Risk-based Authorization in Safety Critical Microservice Architectures

Giorgia Sirigu[1,*,†], Alessandro Burastero[2,†] and Alessandro Armando[1,3,†]

[1]*Cybersecurity National Lab, CINI, Italy*
[2]*ICT, CIMA Research Foundation, Italy*
[3]*DIBRIS, Università degli Studi di Genova, Italy*

Abstract

The increasing adoption of safety-critical platforms based on complex, microservice-based architectures makes them attractive targets for cyberattacks. Security breaches in these systems can have severe safety consequences, demanding strong security mechanisms. A prominent mitigation strategy is enforcing fine-grained access control to confine compromised microservices to their intended activities. In this paper, we propose an OPA-based authorization framework and discuss its effectiveness using Public Warning and Alerting Systems as a reference scenario. We integrate a risk assessment model into the policy decision procedure to enable authorization decisions that are both safety- and (cyber)security-aware. This model accounts for attacks that block legitimate alerts or inject false ones. The experimental results demonstrate that policy evaluation introduces minimal latency and does not affect the timely dissemination of the alerts.

Keywords

Access control, security-risk assessment, safety-risk assessment, microservices

1. Introduction

Safety-critical software systems—such as those supporting transportation, healthcare, industrial control, and civil protection—are increasingly developed and operated using cloud-native technologies. Among these, microservice-based architectures have gained significant traction due to their support for modular development, independent deployment, fault isolation, and rapid evolution. At the same time, the transition from monolithic, perimeter-based systems to highly distributed service meshes substantially increases both the attack surface and the space of possible failure modes. As a result, safety-critical microservices operate at a challenging intersection: they must satisfy stringent dependability, assurance, and certification requirements while remaining resilient in dynamic, large-scale, and frequently changing environments.

Authorization plays a central role in limiting the consequences of cyber incidents in such systems, particularly when security breaches can trigger unauthorized service interactions. Safety-critical platforms are attractive targets for a wide range of adversaries, including cyber-criminal groups, hacktivist organizations, and state-sponsored actors, whose objectives may range from financial extortion to disruption of essential public services and erosion of public trust. In contemporary microservice deployments, authorization is often enforced through static, identity-centric mechanisms—such as role-based access control (RBAC)—configured at design time and enforced at runtime by gateways or service meshes. While effective for enforcing least privilege under stable assumptions, these approaches struggle to cope with the contextual uncertainty typical of safety-critical systems, including partial observability, evolving threat levels, degraded operational modes, and supply-chain compromise. In these conditions, binary allow/deny decisions based solely on identities and static policies may be

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
†These authors contributed equally.
giorgia.sirigu@consorzio-cini.it (G. Sirigu); alessandro.burastero@cimafoundation.org (A. Burastero); alessandro.armando@unige.it (A. Armando)
0000-0002-1112-0871 (G. Sirigu); 0000-0002-5246-2157 (A. Armando)

either overly permissive—enabling lateral movement after compromise—or overly restrictive—hindering timely execution of safety functions during emergencies.

Risk-based authorization (RBA) addresses these limitations by incorporating dynamic risk estimates into access-control decisions. Rather than treating trust as static, RBA evaluates each request against a continuously updated assessment of hazards and threats, leveraging evidence such as workload identity and attributes, runtime posture, behavioral anomalies, safety state, mission phase, and environmental conditions. Depending on the assessed risk, the authorization outcome may remain binary or may involve graduated responses, such as conditional approval, additional verification, or controlled degradation. This approach aligns naturally with zero-trust principles and with established safety engineering practices, which reason explicitly in terms of the likelihood and severity of adverse events. However, applying RBA to microservice-based safety-critical systems introduces unique challenges. Risk models must be sufficiently expressive to capture the interplay between safety hazards and cyber threats, yet remain explainable and auditable to support governance and certification. Moreover, authorization mechanisms must operate within strict real-time constraints, ensuring that security controls do not undermine the very safety objectives they are meant to protect.

Despite growing interest in adaptive access control and zero-trust architectures, existing work provides limited guidance on how to realize risk-based authorization that jointly accounts for safety and security concerns in microservice environments. Most proposals focus on enterprise workloads, emphasize user-to-service interactions rather than east–west service communication, or treat safety and security risks as largely independent dimensions.

In this paper, we propose a risk-based authorization model in which access-control decisions explicitly combine safety and security considerations. We illustrate the approach through IT-Alert, the Italian national public warning system operated by the Department of Civil Protection (DPC), which disseminates location-targeted emergency notifications to populations potentially affected by hazardous events. In such systems, the safety benefits of timely warnings must be carefully balanced against the risks associated with false alerts, which can induce panic, damage institutional credibility, and ultimately reduce compliance with future warnings. This trade-off is further complicated by the presence of cyber threats that may either suppress legitimate alerts or inject false ones.

Beyond the attack techniques themselves, it is essential to consider the threat agents and their motivations. State-sponsored adversaries may target a national alerting infrastructure as part of broader hybrid operations aimed at destabilizing civil society or degrading crisis-response capabilities. Cybercriminal groups may seek financial gain through extortion schemes based on alert suppression or manipulation, while hacktivist or politically motivated actors may attempt to generate public disorder or reputational damage by disseminating false alerts. Additionally, insider threats—such as operators, contractors, or personnel within data-providing organizations—may abuse legitimate access or exploit operational knowledge to interfere with the alerting pipeline. These diverse threat profiles motivate the need for authorization mechanisms capable of constraining microservice behavior under partial compromise and of integrating risk signals into decision-making to preserve both public safety and trust.

