Microservices Authentication and Authorization from a German Insurances Perspective

Arne Koschel
Andreas Hausotter
Hochschule Hannover
University of Applied Sciences & Arts
Faculty IV, Department of Computer Science
Hannover, Germany
Email: {arne.koschel, andreas.hausotter}
@hs-hannover.de

Pascal Niemann
Christin Schulze
Hochschule Hannover
University of Applied Sciences & Arts
Faculty IV, Department of Computer Science
Hannover, Germany
Email: {pascal.niemann, christin.schulze}
@stud.hs-hannover.de

***Abstract*—Even for the more traditional insurance industry, the *Microservices Architecture (MSA)* style plays an increasingly important role in provisioning insurance services. However, insurance businesses must operate legacy applications, enterprise software, and service-based applications in parallel for a more extended transition period. The ultimate goal of our ongoing research is to design a microservice reference architecture in cooperation with our industry partners from the insurance domain that provides an approach for the integration of applications from different architecture paradigms. In Germany, individual insurance services are classified as part of the critical infrastructure. Therefore, German insurance companies must comply with the Federal Office for Information Security requirements, which the Federal Supervisory Authority enforces. Additionally, insurance companies must comply with relevant laws, regulations, and standards as part of the compliance requirements. Note: As Germany is considered relatively strict with respect to the privacy and security demands, meeting these requirements may well be suitable (if not even "over-fulfilling") for insurance companies in other countries. The question raises thus of how insurance services can be secured in an application landscape shaped by the MSA style to comply with the architectural and security requirements depicted above. This article highlights the specific regulations, laws, and standards the insurance industry must comply with. We present conceptual approaches for authentication and authorization in a MSA tailored to the requirements of our insurance industry partners. In particular, we focus on different architectural patterns for service-level authorization as well as approaches for service-level authentication and discuss their advantages and disadvantages.**

Keywords*—*Security; Authorization; Authentication; Insurance Industry; Microservices Architecture.

I. Introduction

In this article, which is an extended version of our previous work [1], we look at *Information Technology (IT)* security within a microservices-based reference architecture for at least German insurance companies. IT security is absolutely a "must-have" for insurance companies, especially for customer data, self-written and third-party applications, and their IT infrastructure in general. General regulations, such as the European *General Data Protection Regulation (GDPR)* [2], are applied to the insurance domain, as well as insurance-specific laws and rules regarding security and other regulations [3] [4], for example, data protection and secured IT communication infrastructure. This article mainly focuses on securing insurance business applications [5]. Over time, several technologies from monolithic mainframe applications, functional decomposition-based software, traditional *Service-Oriented Architecture (SOA)*, and third-party enterprise software, such as SAP systems, were and are used together in insurance business applications.

Recently, the *Microservices Architecture (MSA)* style [6] [7] and cloud computing joined the field. The ultimate goal of our currently ongoing research [8] is to develop a "*Microservice Reference Architecture for Insurance Companies (RaMicsV)*" jointly with partner companies from the insurance domain, which is taking all those typical cornerstones from (overtime grown) insurances into account. Placed within our work on RaMicsV is the question: "how to help secure (insurance) business applications using potentially several logical parts from RaMicsV, mainly including microservices combined with other typical insurance applications technologies"?

Only a few authors (see Section II) look at such technology combinations, and they especially do not take (German) insurance domain specifics into account. Thus, the present article constitutes an initial step in that direction.

In particular, we contribute here our ongoing work and initial results regarding:

- An introduction to IT security regulations in Germany for insurance companies, including:
  - A brief explanation of when an institution is considered critical infrastructure and the resulting consequences.
  - Functions and regulations of the *Federal Office of Information Security (BSI)* and the *Federal Financial Supervisory (BaFin)* in this context.
- Evaluate existing patterns for achieving protection goals and weigh their advantages and disadvantages.

- Take a look at properties of service- and edge-level authentication and authorization.
- An overview of approaches to service-level authentication.
- Consider patterns concerning the requirements of the insurance industry with SOA and an *Enterprise Service Bus (ESB)*.

Especially our Sections V, VI, and VII include several new illustrations and are altogether more detailed than our previous work from [1].

The remainder of this article is structured as follows: After looking at related work in Section II, we place our current work into our initial logical reference architecture from [8] in Section III. Next, Section IV looks at requirements for German insurance companies. Initial work to meet those requirements is contributed in Section V, which discusses edge- vs. service-level authorization and authentication, in Section VI, which examines authorization patterns, and in Section VII, which details authentication patterns. Both parts are evaluated concerning their potential application within our overall work. Finally, Section VIII summarizes our results, draws a conclusion, and looks at future work.

II. Related Work

Our research is based on the literature of well-known authors in microservices, especially Chris Richardson (Microservices Pattern) [6]. His book describes fundamental statements for the advantages and disadvantages of the edge-level security pattern and the service-level security pattern.

We adopted our definition of components for authorization and authentication from the *National Institute of Standards and Technology (NIST)* [9] and the patterns described in Sections V and VI originate from [10]. Furthermore, [10] discusses service-level authentication through mutual transport layer security and a token-based approach, whereas Section VII also briefly discusses this and adds Hypertext Transfer Protocol and Password Authenticated Key Exchange to this topic. In contrast, this paper uses the above content to a certain extent and places it in the context of our reference architecture and the legal scope of our partners in a German insurance company.

Kai Jander et al. compare general transport layer security, transport layer security with service and microservice frameworks for authentication and encryption of microservices. They provide an overview of password authentication, symmetric keys and key pairs, and then present an implementation of a password authenticated key exchange [11]. In contrast, this paper uses a different scope and establishes a connection to legal regulations for German insurance companies.

Regarding legal regulations and specifications, we use, among others, *the Act on Federal Office for Information Security (BSIG)* [12]. Especially the part for critical infrastructures and, accordingly, *the Regulation for the Determination of Critical Infrastructures according to the BSI Act (BSI-KritisV)* [13] is used to underpin the relevance of our reference architecture. In addition, this is supplemented with *the Insurance Regulatory Requirements for IT (VAIT)* [3] published by the BaFin, as this is the responsible authority of the insurance industry.

