-3wKnY4npAuwkslNkk8CYNpS5QQPd6MmenBg' | jq
{
  "sub": "c172a6cd-b23e-473a-b62a-06cb9643b7e6",
  "email_verified": true,
  "name": "Alice Ecila",
  "groups": [
    "LK"
  ],
  "preferred_username": "alice",
  "given_name": "Alice",
  "family_name": "Ecila",
  "email": "alice@poc.ch"
}
```

Figure 4.7: Example of the user information retrieved from Keycloak

3a. Before going to the next step, the PEP validates the NGSI-LD request and check if it follows the NGSI-LD OpenAPI Specification.

4. If both the token and the request are valid, the PEP generates an ACL check request and sends it to the PDP for an authorization decision.

5. If the PDP authorize the action the user wants to do on the specified resource(s), the PEP route the request to the context broker and the specified action is executed.

Note that step 4. and 5. can be inverted for some NGSI-LD read request because they do not have sufficient information on the resource(s) targeted. For example a request to retrieve all element of type building does not contain resource ID. In that case, the PEP will first retrieve the information from Orion-LD and then execute ACL checks on the received resource ID. Finally, it will filter the results and send the filtered result to the client.

During the development and the implementation of the PoC, some issues were encountered. The documentation of FIWARE component is sparse when it exists. It was difficult to deploy the components and make them interact with each other. Furthermore, their configurations are also difficult as they are not always properly documented. Not all components handle TLS in the same way, duplicating the effort to correctly configure the system.

4.2.1 Security assessment

We will now evaluate the security of the proposed solution. To do so, we will use the GSMA IoT Security Assessment [17] which is a set of guidelines focused on IoT infrastructure and devices. It provides a detailed guide with advice on how to follow these guidelines. This assessment framework was chosen because it is mature and simple to use. As this is a Proof of Concept, a thorough risk and threat assessment was not done. But the next step would be to use a common framework as OCTAVE [22] or STRIDE [2] to assess the system fully and thoroughly.

The GSMA IoT Security Assessment Checklist was used to find the potential security flaws in the system and propose improvements to mitigate them.

The following point were identified as critical and potentially not fully or only partially implemented : Privacy, root of trust, authentication of components with each other, and security of systems exposed to public internet.

Privacy considerations

Depending on the use case of Digital Twins, they might contain personal data. Based on our scenario for the PoC, no personal data are used or stored. In the case this would change, a privacy compliance process should be created and the necessary means to ensure privacy and compliance should be used.

Root of trust

During the development of the PoC, the TLS certificates were generated manually. This process should not be done in a real system as it is not manageable and can cause severe security issues. To prevent that, a *Public Key Infrastructure* (PKI) should be used. A PKI such as Vault by Hashicorp [30] can be integrated to the system to handle the creation, revocation, rotation, and management of certificates.

Components authentication to each other

Currently, only SpiceDB requires a token to authenticate the users of its API, and the PEP requires clients' requests to be authenticated. The other components do not enforce that. Furthermore, although FIWARE components support TLS, they do not offer a lot of configuration possibilities and definitely not mutual TLS. As the system is containerized, we could use a microservice or service mesh approach to implement mutual TLS and have all components authenticate to each other.

Systems exposed to public internet

In our system, Keycloak and the PEP are exposed to the public internet. They should therefore implement protection such as load balancing or protection against DDoS to maintain the availability of the system. As the system will be deployed in a cloud infrastructure an API Gateway can be used to implement these protections. That way, this gateway will be the only exposed component to public internet.

In addition to the above points, it should be noted that in the context of this PoC not all components were configured and/or installed respecting the best practices for a production environment. This point should be considered in a real deployment of this system.

4.3 Results

The PoC is implemented and functioning. Authenticated users can perform actions on the Digital Twins platform according to their permissions.

In Figure 4.8, we can see the result of a request without using the access control system. This request will list all element of type Building contained in the context broker. It will be a reference to see what results authenticated and authorized user can get.