Our model evaluates authorization decisions by jointly considering the expected public-safety impact of issuing or suppressing an alert and the likelihood that the alert request is malicious. We demonstrate the application of this approach across representative IT-Alert scenarios, including high-severity regional events and situations characterized by elevated cyber threat levels. We further present a prototype implementation based on the Open Policy Agent (OPA), which integrates risk assessment into policy evaluation at runtime. Experimental results show that the proposed approach is compatible with IT-Alert's operational constraints, introducing minimal latency while enabling explainable and auditable authorization decisions suitable for safety-critical systems.

The remainder of the paper is organized as follows. Section 2 introduces microservice-based architectures and describes the IT-Alert platform. Section 3 presents the threat model and the IT-Alert requirements. Section 4 discusses access control prerequisites, policy enforcement design choices, and examples of policy definition. Section 5 illustrates the risk assessment parameters and functions, and shows how they are integrated into OPA. Section 6 reports on the experimental settings and results.

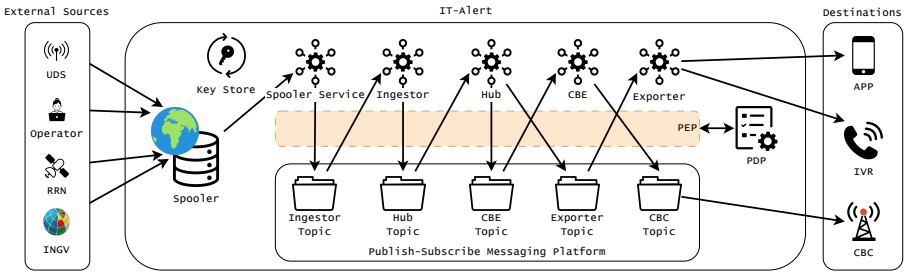

Figure 1: High-level architecture of IT-Alert and interactions among its microservices

Section 7 reviews related work, and Section 8 concludes the paper.

2. Background

This section introduces the architectural paradigm underlying modern safety-critical platforms and then describes IT-Alert, the Italian Public Warning and Alerting System (PWAS), which serves as the reference system throughout the paper.

2.1. Microservice-based Architecture

Microservice-based architecture is a widely adopted software design paradigm in which an application is decomposed into a set of small, loosely coupled services that communicate through well-defined interfaces, rather than being implemented as a single monolithic system [1, 2]. Each microservice runs as an independent process, can be developed and deployed autonomously, and typically encapsulates a specific business capability. This architectural style provides several benefits, including improved scalability, fault isolation, flexibility in technology choices, and accelerated software evolution.

Despite these advantages, microservice-based systems pose significant security challenges. The decomposition of functionality into numerous interacting services substantially increases the system's attack surface [3, 4]. Security must therefore be addressed at the system level rather than through isolated, service-specific defenses [2]. Inter-service communication relies on network-based APIs, often requiring multiple open ports and publicly reachable endpoints. Moreover, microservice deployments frequently assume implicit trust among services once initial authentication has been performed, enabling unverified east–west traffic within the system. This trust model can facilitate lateral movement when a single service is compromised.

Additional risks stem from excessive privileges assigned to individual microservices and from vulnerabilities or misconfigurations in containerization and orchestration platforms. As a result, an attacker gaining control of a single microservice may be able to affect the behavior of the entire application. These issues are particularly critical in safety-sensitive domains, where cyber incidents may directly translate into physical harm or large-scale societal disruption.

2.2. IT-Alert

IT-Alert is the Italian PWAS operated by the Department of Civil Protection (DPC). The platform is designed to disseminate timely, location-targeted alerts to the population in the presence of hazardous events. IT-Alert can be activated for several emergency scenarios, including nuclear or radiological incidents, major industrial accidents, large dam failures, and volcanic activity. Additional scenarios, such as earthquake-generated tsunamis and severe meteorological events, are currently under testing.

The platform continuously ingests environmental and meteorological data from heterogeneous sources. When predefined conditions are satisfied—for instance, when rainfall intensity or storm severity indices exceed specific thresholds—the system generates and disseminates alert messages to individuals located in the affected area.

Figure 1 presents a high-level view of the IT-Alert microservice architecture and the corresponding message flows. Input data originates from multiple external sources, including human operators from the DPC, the National Radar Network (RRN), which provides nationwide meteorological observations, the National Institute of Geophysics and Volcanology (INGV), which supplies seismic and volcanic data, and Unverified Data Sources (UDS). While UDS can enrich situational awareness—for example, through satellite imagery or environmental sensors—their trustworthiness is inherently lower. Incoming data are initially collected and stored in the *Spooler*. The *Spooler Service* retrieves this data, performs preliminary validation, and publishes it to the *Ingestor Topic*. The *Ingestor* then acquires the data, digitally signs the message through interaction with the *Key Store*, and forwards the digitally signed message to the *Hub Topic*. The *Hub* analyzes message attributes and routes the data either to the *Exporter Topic* or to the *Cell Broadcast Entity (CBE) Topic*, based on predefined conditions. Messages published to the *Exporter Topic* are processed by the *Exporter* service and delivered to external applications, such as web platforms and Interactive Voice Response (IVR) systems. Messages routed to the *CBE Topic* are retrieved and digitally signed by the *Cell Broadcast Entity*, again interacting with the *Key Store*, and subsequently forwarded to the *Cell Broadcast Centre (CBC)*. The CBC is responsible for initiating the final broadcast of the alert to mobile devices in the designated geographical area.