In our previous work [8], we presented the logical microservices reference architecture that we created in the German insurance domain with our partners by logical and technical details in the area of logging and monitoring components. So far, components in the area of security have not been considered within this reference architecture, which is now started in the present article.

Additionally, in [14], we dealt with the consistency of microservices, among other things. Here, compliance aspects were described, which arose during the service design using Domain Driven Design. The requirements specific to German insurance companies were briefly mentioned. Based on this, the legal constraints and controlling constitutions are described in more detail.

To the authors' knowledge, this is the first work to address the legal regulations for German insurance companies in the context of a reference architecture for microservices with a focus on patterns for security and, in particular, authentication and authorization. In addition, we address the requirement of this reference architecture for microservices to work together or side by side with an ESB (see Section III).

III. Reference Architecture for Insurance Companies

This section will present our logical reference architecture for microservices in the insurance industry (RaMicsV).

RaMicsV defines the setting for the architecture and the design of a microservices-based application for our industry partners. The application's architecture is out of scope, as it heavily depends on the specific functional requirements.

When designing RaMicsV, a wide range of restrictions and requirements given by the insurance company's IT management have to be taken into account. Concerning this contribution, the most relevant are:

- ESB: The ESB, as part of the SOA, must not be questioned. It is part of a successfully operated SOA landscape, which seems suitable for our industry partners for several years to come. Thus, from their perspective, the MSA style is only suitable as an additional enhancement and only a partial replacement of parts from their SOA or other self-developed applications.
- Coexistence: Legacy applications, SOA, and microservices-based applications will be operated in parallel for quite an extended transition period (several years to come). This means that RaMicsV has to provide approaches for integrating applications from different architectural paradigms – looking at it from a high-level perspective, allowing an "MSA style

best-of-breed" approach at the enterprise architectural level as well.

Figure 1 depicts the building blocks of RaMicsV, which comprises layers, components, interfaces, and communication relationships. Components of the reference architecture are colored yellow; those out of scope are greyed out.

A component may be assigned to one of the following *responsibility areas*:

- **Presentation** includes components for connecting clients and external applications such as SOA services.
- **Business Logic & Data** contains the set of microservices to provide the desired application-specific behavior.
- **Governance** consists of components that contribute to meeting the IT governance requirements of our industrial partners.
- **Integration** contains system components to integrate microservices-based applications into the industrial partner's application landscape.
- **Operations** consist of system components to realize unified monitoring and logging, which encloses all systems of the application landscape.
- **Security** consists of components to provide the goals of information security, i.e., confidentiality, integrity, availability, privacy, authenticity & trustworthiness, non-repudiation, accountability, and audibility.

Components communicate via HTTP—using a RESTful API, or message-based—using a *Message-Oriented Middleware (MOM)* or the ESB. The ESB is part of the *integration* responsibility area, which contains a message broker (see Figure 1).

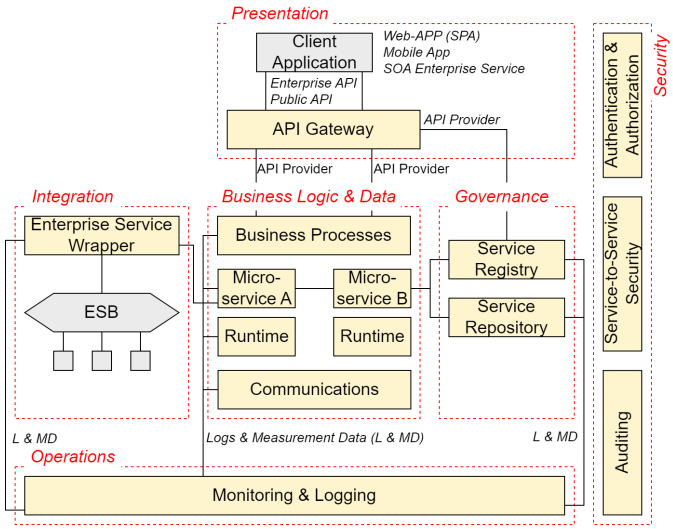

Fig. 1. Building Blocks of the Logical Reference Architecture RaMicsV

In addition to data transformation and message routing and delivery, an ESB also implements security policies. For example, WS02 ESB supports *Web Services (WS)*-Security and WS-Policy specifications [15]. Beyond that, the WSO2 Identity Server can be used to generate an *OAuth Base Security Token* that microservices may employ to authenticate and authorize client applications and API clients. This corresponds to the edge- level authentication & authorization depicted in Section V.

In the next sections, we will look at the *security* responsibility area.

IV. Requirements for German Insurance Companies

Security is a fundamental aspect of any software architecture and should never be neglected, mainly when there is a legislative framework where specific regulations exist. In Germany, insurance companies, regarded as critical infrastructure, are obligated to comply with the requirements of the BSIG, which the BaFin enforces. The Federal Office for Information Security has determined this consideration. Note: In our work, we did not look at regulations and legal requirements in other countries, but, as stated above, German regulations are seen as "somewhat tough" already.

A. Federal Office for Information Security and Critical Infrastructures

The BSI is a federal agency in Germany responsible for security standards inside federal authorities and is a central reporting point for security incidents. Companies that are running critical infrastructures are obligated to report to the BSI. The Council of the European Union defined that a critical infrastructure "... is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State ..." [16]. Therefore an ordinance (BSI-KritisV [13]) from 2016 defines critical infrastructures in Germany. It could easily have dramatic consequences for the economy, state, and society if an infrastructure from one of the seven mentioned sectors (energy, water, food, information technology and telecommunications, health, finance and insurance, transport, and traffic) were attacked. Under Section 7 (1) no. 1 to 5, examples are given of critical financial and insurance services, which are of corresponding importance. Some examples mentioned are payment transactions or, among other things, insurance services and social security benefits. However, either a system or a part of it must be assigned to column B (System category) of Annex 6 Part 3 and, at the same time, exceed the corresponding threshold value in column D of the specific metric to be considered critical infrastructure. A general example would be a contract administration system in which the number of life insurance claims per year exceeds 500,000. Therefore, some of our partners' systems are considered critical infrastructure and are liable to other requirements.