```
➜  ~ curl --request GET \
  --url 'http://localhost:1026/ngsi-ld/v1/entities/?type=Building&attrs=name&options=keyValues' \
  --header 'Accept: application/json' \
  --header 'Link: <https://smartdatamodels.org/context.jsonld>; rel="http://www.w3.org/ns/json-ld#context"; type="application/ld+json"' | jq
[
  {
    "id": "urn:ngsi-ld:building:TourOpale",
    "type": "Building",
    "name": "Tour Opale"
  },
  {
    "id": "urn:ngsi-ld:building:TourBalex",
    "type": "Building",
    "name": "Tour Balexert"
  },
  {
    "id": "urn:ngsi-ld:building:TourTest",
    "type": "Building",
    "name": "Tour Test"
  }
]
```

Figure 4.8: Result of the NGSI-LD request to list all element of type Building without using the access control system

Figure 4.9 shows an access token for the user Alice and its decoded version. Note that the “sub” field correspond to the user ID and it is this information that is used in SpiceDB.

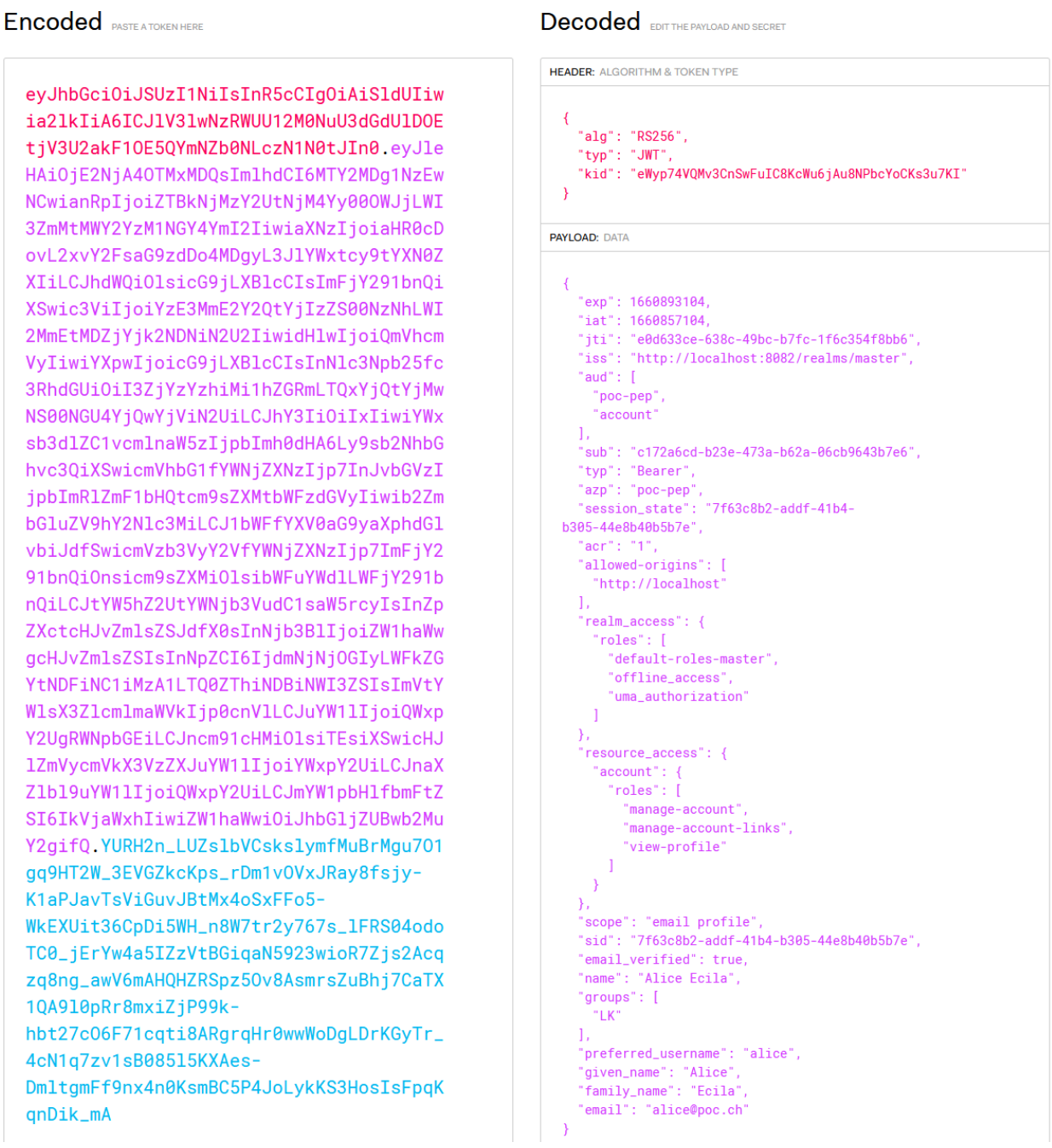

Figure 4.9: Access token of Alice in JWT form and decoded

Figure 4.10 shows the building to which Alice has read access in SpiceDB. She only has access to the building TourBalex.

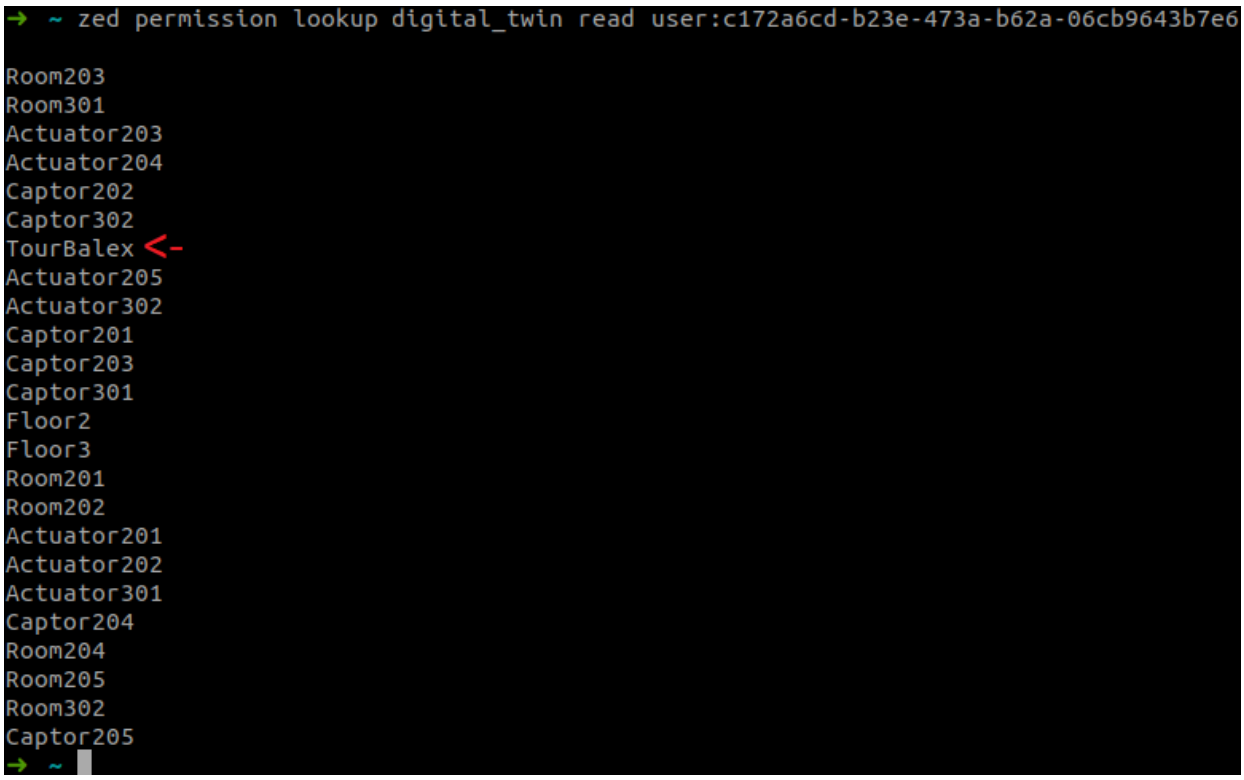

Figure 4.10: Request to check which Digital Twin Alice has permission read on

Finally, Figure 4.11 shows the result of the same request but this time going through the access control and authenticated as the user Alice. We can see that the result is correctly reduced to return only the building Alice has read permission on.