All read and write operations within IT-Alert are mediated by an access control mechanism composed of a *Policy Decision Point* (PDP) and a *Policy Enforcement Point* (PEP). The PDP evaluates whether a microservice is authorized to perform a requested operation, while the PEP enforces the decision by allowing or denying the action. This separation is essential to limit the impact of compromised components and to ensure consistent enforcement across the system.

Throughout the entire workflow, from data ingestion to alert dissemination, information is encapsulated as digitally signed CAP-IT messages [5]. CAP-IT is an extension of the CAP 2.0 standard [6], providing a machine-readable, XML-based format for representing alerts and warnings, including hazard type, severity, certainty, and the geographical area affected.

3. Threat Model

The key security properties for PWAS are integrity and availability. Any violation of these properties may result in severe physical, social, and reputational consequences. (i) an attacker prevents the generation or dissemination of a legitimate alert when hazardous conditions warrant it; and (ii) an attacker causes the system to generate and disseminate a false alert. Both scenarios can be realized either by compromising one or more microservices within the platform or by subverting external data sources that provide inputs to the alerting pipeline.

A successful false-alert injection may induce public panic, disrupt emergency services, and trigger secondary safety incidents. In addition, repeated false alarms can significantly erode public trust in the warning system, reducing compliance with future alerts and ultimately undermining the effectiveness of the platform. Conversely, suppressing or delaying legitimate alerts prevents timely protective actions, exposing the population to the full impact of hazardous events.

To mitigate threats arising from partial compromise of internal components, IT-Alert adopts a zero-trust security posture, whereby no implicit trust among microservices is assumed. Beyond traditional cryptographic protections for data at rest and in transit, access control plays a central role in limiting the damage that a compromised component can inflict. Given the operational constraints of safety-critical infrastructures, access control mechanisms must be non-intrusive, easy to integrate, and must not significantly increase architectural complexity, as additional components may themselves enlarge the attack surface. A further requirement is a strict separation between policy decision-making and policy enforcement. Locating authorization logic outside individual microservices reduces the risk that a compromised service can manipulate authorization outcomes [7].

Among the potential attack vectors, external data sources represent a particularly exposed surface. While core components such as the Cell Broadcast Entity (CBE) and the Cell Broadcast Centre (CBC) are developed and operated under strong security controls, the same level of protection is difficult to

attain for geographically distributed sensing infrastructures. These include meteorological stations, environmental sensors, and third-party data providers, which may operate under heterogeneous security assumptions.

Accordingly, we consider threats targeting the integrity of data ingested by the PWAS. Under this assumption, we consider two classes of attacks:

False Negative Attack (A_{FN}) The attacker manipulates incoming data so as to conceal the presence of a high-risk hazard. As a consequence, the PWAS fails to generate an alert, leaving the affected population unaware of the impending danger and exposed to its full impact.

False Positive Attack (A_{FP}) The attacker alters incoming data to simulate a non-existent high-risk hazard. This causes the PWAS to disseminate a false alert, potentially inducing panic, unnecessary evacuations, and reputational damage that reduces the system's future credibility and effectiveness.

These attack classes capture the fundamental safety–security trade-off faced by PWAS. On the one hand, conservative alerting policies may increase false positives and erode trust; on the other hand, restrictive policies may increase false negatives and expose the population to harm. In the remainder of the paper, we leverage this threat model to design authorization mechanisms that explicitly account for both safety impact and cybersecurity risk, enabling informed and context-aware decisions even in the presence of partial compromise or uncertain data.

4. Declarative Specification of Authorization Policies

In a safety-critical platform built on a microservice-based architecture, each service participates in tasks such as data ingestion, alert generation, message validation, and notification dissemination. Ensuring that microservices perform only authorized operations requires an expressive and maintainable policy framework. Defining access control policies in a microservice-based system is challenging, as excessive policy complexity increases the likelihood of misconfigurations and may result in unintended privilege escalation, undermining the system's security. Indeed, a policy must determine whether a microservice is authorized to send a message by verifying environmental and meteorological indicators, validating the legitimacy of the sender, and ensuring the consistency of message attributes within the reported event (e.g., location). Therefore, a key requirement for safety-critical platforms is that access control policies must be easy to express, maintain, and update [8]. The policy language plays a fundamental role, as it should minimize the cognitive effort for developers and reduce the risk of semantic errors.