Because of the BSIG from 2009 [12], under Section 8a, "Security regarding the information technology of critical infrastructures," institutions with critical infrastructures are obligated to a security standard. They need to provide evidence to the BSI every two years that they took precautionary

measures to achieve the protective goals of IT security. Specifically mentioned are **availability, integrity, authenticity, and confidentiality**. In addition, precautions are described here as reasonable if the effort required to secure the protection goals is in proportion to the consequences of the failure. Moreover, the BSI has published a document [17] that specifies the requirements imposed by Section 8a (1) BSIG.

Section 8a (2) of the BSIG states that it is possible to establish an industry-specific security standard that meets the requirements. The Federal Office of Civil Protection and Disaster Assistance and the corresponding regulatory authority will determine whether this standard is appropriate. Thus, there has to be a Federal Office that determines whether the company is complying with the requirements.

B. Federal Financial Supervisory Authority

The BaFin is responsible for the supervision of banks and financial and insurance providers. They published VAIT [3] in the year 2018. This publication contains the general conditions and specifications for IT risk and security management. There is a reference to the BSI-KritisV, which has an entire section dedicated to critical infrastructures. All aspects are essential, from detection over definition to implementation of security measurements. The goal is to secure the protective objectives of IT security, which are named in Subsection IV-A, and to minimize all risk factors inside the critical infrastructure. Therefore, German insurance companies must provide evidence through audits, certificates, or examinations every two years to fulfill their obligations. That is why every aspect of security needs to be addressed while or even better before implementing new systems.

C. Further Motivation for the Commitment to Confidentiality

There is a wide range of security aspects that need to be addressed. At this point, we would like to refer to a document published by the BSI entitled ”Supervision of critical infrastructures in finance and insurance” [4]. This briefly discusses the legal requirements for critical infrastructures and the introduction of these requirements in 2019. The document states that most of the deficiencies and shortcomings did not pose a direct threat to maintaining the operation of the infrastructures concerned. Nevertheless, according to ISO/IEC 27002, eight percent of the deficiencies were attributable to access control.

Additionally, in 2021 the *Open Web Application Security Project (OWASP)* Top Ten 2021, first place is ”Broken Access Control,” and seventh place is ”Identification and Authentication Failures” [18]. Compared to 2017, ”Broken access control“ came up from place 5 [19]. This shows that the importance of authorization and authentication continues to increase. As a result, it is increasingly important to find mechanisms that protect system boundaries with a low potential for error by business logic development teams.

Concerning Subsections IV-A and IV-B, the four security properties that are explicitly named are listed below:

- **Confidentiality** includes read access by authorized subjects only.
- **Integrity** describes writing access by authorized subjects only.
- **Availability** implies access by authorized subjects at any time.
- **Authenticity** verifies the identity of the sender.

Through conversations with our partners, the focus of this paper will first be on different patterns of the service-level authorization aspects as part of the confidentiality and partly integrity protection goal. Since authorization can be close to authentication in terms of implementation, it will also be included in the following section concerning the implementation location.

V. Authorization and Authentication - Edge- vs. Service-level

In distributed systems, authentication and authorization can be realized at different locations. While there is typically one place where authentication and authorization are performed in monolithic systems, there are various system locations where authentication and authorization might occur in distributed systems. This section describes the fundamental differences in properties when using authorization or authentication for microservices depending on the implementation location.

Authentication and authorization have a crucial difference in the choice of location. As seen in Figure 2, the distinction between the two locations is easy to see. Already recognizable from the name, the service-level is located on the microservices level. On the other hand, the edge-level is the boundary to the outside, represented by the API Gateway.

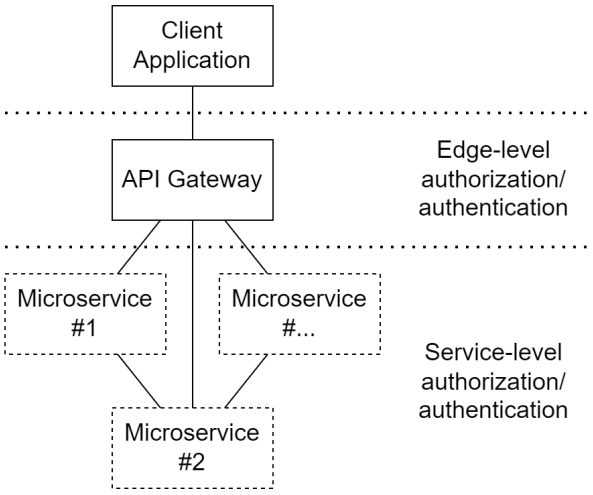

Fig. 2. Visual abstraction of a part of RaMicsV to represent the location of edge- and service-level.

Scalability is the critical factor in positioning authentication, as there is no business reason to prefer edge-level or service-level. Authentication needs a database to check credentials and calculate any security token; domain knowledge is unnecessary [6]. In the case of authorization, on the other hand, it

is not only scalability that is important but also how access is controlled. If *Role-based Access Control (RBAC)* is the only requirement, decisions can be made without domain knowledge, e.g., by roles per URL path. In this case, edge-level authorization is usable. An *Access Control List (ACL)* is called when more explicit authorization is required. In this case, domain information is needed, and service-level authorization is suitable.

This section does not discuss technical authentication and authorization solutions but highlights the authentication and authorization positioning and the resulting properties for the system's performance and development. For both authentication and authorization, two fundamentally different approaches are possible. At the edge-level, the required components are frequently located in an API Gateway, whereas at the service-level, the components are located in each service. In the following section, we first discuss edge-level authentication.

A. Edge-level Authentication

If there is an API Gateway, it may be used for authentication decisions. This is a quick-to-develop but hard-to-scale solution. Using an API Gateway has the following properties [6]:

- Domain logic development teams are very little involved with authentication.
- API Gateway development teams have to deal with more complexity.
- Only one team is responsible for the authentication. This lowers the risk of security vulnerability.
- Faster development by lower complexity.
- Poor scalability due to a single point of control.
- Risk of too strong coupling of API Gateway and microservices; independent deployment is usually impossible.