Figure 4.11: Result of the NGSI-LD request to list all element of type Building for the authenticated user Alice

As seen in the security evaluation section, although the system is working as intended, there are some security flaws. Most of the flaws can easily be mitigated by using industry standards component such as a PKI, an API gateway, and using a service mesh approach to enforce mutual TLS on every component. Furthermore, it is paramount to correctly configure all components with their best practices for a production environment.

Chapter 5

Conclusion

The goal of this work is to design a security concept for a Digital Twins platform. It has been identified that such platform requires an efficient, flexible and fine-grained access control system. The current implementations of Digital Twins platform offer security concepts, but they are limited. In the Microsoft's platform Azure Digital Twins, an RBAC system is used, and it does not allow an enough fine-grained access control. In the FIWARE platform, a group of components are dedicated to security. Its access control system uses ABAC and XACML files. It allows fine-grained access control but the use of XACML makes it less flexible as it is not easy to write, update and maintain XACML rules.

For these reasons, an approach based on the FIWARE platform with updated security components are proposed. The Policy Decision Point of FIWARE is replaced with SpiceDB, an Open-Source implementation of Zanzibar, Google's Consistent and global authorization system. Furthermore, the Identity provider from FIWARE is replaced with the Open-Source, mature and efficient Identity provider Keycloak. Finally, the Policy Enforcement Point that handles requests authentication and authorization was improved to connect with the new components and to further increase the validation of incoming requests.

A Proof of Concept was successfully developed and used with a simple use-case to verify that the system works as intended. The implementation of the PoC revealed some pain points. The documentation of the FIWARE components is sparse and sometimes even lacking which make the usage and configuration of these components hard and prone to security errors. Additionally, although SpiceDB does not require to write XACML policies and offers enough flexibility to model all kind of use-case, it is not easy to design the correct schema.

After the implementation, a first security evaluation of the system was done and it revealed some security flaws in the design. There is currently no root of trust for the TLS certificates, the components do not authenticate to each other, and the components exposed to the public internet need to be hardened. These flaws can be fixed by using best practices such as using a PKI to manage certificates, and use a service mesh approach to enforce mutual TLS on all the

components.

Even if the proposed approach seems promising to secure complex relational entities ecosystem, it comes with the costs of creating and maintaining multiples models. In fact, on one hand, we have the NGSI-LD model that define the digital twins of the physical objects, and their relations. And on the other hand, we have to model such relations and access policies in the SpiceDB schema.

To foster the adoption of our security concept further research should focus on a common model for both Digital Twin modelization and access control management. We could extend the NGSI-LD schema to integrates security properties and SpiceDB to apply policies on such properties. This makes sense as the owner of a resource, or at least the people that design and create the digital twin has the knowledge to define which access control is needed.

Bibliography

[1] *A Survey of Industry Data Models and Reference Data Libraries*. DT Hub Community. URL: https://digitaltwinhub.co.uk/archive/a-survey-of-idms-and-rdls-telecommunication_ontologies_for_dts/ (visited on 07/18/2022).

[2] Archiveddocs. *The STRIDE Threat Model*. URL: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20) (visited on 08/18/2022).