To overcome these shortcomings and to meet the design requirements established above, we adopt Open Policy Agent (OPA)[1], an open-source, general-purpose policy engine that facilitates policy decision-making within applications. OPA provides well-defined APIs for submitting policy evaluation queries, eliminating the need for developers to implement a custom policy decision logic. Moreover, OPA leverages Rego, a datalog-based, declarative language that simplifies the policy specification activity, thereby reducing the time spent on writing and reviewing policies. As we will see, policies written in Rego can express fine-grained constraints on message publication, sender legitimacy, and contextual validation, including environmental parameters contained in the CAP-IT messages.

Policy evaluation boils down to submitting a structured query to OPA containing relevant contextual information (e.g., meteorological indices) through its APIs. OPA evaluates the input against the predefined policies and optional auxiliary data, returning a decision that the application can enforce. The interoperability of such a policy engine with Kafka, Kubernetes, and Docker makes it suitable for deployment in a microservice-based architecture. By simplifying policy authoring and automating policy decision-making, OPA reduces configuration errors and strengthens the security guarantees of the alert dissemination pipeline.

[1] https://www.openpolicyagent.org/docs

```
{
    "allowed_type_by_sender":{
        "rrn": ["heavy-rain"]
        },
    "allowed_type_by_area":{
        "area1": ["heavy-rain", "radio-chemical"],
        "area2": ["heavy-rain", "volcanic"],
        "area3": ["radio-chemical"]
    }
}
```

(a)

```
{
    "destinations":[
        {"id": "cbe",
        "severity": ["extreme", "severe"],
        "certainty": ["observed", "likely"]},
        {"id": "exporter",
        "severity": ["extreme", "severe", "moderate", "minor"],
        "certainty": ["observed", "likely", "possible", "unlikely"]}
    ]
}
```

(b)

Figure 2: Example of configuration parameters used for evaluating policy decision queries

```
1  default allow = false
2  event := input.parameters.event
3  sender := input.parameters.sender
4  area := input.parameters.area
5  allowed_type_by_sender := data.allowed_type_by_sender[sender]
6  allowed_type_by_area := data.allowed_type_by_area[area]
7  allow if {
8      event in allowed_type_by_sender
9      event in allowed_type_by_area
10 }
```

Figure 3: Rego policy restricting the publishing activity of the Ingestor on the Hub Topic

Therefore, we consider OPA and its policy language, Rego, well-suited for expressing authorization policies. As an illustrative example, we consider the microservices in IT-Alert and define a policy that prevents the Ingestor component from publishing a message on the Hub Topic unless specific policy-defined conditions are satisfied:

- The sender must be authorized to generate messages for the given event type.
- The event must be permitted within the geographic area declared in the message (e.g., the volcanic eruption is a valid event for a region where a volcano exists).

To this end, we need to specify the data OPA uses when evaluating authorization queries. This data represents the system's configuration parameters.

For instance, Figure 2a lists the parameters that OPA verifies when evaluating a policy decision query. In this example, they are related to the Ingestor component, which is permitted to publish at the Hub Topic only if the specified conditions are met. The parameter *allowed_type_by_sender* defines the list of event types about which a sender (e.g., RRN) can generate a message. The parameter *allowed_type_by_area* specifies the acceptable alert types for each geographic area. For instance, *area1* accepts *heavy-rainfall* and *radio-chemical* events, whereas *area2* allows *tsunami* and *volcanic* events.

Figure 3 shows a possible implementation of this policy in Rego. Line 1 sets the default value of the variable *allow* to *false*, meaning that requests are denied unless explicitly permitted. Lines 2-4 extract the message event, sender, and area values from the input, whilst Lines 5 and 6 load the system parameters. The following lines include the rule that OPA evaluates. Line 7 specifies that *allow* will be set to *true* if the conditions inside the rule are satisfied. Lines 8 and 9 state that the event must be allowed for both the sender and the declared area. The variable *allow* contains the final result, which is returned to the microservice as the authorization decision.

Figure 4 represents two examples of input queries that the Ingestor may generate. By comparing these queries with the system parameters in Figure 2a, we can determine the resulting decision from OPA. The query on the left, Figure 4a, yields a positive decision because the event (*heavy-rainfall*) belongs to the list of events for which the sender (*rrn*) can generate the message, and the event happens in an area (*area1*) where it can happen. Consequently, OPA returns *true*.

Conversely, the input query in Figure 4b results in a negative decision, as the type of event (*volcanic*) is not included in the allowed events of the sender. Hence, once the Hub component queries OPA for the *allow* value, the returned result is *false*.

```json
{
    "parameters": {
        "event": "heavy-rain",
        "sender": "rrn",
        "area": "area1",
        "severity": "extreme",
        "certainty": "observed"
    }
}
```

(a)

```json
{
    "parameters": {
        "event": "volcanic",
        "sender": "rrn",
        "area": "area1",
        "severity": "extreme",
        "certainty": "observed"
    }
}
```

(b)

Figure 4: Examples of input queries

```
1  severity := input.parameters.severity
2  certainty := input.parameters.certainty
3  send_to contains dest if {
4      destination := data.destinations[_]
5      severity == destination.severity[_]
6      certainty == destination.certainty[_]
7      dest := destination.id
8  }
```

Figure 5: Rego policy determining the recipients to which the Hub must forward the message

To illustrate the expressive power of Rego, we provide an additional example of how it can be integrated into IT-Alert to implement more advanced decision logic. After a message has been published on the Hub Topic, the Hub component reads it and decides whether the message should be forwarded to the Exporter Topic, the CBE Topic, or both. Instead of developing this logic within the Hub, the decision can be delegated to OPA, which returns the list of recipients that should receive the message. Figure 2b shows the configuration parameters that OPA must evaluate during the decision procedure. Writing on the CBE Topic is allowed when the severity is *extreme* or *severe*, and the certainty is either *observed* or *likely*. Instead, writing on the Exporter Topic is permitted in the same cases and, additionally, when the severity is *moderate* or *minor*, and the certainty is *possible* or *unlikely*.