B. Service-level Authentication

An alternative to the API Gateway implementation is authentication at the service-level. This solution is slow and expensive to develop but scales well. The service-level authentication has the following properties [6]:

- Domain logic development teams have to deal with more complexity.
- Higher risk for security vulnerabilities due to multiple development teams.
- Slower development due to higher complexity in any microservice.
- Higher scalability, which stresses one of the essential properties of an MSA.
- If RBAC is used and only one role exists for a specific microservice, e.g., only the admin, authentication failures play less of a role for this microservice because regular users are not allowed to access it anyway.

The difference between authentication at the edge- and service-level should have become clearer now: Both approaches provide the authentication basis for the protection goals of confidentiality and integrity, which are described in Section IV. There are different strategies to deal with service-level authentication. These will be mentioned in Section VII. In the next section, edge-level and service-level authorization will be presented.

C. Edge-level Authorization

With edge-level authorization, all the logic resides in the API Gateway. This brings the following characteristics:

- Easy implementation and maintenance.
- It may create problems when scaling.
- Designing complex systems can be challenging.
- Back-end microservices must only be accessible via the API Gateway.
- Risk of too strong coupling of API Gateway and microservices—no independent deployment is possible.

This is a suitable solution for a lightweight MSA with few roles. Next, we will look at service-level authorization, which is increasingly attractive for more complex systems [10].

D. Service-level Authorization

Like authentication, authorization can also be implemented at the service-level. An additional component is added to each microservice for authorization, authentication, or both. In this context, the following terms are important (Figure 3) [9]:

- ***Policy Enforcement Point (PEP)*** enforces the authorization decision.
- ***Policy Decision Point (PDP)*** computes the authorization decision.
- ***Policy Administration Point (PAP)*** comprises an interface to administrate the policies.
- ***Policy Information Point (PIP)*** provides additional information for the PDP to make authorization decisions [9].

As shown in Figure 3, the PEP and PDP form the authorization functionality.

The subsequent patterns are determined by the localization of the PEP and PDP in relation to a microservice. PAP and PIP are only mentioned for completeness. At first, we consider the general properties change compared to edge-level:

- Responsibility moves from the API development team to the microservices development team.
- Complex microservices environments are possible.
- Implementation and maintenance are more complex because changes affect each microservice.

Overall, this sets out the fundamentals. In the following section, different patterns regarding service-level authorization are presented. These take an essential role in architectural decisions regarding the use of microservices.

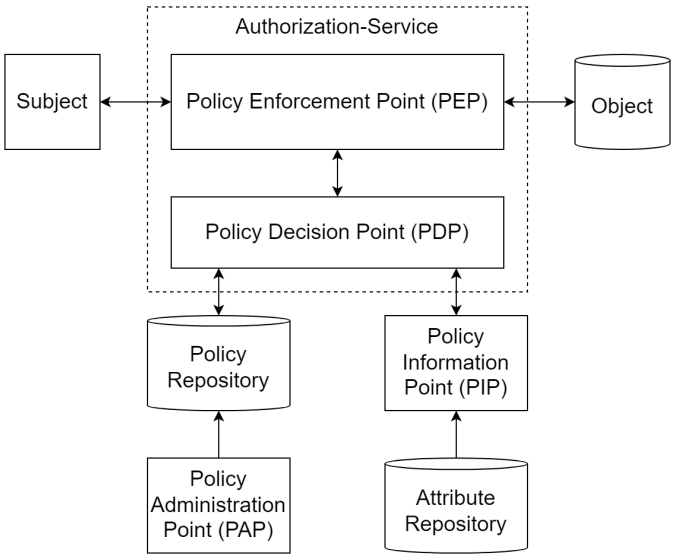

Fig. 3. Fundamental points of ACM [9].

VI. Service-level Authorization - Patterns

There are three different patterns of service-level authorization: Decentralized pattern, centralized pattern with a single PDP and centralized pattern with embedded PDP. Each pattern offers different advantages and disadvantages and differs architecturally. In the following, the three patterns are fundamentally described and architecturally visualized. A brief theoretical evaluation of the properties in the context of our reference architecture RaMicsV will be provided in order to simplify decision-making depending on the context.

A. Decentralized pattern

The decentralized pattern is the solution to create a microservice that is wholly controlled by the development team. All software and data components for making authorization decisions reside inside the microservice. A visualization of this can be seen in Figure 4. This is optimal for scaling, but it requires much effort to implement and maintain since any change in the authorization process requires changes in each microservice. Another challenge is propagating policy or attribute changes to all microservices. The challenge just mentioned becomes even more complex and grows linearly with the increasing number of microservices. This must also take into account that microservices can fail at this point and not receive the information. Thus, ensuring that the information is passed on has a high priority. On the other hand, there are scenarios where this pattern may be suitable, e.g., if there is a microservice with a high number of requests [10]. Furthermore, it has the advantage that no additional functionality needs to be provided by the ESB within RaMicsV since all functionality is contained within the respective microservices themselves.

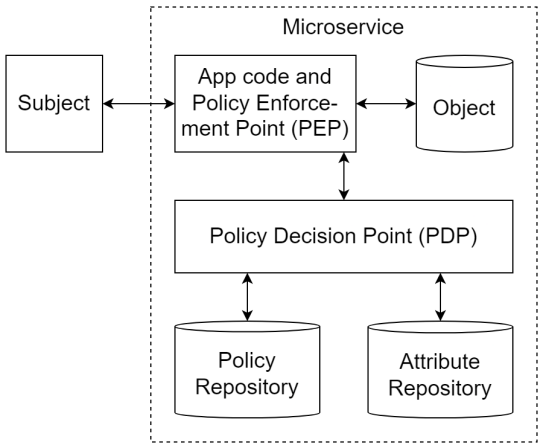

Fig. 4. Service-level Authorization - Decentralized pattern [10].