[3] *Azure Digital Twins Documentation - Tutorials, API Reference*. URL: https://docs.microsoft.com/en-us/azure/digital-twins/ (visited on 07/20/2022).

[4] Martin Bauer, Flavio Cirillo, Jonathan Fürst, Gürkan Solmaz, and Ernö Kovacs. "Urban Digital Twins – A FIWARE-based model". In: *at - Automatisierungstechnik* 69.12 (Dec. 1, 2021). Publisher: De Gruyter (O), pp. 1106–1115. ISSN: 2196-677X. DOI: 10.1515/auto-2021-0083. URL: https://www.degruyter.com/document/doi/10.1515/auto-2021-0083/html (visited on 07/07/2022).

[5] Trevor Braun, Benjamin C. M. Fung, Farkhund Iqbal, and Babar Shah. "Security and privacy challenges in smart cities". In: *Sustainable Cities and Society* 39 (2018), pp. 499–507. ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2018.02.039. URL: https://www.sciencedirect.com/science/article/pii/S2210670717310272.

[6] *Building your own IoT platform using FIWARE GEis*. 2017. URL: https://www.fiware.org/wp-content/uploads/2017/01/How-different-GEs-integrate-together.-sample-architectures.pdf (visited on 08/02/2022).

[7] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. "Spanner: Google's Globally-Distributed Database". In: *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*. Hollywood, CA: USENIX Association, Oct. 2012, pp. 261–264. ISBN: 978-1-931971-96-6. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/corbett.

[8] Fan Deng, Zhenhua Yu, Wenjing Liu, Xiaoqing Luo, Yu Fu, Ben Qiang, Chaoyang Xu, and Zhiwu Li. "An efficient policy evaluation engine for XACML policy management". In: *Information Sciences* 547 (2021), pp. 1105–1121. ISSN: 0020-0255. DOI: https://doi.org/

10.1016/j.ins.2020.08.044. URL: https://www.sciencedirect.com/science/article/pii/S0020025520308148.

[9] *Digital Twins Definition Language*. original-date: 2019-04-30T23:42:55Z. URL: https://github.com/Azure/opendigitaltwins-dtdl (visited on 07/20/2022).

[10] Martin J. Dürst and Michel Suignard. *Internationalized Resource Identifiers (IRIs)*. Request for Comments RFC 3987. Num Pages: 46. Internet Engineering Task Force, Jan. 2005. DOI: 10.17487/RFC3987. URL: https://datatracker.ietf.org/doc/rfc3987 (visited on 07/18/2022).

[11] Erich Barnstedt, Birgit Boss, Erich Clauer, Dan Isaacs, Shi-Wan Lin, Somayeh Malakuti, Pieter van Schalkwyk, and Thiago Weber Martins. "Open Source Drives Digital Twin Adoption - Journal of Innovation: March 2021". In: *Open Source Drives Digital Twin Adoption* (IIC Journal of Innovations Mar. 2021). URL: https://www.iiconsortium.org/pdf/2021_March_JoI_Open_Source_Drives_Digital_Twin_SA.pdf (visited on 07/07/2022).

[12] *ETSI - Welcome to the World of Standards!* ETSI. URL: https://www.etsi.org/ (visited on 07/20/2022).

[13] *ETSI GS CIM 009 - V1.5.1 - Context Information Management (CIM); NGSI-LD API*. ETSI GS CIM 009 - V1.5.1 - Context Information Management (CIM); NGSI-LD API. Nov. 2021. URL: https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.05.01_60/gs_CIM009v010501p.pdf (visited on 07/20/2022).

[14] *eXtensible Access Control Markup Language (XACML) Version 3.0*. Jan. 22, 2013. URL: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (visited on 07/28/2022).

[15] *FIWARE - Open APIs for Open Minds*. URL: https://www.fiware.org/ (visited on 07/20/2022).