Figure 5 reports the Rego policy implementing such a behavior. Lines 1 and 2 extract the input parameters associated with the severity and the certainty. Line 3 defines the rule that populates the list *send_to*, which will be returned as output of the OPA query evaluation. Line 4 takes one destination at a time from the configuration settings. Lines 5 and 6 verify whether the message severity and certainty match the values allowed for the destination. Finally, line 7 returns the identifier of the destination that it is attached to *send_to* when the conditions are satisfied. For example, evaluating the query in Figure 4a returns $[cbe, exporter]$, indicating that the Hub must forward the message to both Exporter and Hub Topics.

5. Risk-based Authorization Policies

In this section, we present our risk-based authorization model for PWAS. The model leverages the assessment of the risk associated with the possible occurrence of a hazard. The assessment is based on environmental data (e.g., rainfall) collected from external sources. Depending on the risk, the PWAS decides whether to send the alert message to the population exposed to the hazard. The decision to issue an alert message takes into account a number of factors:

- the set of possible hazards $\mathcal{H}$;
- the probability of the occurrence of an hazard, $P(H) \in [0, 1]$ with $H \in \mathcal{H}$;
- the impact of the hazard, $I(H) \in \mathbb{N}$: for the sake of simplicity, we measure the impact as the number of casualties caused by the hazard;

- the *efficacy* of the system, $\mathcal{E} \in [0, 1]$, measured as the fraction of the population that reacts appropriately upon receipt of an alert;
- the set of possible cyberattacks $\mathcal{A}$: the cyberattacks against the PWAS, the sources of data used to generate the alert, and the telco operators in charge of distributing it;
- the probability of the occurrence of a cyberattack, $P(A) \in [0, 1]$ with $A \in \mathcal{A}$;
- the impact of a false alarm, $I_F \in \mathbb{N}$: the number of casualties due to the dysfunction caused by the false alarm; this includes both the harm caused by the panic induced by the alarm as well as the damage to the reputation of the system, which ultimately reduces its efficacy *in the future*.

As a baseline, we define the (unmitigated) risk associated with a hazard H to be

$$R_U(H) = P(H) * I(H) \tag{1}$$

where $P(H)$ and $I(H)$ are the probability and the impact of the occurrence of H, respectively.

By sending an alert, it is possible to mitigate the risk to

$$R_M(H) = P(H) * I(H) - RR(H) \tag{2}$$

where

$$RR(H) = P(H) * I(H) * \mathcal{E} \tag{3}$$

is the risk reduction offered by the PWAS. For example, if $\mathcal{E} = 0.8$ then 80% of the population is expected to take precautions to get to safety, thereby leading to a 80% reduction of the risk.

Thus, the PWAS must weigh the reduction of the risk obtained by sending an alarm with the increase of the risk due to the possibility of the message being a false alarm. This boils down to computing $RR(H)$ and checking whether $RR(H) > 0$. If $RR(H) > 0$, then the alert is sent. Otherwise, the risk of sending a false alarm outweighs the risk mitigation offered by the alerting system, and the message is not sent.

It must be noted that the computation of $RR(H)$ is done by leveraging estimates of $P(H)$, $I(H)$, $\mathcal{E}$, and I_F. While estimates of $I(H)$, $\mathcal{E}$, I_F can be derived on historical data, the estimate of $P(H)$ critically relies on data originating from the available data sources. Thus, whenever the likelihood of a cyberattack is not negligible, we must take into account the possibility that the available estimate of $P(H)$ is not reliable. Besides, we assume that indicators of compromise (IoC) provided by cybersecurity monitoring tools, e.g., Intrusion Detection Systems (IDS) or Security Information and Event Management Systems (SIEM), are available and that the PWAS can leverage this and other data (e.g., public sources) to estimate the likelihood $P(A)$ of an attack.

Let us consider the case of a cyberattack of type A_{FN}. In this situation, the PDP must weigh the pros and cons of sending the alert. This is done by assessing the cyber risk as follows:

$$C(A_{FN}) = P(A_{FN}) * R_U(H) + (1 - P(A_{FN})) * R_M(H) + (1 - P(H)) * I_F \tag{4}$$

This expression considers the damage caused by the occurrence of A_{FN}, the damage in the case of the correct functioning of the PWAS, as well as when an alert for a hazard H is issued, but the calamitous event does not occur. In the latter, the alert is inevitably considered a false alarm by the targeted population, which affects the reputation of the PWAS and, consequently, its future efficacy.

The alert message is sent if the following condition holds:

$$C(A_{FN}) < R_U(H) \tag{5}$$

Namely, the alert is sent if the cyber risk is smaller than the damage caused when the PWAS is not operative.