B. Centralized pattern with single PDP

With the centralized single PDP pattern, the PEP is located within each microservice, and the PDP resides in a different central location, as shown in Figure 5. This implies that every request to the microservice will result in a network call to the PDP. Thus, this is not a suitable solution if a very low response time is required. Also, if high scalability is needed, a single decision point is associated with limitations. In addition, updating policies and attributes is unproblematic due to relocation. Accordingly, the just mentioned things can be updated detached from the microservices.

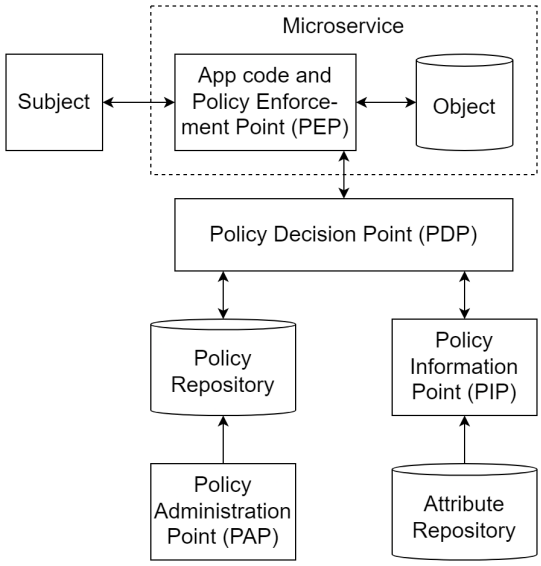

Fig. 5. Service-level Authorization - Centralized pattern with single PDP [10].

However, in the case of a central PDP, all microservices are independent of changes within the PDP. It should be taken into account that possible failures of the latter can have critical consequences. Moreover, this approach could be faster to be implemented in cooperation with a required

ESB (see Section III), because then, the PEP resides in each microservice, and the PDP is provided by the ESB [10].

C. Centralized pattern with embedded PDP

In the centralized pattern with embedded PDP, the data and attributes are centralized, but the PDP is part of each microservice. This is why Figures 5 and 6 are almost identical, since only the content of the microservice changes, and in this case, the PDP is within that service. Unlike the decentralized pattern (see Subsection VI-A), the PDP is not part of the code but is embedded using a microservices library. So, the PDP is part of the microservice for quick decisions, but the development team doesn't have a lot of development work. Similar to the centralized pattern with a single PDP (see Subsection VI-B), unlike the decentralized pattern (see Subsection VI-A), there is no difficulty in propagating policy or attribute changes to all microservices.

For interoperation with the required ESB (see Section III), this pattern combines the advantages of a decentralized pattern and a quick implementation. The ESB could be used for data and attribute sharing. All other components could make fast decisions through the microservices [10]. Concerning the protection goals described in Section IV, the authorization enforces confidentiality and integrity.

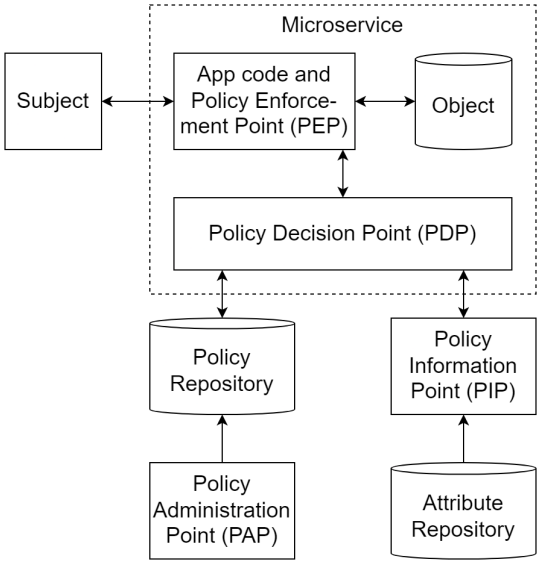

Fig. 6. Service-level Authorization - Centralized pattern with embedded PDP [10].

D. Summary

Insurance companies are running large and complex systems with many different services and fine-grained access control. For this reason, edge-level authorization is suitable only in specific scenarios, for example, if RBAC can be used for a given microservice.

The application landscape of our partners in the insurance industry comprises an ESB as part of the reference architecture (see Section III). Therefore, each pattern has its use case, as we explained above. The decentralized pattern (see Subsection VI-A) is recommended when performance is the most crucial requirement. The centralized pattern with a single PDP (see Subsection VI-B) is suitable if performance is less critical and RBAC is needed. In addition, there is also the possibility that the PDP could be integrated directly within the ESB. The centralized pattern with embedded PDP (see Subsection VI-C) brings together the advantages of the previously mentioned patterns and is, therefore, from our point of view, the most promising one.

Nevertheless, even if service-level authorization can be ensured, the counterpart of authentication still needs to be addressed. Accordingly, within the following section, different approaches of service-level authentication within our context will be described and considered.

VII. Service-level Authentication - Approaches

As in the previous section, this part is based on a theoretical consideration of possible approaches to service-level authentication in the context of the BSI specifications and our partners' reference architecture. With regard to the reference architecture, the aim is to keep the overhead as low as possible by using an appropriate service-level authentication. Accordingly, it does not deal with technical implementations or the precise flow of protocols, but references for more in-depth information are given where appropriate. This is to give a rough overview. The most critical points concerning authentication, which are at the authors' discretion, are highlighted. The sequence of approaches builds on each other in specific parts and tries to solve a posed problem of the previously mentioned approach. In its simplest form, the handling of authentification can be such that it is not taken into account. Instead, essential trust is established within the system. Since this is questionable under consideration of different security aspects to carry out, different approaches follow now.