[16] *Fiware for Digital Twins - Position paper*. FIWARE. URL: https://www.fiware.org/wp-content/uploads/FF_PositionPaper_FIWARE4DigitalTwins.pdf (visited on 07/18/2022).

[17] *GSMA IoT Security Guidelines and Assessment*. Internet of Things. URL: https://www.gsma.com/iot/iot-security/iot-security-guidelines/ (visited on 08/18/2022).

[18] Mark Hearn and Simon Rix. "Cybersecurity Considerations for Digital Twin Implementations - Journal of Innovation: November 2019". In: *Cybersecurity Considerations for Digital Twin Implementations* (IIC Journal of Innovations Nov. 2019). URL: https://www.iiconsortium.org/news-pdf/joi-articles/2019-November-JoI-Cybersecurity-Considerations-for-Digital-Twin-Implementations.pdf (visited on 07/07/2022).

[19] David Holmes, Maria Papathanasaki, Leandros Maglaras, Mohamed Amine Ferrag, Surya Nepal, and Helge Janicke. "Digital Twins and Cyber Security – solution or challenge?" In: *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. 2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM). Sept. 2021, pp. 1–8. DOI: 10.1109/SEEDA-CECNSM53056.2021.9566277.

[20] Authzed Inc. *authzed: the managed permissions database for everyone.* URL: https://authzed.com/ (visited on 08/13/2022).

[21] Authzed Inc. *The One Crucial Difference Between Spanner and CockroachDB.* Section: blog. Oct. 14, 2021. URL: https://authzed.com/blog/prevent-newenemy-cockroachdb/ (visited on 08/13/2022).

[22] *Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process.* URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=8419 (visited on 08/18/2022).

[23] Michael Jacoby and Thomas Usländer. "Digital Twin and Internet of Things—Current Standards Landscape". In: *Applied Sciences* 10.18 (Jan. 2020). Number: 18 Publisher: Multidisciplinary Digital Publishing Institute, p. 6519. ISSN: 2076-3417. DOI: 10.3390/app10186519. URL: https://www.mdpi.com/2076-3417/10/18/6519 (visited on 07/07/2022).

[24] Seungmyeong Jeong, Seongyun Kim, and Jaeho Kim. "City Data Hub: Implementation of Standard-Based Smart City Data Platform for Interoperability". In: *Sensors (Basel, Switzerland)* 20.23 (Dec. 7, 2020), p. 7000. ISSN: 1424-8220. DOI: 10.3390/s20237000. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7731156/ (visited on 07/07/2022).

[25] *JSON-LD 1.1.* URL: https://www.w3.org/TR/json-ld/ (visited on 07/18/2022).

[26] Noboru Koshizuka, Stephan Haller, and Ken Sakamura. "CPaaS.io: An EU-Japan Collaboration on Open Smart-City Platforms". In: *Computer* 51.12 (Dec. 2018). Conference Name: Computer, pp. 50–58. ISSN: 1558-0814. DOI: 10.1109/MC.2018.2880019.

[27] Ruoming Pang, Ramon Caceres, Mike Burrows, Zhifeng Chen, Pratik Dave, Nathan Germer, Alexander Golynski, Kevin Graney, Nina Kang, Lea Kissner, Jeffrey L. Korn, Abhishek Parmar, Christina D. Richards, and Mengzhi Wang. "Zanzibar: Google's Consistent, Global Authorization System". In: *2019 USENIX Annual Technical Conference (USENIX ATC '19)*. Renton, WA, 2019.

[28] Peter Salhofer. *Evaluating the FIWARE Platform.* Jan. 3, 2018. ISBN: 978-0-9981331-1-9. URL: http://hdl.handle.net/10125/50615 (visited on 07/20/2022).

[29] *SpiceDB and Authzed | authzed.* URL: https://docs.authzed.com/ (visited on 08/13/2022).

[30] *Vault by HashiCorp.* Vault by HashiCorp. URL: https://www.vaultproject.io/ (visited on 08/18/2022).