For instance, assuming that H is likely to occur, $P(H) = 0.9$, with $I(H) = 9$, but the attack does not, $P(A_{FN}) = 0.1$. Also, the efficacy of the alert on the population is high, $\mathcal{E} = 0.8$, and the impact in the case of a false alert is low, $I_F = 1$. In this case, the PWAS will send the alert message because $R_U(H) \approx 8.1$ and $C(A_{FN}) = 2.37$, which satisfy the condition $C(A_{FN}) < R_U(H)$.

```
1   r_u := x if {
2       x := p_h * i_h
3   }
4   expected_risk := y if {
5       rr := p_h * i_h * efficacy
6       r_m := p_h * i_h - rr
7       y := p_a * r_u + (1 - p_a) * r_m + (1 - p_h) * i_f
8   }
9   risk_assessment := "send" if {
10      expected_risk < r_u
11  }
```

Figure 6: Security and safety risk assessment logic for an attack $A \in \{A_{FN}, A_{FP}\}$ expressed in Rego

Consider now the case where the occurrence of H is improbable, $P(H) = 0.1$, but the A_{FN} is probable, $P(A_{FN}) = 0.9$. Again, keeping $I(H) = 9$, $\mathcal{E} = 0.8$, and $I_F = 1$. In this case, the PWAS will not send the alert message because the Condition (5) is not satisfied, as $R_U(H) \approx 1$ and $C(A_{FN}) = 1.73$.

The case of a cyberattack of type A_{FP} can be dealt with analogously. In this case, the cyber risk evaluates this way:

$$C(A_{FP}) = P(A_{FP}) * R_U(H) + (1 - P(A_{FP})) * R_M(H) + (1 - P(H)) * I_F \tag{6}$$

which includes the cases where the hazard event happens, and the PWAS sends a wrong alert due to A_{FP}, only H happens, and the PWAS works properly, and when an alert is sent despite no H occurring.

The alert message is then sent if the following condition holds:

$$C(A_{FP}) < R_U(H) \tag{7}$$

Assuming a high likelihood of H to occur with a low impact on the population, and an improbable attack. We might have following likelihoods and impacts: $P(H) = 0.9$, $I(H) = 1$, $P(A_{FP}) = 0.1$, $\mathcal{E} = 0.8$, and $I_F = 1$. After assessing $C(A_{FP}) = 0.35$ and $R_U(H) \approx 1$, the PWAS will send the alert because the Condition (7) is satisfied. Let us use the same impacts and efficacy of the previous example, but the opposite situation in which $P(H) = 0.1$ and $P(A_{FP}) = 0.9$. Here, $C(A_{FP}) > R_U(H)$ because $R_U(H) \approx 0$ and $C(A_{FP}) = 0.99$ and thus the alert is not sent.

An estimate of $P(A)$ for $A \in \{A_{FN}, A_{FP}\}$, can be obtained by combining the following indicators:

- $\iota \in 0..1$, the IoC from a cybersecurity monitoring system,
- $\tau \in 0..1$, the *trustworthiness* of the data source, and
- $\pi \in 0..1$, the *plausibility* of the data.

as follows:

$$\widehat{P}(A) = 1 - (1 - \iota)\tau\pi \tag{8}$$

For example, for IT-Alert, possible values of τ could be 1 when the issuer is DPC, 0.6 when the issuer is RRN, and 0.2 when the issuer is an Unverified Data Source, while possible values of π could be 0 if the alarm is about a tsunami affecting a region that is not a coastal area, 0.3 if the alarm is about a earthquake affecting an area which belongs to a low seismic risk category.

To include the safety and security risk assessment, we leverage the conditional assignment provided by Rego. Figure 6 shows the implementation of the risk assessment logic for an attack $A \in \{A_{FN}, A_{FP}\}$. Here, p_a can be either $P(A_{FN})$ or $P(A_{FP})$, since the computation of the cyber risk follows the same formulation for both attacks and differs only in the corresponding attack probability. Lines 1 to 3 evaluate the risk associated with the non-use of the PWAS—i.e., $R_U(H)$ in Conditions 5 and 7. Instead, lines 4-8 compute the cyber risk of sending an alert message—i.e., $C(A_{FN})$ and $C(A_{FP})$ of Conditions 5 and 7 respectively. The resulting values are eventually compared at lines 9-11 to determine whether the alert should be sent.

```
decision := {
    "allow": allow,
    "R_U": r_u,
    "expected_risk": expected_risk,
    "risk_assessment": risk_assessment
}
```

```
{
  "allow": false,
  "R_U": 1,
  "expected_risk": 0.35,
  "risk_assessment_fp": "send"
}
```

Figure 7: The query output expressed in Rego (left) and the resulting decision (right)

Rego allows specifying the decision returned by OPA directly within the policy. An example of this feature is illustrated in Figure 7, where the left image shows the Rego specification and the right image reports the decision returned by the policy query defined in Figures 3 and 6.