A. Hypertext Transfer Protocol

Within the *Hypertext Transfer Protocol (HTTP)*, there are two ways of authentication. One is the "Basic Authentication" and the other is the "Digest Access Authentication". In the first variant, authentication takes place using credentials, and the message is encoded using Base64. In this case, anyone can read the message exchange. In the second variant, a challenge is also introduced, in which a nonce and a checksum verified at the end ensure that both parties know the secret [20], [21]. With this type of authentication and communication, sending messages in plain text is particularly critical. As also described in one of the publications of the BSI with regard to requirements to be implemented for critical communication paths in Chapter 2 in the section of technical information security number 33, encryption and authentication must be provided when transmitting data of a critical service [17]. Therefore, it is obligatory to disregard this type of authentication due to

the non-existent encryption and consider its advanced security variant of it.

B. Hypertext Transfer Protocol Secure

The *Hyper Text Transfer Protocol Secure (HTTPS)* consists of the implementation of HTTP using *Secure Sockets Layer (SSL)* or *Transport Layer Security (TLS)*. According to the minimum standard of the BSI for the use of Transport Layer Security according to §8 paragraph 1 sentence 1 BSIG version 2.3 dated 15.03.2022, the recommendations of the technical guideline TR-02102-2 Cryptographic Procedures dated 24.01.2022 [22] must be complied with. Following this technical guideline, TLS1.0, TLS1.1, SSL v2, and SSLv3 are not recommended. Instead, only TLS 1.2 and TLS 1.3 are recommended. Barabanov and Makrushin list six different sources that declare mTLS as the most popular variant of authentication for microservices [10]. As can be seen in Figure 7, the use of HTTPS requires a certificate authority, which is an anchor of trust. Basically, none, one, or both communication partners can authenticate each other [23]. The latter is referred to as *mutual TLS (mTLS)*. The issuing and verification of certificates take place within the certificate authority. But the problem of key management must be overcome. Both recalls and rotations must be made possible. The options to manage this are not mentioned, but for the sake of completeness, however, this aspect has been included. In addition, the use of a PKI results in further obligations with regard to measures to be implemented towards the BSI in the case of using a critical infrastructure. Apart from the problems of key management also mentioned above, PKI generates a certain overhead, which is attempted to be avoided within the reference architecture in this context. However, no additional PKI should possibly be implemented. Therefore, it would also be possible to use *JSON web tokens (JWT)* for authentication.

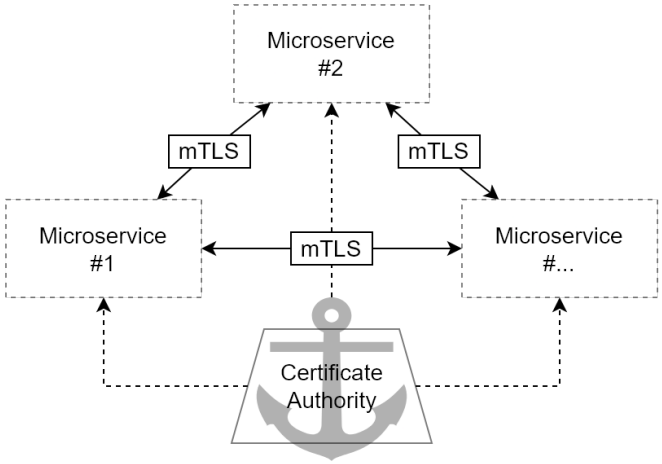

Fig. 7. Authentication with HTTPS.

C. JSON Web Tokens

A wide variety of information can be stored within JWT [24]. Authentication can therefore be performed using context/user information in particular. If the microservices create and sign JWT themselves, a *Public Key Infrastructure (PKI)* is also required. Otherwise, a *Security Token Service (STS)* is introduced, which can issue new tokens and act as an intermediary between two microservices. This process is illustrated in Figure 8. In this case, when two microservices communicate, a new JWT is used each time. Access controls can be performed at the STS. *JSON Web Encryption (JWE)* [25] should be used as the format of the tokens at this point in order to meet the BSI's requirements for encrypting data in critical infrastructure [17]. As with HTTPS, other things have to be taken care of. For example, the detection of recalled or compromised tokens. Basically, JWT can also be used in combination with mTLS, whereby certain security aspects can be handled in each case. Basically, as with HTTPS, the problem of increasing overhead also arises here. There is another option for service-level authentication without the additional need for another infrastructure, such as PKI or STS. In that case, the *Password Authenticated Key Exchange by Juggling (J-PAKE)* protocol is promising.

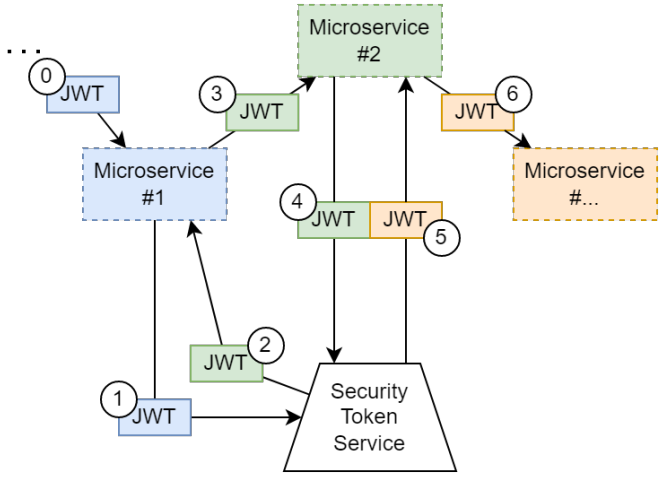

Fig. 8. Authentication with JWT.

D. Password Authenticated Key Exchange by Juggling

The J-PAKE protocol [26] belongs to the family of password authenticated key exchange protocols. This protocol is used to create a cryptographic key based on a shared secret, which is used for further secure communication. No PKI or third party is additionally needed. It also covers other security features. For example, *Perfect Forward Secrecy (PFS)* is also required by the BSI as a property when using TLS [22]. PFS describes the property of the keys so that when a key is known, previous and still later following keys cannot be determined [27]. For the sake of completeness, other security features are also only mentioned but can be traced in some detail within the RFC [26]: Offline and online dictionary attack resistance and known-key security.

However, since this protocol works on the basis of a common shared secret, one difficulty is how and when the services'

secrets are created and propagated accordingly. A more in-depth description of an implementation of the J-Pake protocol in the context of an MSA with an example of using the Jadex Active Components framework was given by Kai Jander et al. [11]. A possible problem of this protocol within an MSA is the level of awareness and the number of implementations, as they do not seem to be very widespread in this case. Especially in the enterprise area of an insurance company. Nevertheless, according to the RFC, J-PAKE has been used in applications such as Firefox sync, Pale moon sync, and Google Nest products [26]. In addition, some protocols of this family also have patents, which makes it difficult to use them, and therefore extra new protocols have been developed [28]. The potential advantage of service-level authentication without additional infrastructure and the fact that it fulfills certain requirements of the BSI in the area of German insurances should be enough to encourage further research in this area.

E. Summary

There are several ways to deal with the authentication of service-level communication. Especially in the context of German insurance or the special requirements of the BSI, it should not be assumed that one's own system will never be compromised. Therefore, there should also be no basic trust in the network, and the communication should be designed accordingly and secure. For service-level authentication, it would make sense to combine both JWT and mTLS so that, for example, authentication can be guaranteed by JWT and further context information can be sent. However, at the same time, the message itself can also be encrypted by mTLS. Within a large insurance company, a multitude of services exist, and the existing system landscape has reached a certain level of complexity. Therefore, the use of mTLS and JWT creates a larger overhead since both a PKI and an STS have to be maintained at the same time. In addition, there are also other problem areas, such as key management. A suitable but, at the same time, theoretical solution to the problem would be the use of the J-PAKE protocol. This protocol enables authentication and the additional creation of a secure communication channel without the need for a PKI or other additional infrastructure. Therefore, the use of J-PAKE seems promising and is recommended by the authors. Nevertheless, more in-depth research, especially on the practical application and also on compliance with regulatory requirements, is needed when using the protocol in an MSA with critical infrastructure within a German insurance company. In addition, there are questions regarding the practical implementation and safeguarding of governance, which should not be neglected. Finally, this approach should continue to be considered until further commercial uses develop.

VIII. Conclusion and Future Work

The security aspect is indispensable in any realization or evolution of application architecture. Especially in Germany, insurance companies have to fulfill legal requirements according to the BSIG if general framework conditions are met, and the resulting status of critical infrastructure is achieved. Every two years, proof must be provided to the BSI that the corresponding security standard is met. The BaFin is responsible for the regulation of this proof. Our partners from the insurance industry, thus, should still be compliant with those requirements if adding a critical (defined based on BSI-KritisV) system part based on RaMicsV.

For better guidance on authorization patterns from a confidentiality perspective, authentication has also been included, as the two security properties are usually close in terms of implementation. Relevant points regarding the implementation at the service-level and edge-level have been included. The paper's main focus was on the different patterns of service-level authorization, which were considered and evaluated in the context of our partners within the insurance industry. Furthermore, approaches to service-level authentication were also described, and their challenges were fundamentally addressed.

Finally, the advantages and disadvantages of the individual patterns were weighed up. The pattern of choice depends on the requirements for scalability and performance. In the context of (grown) insurance and microservices, implementation at the service-level seems the most appropriate. Furthermore, the centralized pattern with the single or the embedded policy decision point comes in closer selection due to the use of the required ESB within RaMicsV. Approaches to service-level authentication were also described, and their challenges were fundamentally addressed. While mTLS seems to be the common standard, PAKE protocols are promising and may take on a larger role in microservices in the future than they have in the past. Thus, an important part of the protection goal confidentiality was addressed. Still, it also took another step closer to answering the initially asked question: "how to help secure (insurance) business applications using potentially several logical parts from RaMicsV, mainly including microservices combined with other typical insurance applications technologies"?

Within this contribution, some guidelines for selecting patterns regarding authorization and authentication at service- and edge-level of critical infrastructure have been started and will be continued within our future work. In addition, our future work also deals with the approach of validity and consistency of embedded policies. To continue to remain oriented towards the protection goals, a prominent topic, service-to-service authentication, will be addressed in more detail in future work as well. In particular, the practical implementation of the theoretical approaches will be considered. It will also look at how the industry standard deals with this. Here, the available options for implementing authentication will be considered inside RaMicsV, and the respective advantages and disadvantages will be weighed against each other. In particular, the highlighted and recommended protocol J-PAKE will also be considered in more depth. Both in terms of legal regulations to be complied with in the context of critical infrastructures of a German insurance company, as well as problems regarding a practical implementation based on RaMicsV.

Furthermore, relevant and current aspects of the broad subject's availability and integrity will then be evaluated one by one to address later emerging security aspects of the MSA, such as deployment options and resulting security domains. The exact order is made in consultation with our partners from the insurance industry, depending on current topics or preferences.

Initial prototypes and proof of concepts have been developed and implemented for the reference architecture and were described in previous publications [8] and [14]. While similar work has not yet been done for the security domain from this publication, the effort required to implement parts or all of the reference architecture in a commercial system depends on the existing SOA, specific functional requirements, and the number of critical systems components to be implemented.

References

[1] A. Koschel, A. Hausotter, R. Buchta, P. Niemann, C. Schulze, C. Rust, and A. Grunewald, "The Need of Security Inside a Microservices Architecture in the Insurance Industry," in *Proceedings of the Fourteenth International Conference on Advanced Service Computing (Service Computation 2022)*, Barcelona, Spain, 2022, [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=service_computation_2022_1_20_10006. [accessed: 2022-12-15].

[2] The European Parliament and the Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation - GDPR))," [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng. [accessed: 2022-12-15].

[3] Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin) - Federal Financial Supervisory (BaFin), "Versicherungsaufsichtliche Anforderungen an die IT (VAIT) (Insurance Supervisory Requirements for IT (VAIT))," [Online]. Available: https://www.bafin.de/SharedDocs/Downloads/EN/Rundschreiben/dl_rs_1810_vait_va_en.pdf?__blob=publicationFile&v=5. [accessed: 2022-12-15].