The expressions used for assessing the cyber risk of both A_{FN} and A_{FP} are identical, as well as the conditions applied in the automatic decision to send the alert message. Instead, the difference lies in the interpretation of the resulting values. Let us consider the case where both the hazard event probability and the likelihood of the attack are high: $P(H) = 0.9$ and $P(A) = 0.9$. Assume $I(H) = 1$, $\mathcal{E} = 0.8$, and $I_F = 1$. The calculated cyber risk is $C(A) = 7.55$, while the unmitigated risk is $R_U(H) \approx 8$. In this case, both Conditions (5) and (7) are satisfied, and thus the PWAS would send the alert. Focusing on A_{FN}, transmitting the alert message may be the best choice, as the attack aims to suppress it. In contrast, for A_{FP}, sending the alert might propagate false data, potentially increasing the actual hazard impact. Therefore, prior validation of the message content is essential. To address these varying scenarios, the PWAS should notify system operators—or administrators—if a potential attack is detected, so that they can intervene promptly. This notification might trigger when $\widehat{P}(A) > Th$, with Th as a stated threshold indicating the system estimates a possible ongoing cyberattack. Notably, $\widehat{P}(A)$ represents only an estimation of the probability of an attack, and not a definitive indicator. Consequently, an analysis of the PWAS is necessary to confirm the presence of the attack.

6. Implementation and Experiments

We have developed a simplified prototype of the IT-Alert platform to emulate microservices and their interactions. Our primary goal is to estimate the time required to assess authorization through OPA and to publish or read messages on a topic. A timely alert is necessary in case of an emergency, as the population must be immediately informed about the risk affecting them, so they can take the necessary countermeasures to safeguard their lives. Therefore, computation time measurements are meaningful for IT-Alert because the associated delays directly impact the overall time it takes for an alert message to reach the population at risk.

For testing purposes, we employ Kafka as the publish-subscribe platform and OPA as the policy decision engine. We emulate the Ingestor as a representative microservice for the experiments. This choice is not restrictive, as the other entities, such as the Hub, the Exporter, and the CBE, perform the same sequence of actions when interacting with OPA and would therefore behave identically. Specifically, we develop the logic of the Ingestor publishing to the Hub Topic. This microservice first queries OPA to assess authorization and then publishes the message to Kafka.

We deploy Kafka[2] and OPA[3] as Docker containers. The Hub and monitor microservices are implemented in Python v3.13.3[4]. The experiments are executed on a VMWare virtual machine[5] running Fedora Linux 42[6], equipped with 16.8 GB of RAM and an Intel® Core™ Ultra 7 155H (12 cores). The code is available on GitHub[7].

[2]https://hub.docker.com/r/wurstmeister/kafka
[3]https://hub.docker.com/r/openpolicyagent/opa
[4]https://www.python.org/
[5]https://www.vmware.com/
[6]https://fedoraproject.org/it/workstation/
[7]https://github.com/GiorgiaS/it-alert_access_control.git

For the enforcement, we measure the time required for the Ingestor microservice to query OPA, with the input shown in Figure 2a, wait for the decision about the policy in Figure 3, and publish the message to a Kafka topic. We perform the same measurements for the risk assessment, with the only difference being the policy evaluated by OPA, which is more complex and requires additional computations.

To ensure robustness of the results, we repeat the experiments multiple times and calculate the average time.

The results demonstrate the affordability of integrating OPA into the IT-Alert platform. In both the basic assessment and risk assessment cases, microservices require a few milliseconds to complete their operations. For the case without the risk assessment, the Ingestor requires, on average, $0.135\,\text{seconds}$ to query OPA and publish a message to a Kafka topic. This average time slightly increases to $0.137\,\text{seconds}$ when risk assessment is included, as OPA performs additional policy computation.

Overall, OPA policy decisions introduce only a negligible latency, resulting in a minimal contribution to the overall system delay. As observed in the experiments, the additional computation logic introduced by the risk assessment does not affect the alert dissemination time, satisfying the strict timeliness requirements of PWAS platforms such as IT-Alert.

7. Related Work

This section reviews prior work on access control measures for microservice architectures and existing approaches to cybersecurity risk assessment. These topics are inherent to our contribution: a non-intrusive access control mechanism and a risk assessment model tailored for safety-critical platforms.

7.1. Access Control in Microservice Architectures

Integrating security into microservice-based platforms has increasingly attracted research interest in recent years [4]. Several access control solutions rely on existing technologies. In [9], blockchain enhances decentralized access control through a permission smart contract to store and manage microservice permissions, and an application smart contract to manage the access control policies and authorize microservice requests. Another solution, proposed in [10], presents a two-step validation token-based access control for both centralized and decentralized microservice architectures. The authors of [11] propose a fine-grained access control framework for microservices that combines OAuth2 for user consent and token-based authorization, and XACML to define and evaluate policies. Also, [12] proposes a dynamic, fine-grained, multi-party policy-based access control framework for federations of cloud and edge microservices. Policies can be expressed in various languages, while enforcement leverages UMA 2.0 and OAuth 2.0 token-based mechanisms.

Besides token- or blockchain-based approaches, other works integrate OPA and Rego. Among these, [13] adopts the policy-as-code paradigm to enforce security constraints in an edge-cloud environment, providing a three-layer architecture, of which the control plane layer is for policy and deployment management. A second solution is [14], which integrates policy modeling within DevSecOps workflows and uses OPA to evaluate the configuration artefacts against the defined policies. Additionally, [15] proposes the Open policy agent Wrapper State Manager to enable stateful access control in OPA, without altering existing policies or the evaluation engine.