[4] Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office of Information Security (BSI), "Aufsicht über Kritische Infrastrukturen im Finanz- und Versicherungswesen (Supervision of Critical Infrastructures in the Finance and Insurance Industry)," [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/KRITIS/Nachweispruefungen_im_Finanz-und_Versicherungswesen.pdf?__blob=publicationFile&v=3. [accessed: 2022-12-15].

[5] Gesamtverband der Deutschen Versicherungswirtschaft e.V. (General Association o.t. German Insurance Industry)), "VAA Final Edition. Das Fachliche Komponentenmodell (VAA Final Edition. The Functional Component Model)," 2001.

[6] C. Richardson, *Microservices Patterns: With examples in Java*. Shelter Island, New York: Manning Publications, 2018.

[7] S. Newman, *Building microservices: designing fine-grained systems*. Sebastopol, California: O'Reilly Media, Inc., 2015.

[8] A. Koschel, A. Hausotter, R. Buchta, A. Grunewald, M. Lange, and P. Niemann, "Towards a Microservice Reference Architecture for Insurance Companies," in *Proceedings of the Thirteenth International Conference on Advanced Service Computing (Service Computation 2021)*, Porto, Portugal, 2021, [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=service_computation_2021_1_20_10002. [accessed: 2022-12-15].

[9] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to attribute based access control (abac) definition and considerations (draft)," *NIST special publication*, vol. 800, no. 162, pp. 1–54, 2013.

[10] A. Barabanov and D. Makrushin, "Authentication and authorization in microservice-based systems: survey of architecture patterns," *CoRR*, vol. abs/2009.02114, 2020, [Online]. Available: https://arxiv.org/abs/2009.02114. [accessed: 2022-12-15].

[11] K. Jander, L. Braubach, and A. Pokahr, "Defense-in-depth and role authentication for microservice systems," *Procedia Computer Science*, vol. 130, pp. 456–463, 2018, the 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018), [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918304009. [accessed: 2022-12-15].

[12] Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office of Information Security (BSI), "Act on the Federal Office for Information Security (BSI Act - BSIG) - courtesys translation -," [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/BSI/BSI_Act_BSIG.pdf?__blob=publicationFile&v=4. [accessed: 2022-12-15].

[13] Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office of Information Security (BSI), "Verordnung zur Bestimmung Kritischer Infrastrukturen nach dem BSI-Gesetz (BSI-Kritisverordnung - BSI-KritisV) (Regulation for the Determination of Critical Infrastructures according to the BSI Act (BSI-Kritisverordnung - BSI-KritisV))," [Online]. Available: https://www.gesetze-im-internet.de/bsi-kritisv/BJNR095800016.html. [accessed: 2022-12-15].

[14] A. Koschel, A. Hausotter, M. Lange, and S. Gottwald, "Keep it in Sync! Consistency Approaches for Microservices - An Insurance Case Study," in *SERVICE COMPUTATION 2020, The Twelfth International Conference on Advanced Service Computing*, Nice, France, 2020, [Online]. Available: http://www.thinkmind.org/index.php?view=article&articleid=service_computation_2020_1_20_10016. [accessed: 2022-12-15].

[15] WSO2, "WS02 Enterprise Service Bus Documentation: Securing APIs," [Online]. Available: https://docs.wso2.com/display/ESB481/Securing+APIs. [accessed: 2022-12-15].

[16] The Council of the European Union, "COUNCIL DIRECTIVE 2008/114/EC," [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32008L0114. [accessed: 2022-12-15].

[17] Bundesamt für Sicherheit in der Informationstechnik (BSI) - Federal Office of Information Security (BSI), "Konkretisierung der Anforderungen an die gemäß § 8a Absatz 1 BSIG umzusetzenden Maßnahmen (Specification of the requirements for the measures to be implemented in accordance with Section 8a (1) BSIG)," [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/KRITIS/Konkretisierung_Anforderungen_Massnahmen_KRITIS.pdf. [accessed: 2022-12-15].

[18] CWE Content Team, "Weaknesses in OWASP Top Ten (2021)," 2022, [Online]. Available: https://cwe.mitre.org/data/definitions/1344.html. [accessed: 2022-12-15].

[19] OWASP, "Welcome to the OWASP Top 10 - 2021," 2022, [Online]. Available: https://owasp.org/Top10/. [accessed: 2022-12-15].

[20] J. Reschke, "The 'basic' http authentication scheme," Internet Requests for Comments, RFC Editor, RFC 7617, September 2015.

[21] J. Franks, P. M. Hallam-Baker, J. L. Hostetler, S. D. Lawrence, P. J. Leach, A. Luotonen, and L. C. Stewart, "Http authentication: Basic and digest access authentication," Internet Requests for Comments, RFC Editor, RFC 2617, June 1999.

[22] Bundesamt für Sicherheit in der Informationstechnik (Federal Office for Information Security)), "Technische Richtlinie TR-02102-2 Kryptographische Verfahren: Empfehlungen und Schlüssellängen (Technical Guideline TR-02102-2 Cryptographic methods: Recommendations and key lengths)," [Online]. Available: https://verdata.de/wp-content/uploads/2021/10/BSI-TR-02102-2.pdf. [accessed: 2022-12-15].

[23] E. Rescorla, "Http over tls," Internet Requests for Comments, RFC Editor, RFC 2818, May 2000.

[24] M. Jones, J. Bradley, and N. Sakimura, "Json web token (jwt)," Internet Requests for Comments, RFC Editor, RFC 7519, May 2015.

[25] M. Jones and J. Hildebrand, "Json web encryption (jwe)," Internet Requests for Comments, RFC Editor, RFC 7516, May 2015.

[26] F. Hao, "J-pake: Password-authenticated key exchange by juggling," Internet Requests for Comments, RFC Editor, RFC 8236, September 2017.

[27] C. Eckert, *IT-Sicherheit: Konzepte - Verfahren - Protokolle (IT-Security: Concepts - Procedures - Protocols)*. Berlin, München, Boston: De Gruyter Oldenbourg, 2014.

[28] T. Wu, "What is SRP?" [Online]. Available: http://srp.stanford.edu/whatisit.html. [accessed: 2022-12-15].