Beyond policy enforcement, other works address automatic policy generation. The system in [16] analyzes Remote Procedure Calls logs generated by microservices to create and update access control rules based on an interaction topology graph. Another proposal is AUTOARMOR [17], which analyzes the source code of microservices to extract network APIs, builds a control flow graph and a permission graph, and derives policies that are updated as services evolve.

However, these proposals do not satisfy the requirements for critical-safety systems. Most of them require significant architectural modifications—such as additional layers, blockchain, policy evaluation logic, and token-based exchanges—which introduce additional complexity and operational overheads into the system. Safety-critical platforms instead require security mechanisms that are minimally

intrusive, lightweight, and easy to maintain. Moreover, due to their critical nature, they cannot rely solely on automatically-generated policies; thus, human checks are required.

7.2. Risk Assessment

Much research has been done on cybersecurity risk assessment, covering a wide range of application domains. A solution for an access control system is [18], which applies the NIST SP 800-30 framework to quantify security risk by combining object sensitivity, subject trustworthiness, and action impact. [19] proposes a risk assessment framework for cyber systems grounded in the Multicriteria Decision Analysis (MCDA). In this approach, scores are assigned to three top-level criteria (Threat, Vulnerability, and Consequence) and their subcriteria, and then aggregated through a weighting system to derive the overall risk level. Similarly, [20] evaluates cybersecurity risks in autonomous ships using the Bayesian Best-Worst Method. The process consists of defining the risk criteria and sub-criteria, weighing each of them, and finally aggregating these weights to estimate the comprehensive risk. Authors in [21] combine CVE with CVSS to identify vulnerabilities and assign corresponding risk scores. They propose two risk assessment models: a base and a graph-based one. [22] presents a security risk assessment framework for medical devices where, for each stakeholder, identifies risks and assigns the best risk control. [23] introduces a data-driven risk assessment approach that leverages data flow diagrams to model personal data and the system components involved in its storage, processing, and transmission. [24] uses a neural network-driven method for cybersecurity risk assessment in dynamic scenarios such as smart cities. Finally, [25] focuses on civil aviation and assesses risk as the product of the likelihood of an event occurring and its potential impact, and identifies the security measures to be employed. To the best of our knowledge, only one research solution for microservice architecture is available in the literature: [26], which introduces CyberWisePredictor, a framework for security risk assessment in microservice architectures. It includes four phases: 1) vulnerability scanning; 2) mapping to extract CVSS metrics; 3) analysis using NLP to predict missing CVSS data; and 4) risk assessment combining known and predicted metrics.

Unlike these solutions, which only assess cybersecurity risks, our objective is to evaluate safety risks while treating the likelihood of a cyberattack as a relevant parameter. Our goal is to measure how much the warning system reduces the impact of a hazardous event, assuming correct system operation, while still accounting for security risk that may undermine the credibility of the entire platform.

8. Conclusion and Future Work

In this paper, we presented a solution to enhance microservice security in microservice-based PWAS. In particular, we focused on public warning systems and used IT-Alert as a case study. These platforms have significant safety implications, as they warn people about hazardous events that may threaten their lives and provide guidance on protective actions. Our approach enforces an access control mechanism that limits each microservice's permissions to those required to perform its tasks. To this end, we employ OPA and its declarative policy language, Rego. OPA enables the separation of the policy decision-making process from the microservices themselves, while Rego simplifies policy definition and reduces the likelihood of misconfigurations.

We provided concrete Rego policies demonstrating how they can meet the security requirements of PWAS. Furthermore, we incorporated a risk assessment model that quantifies the impact of unintended behavior of the whole system—namely, false positive and false negative attacks—in terms of casualties. Finally, we evaluated these scenarios by simulating part of the IT-Alert message flow to measure the latency introduced by policy decision, showing that the overhead is negligible and does not affect alert dissemination. Overall, our findings demonstrate that access control can strengthen the platform's security without compromising its operational performance.

As future work, we propose integrating PWAS with an automated policy creation and update framework. Such a mechanism can support human operators in generating correctly configured policies, aiming at minimizing the risk of errors. Automatic policy generation could be achieved by

analyzing network APIs [17] and RPCs logs [16]. Furthermore, the risk assessment depends on impact parameters that are difficult to obtain. While the impact of an event can be quantified by analyzing past incidents and extracting causality data, the effect of a false alert is more challenging to quantify, as no historical data is available. A possible direction is to conduct surveys asking participants whether they would rely on the alerting systems after receiving a false alert. This trust metric could be integrated into the risk model to derive a more accurate estimate of user response and system effectiveness.

Other improvements may concern the proposed risk assessment framework. Among these, further types of attacks can be incorporated into the evaluation. For instance, a spamming attack may be considered, in which an attacker compromises the PWAS to flood people with alert messages. Additionally, the current risk assessment method returns a distinct decision for every type of attack. However, having a global risk decision may potentially improve both decision consistency and operational efficiency. A conservative strategy would consist of blocking the forwarding of the alert message if at least one attack is detected. Otherwise, the cyber risks associated with different threats can be aggregated through quantitative or qualitative assessment techniques. The former includes methods such as the weighted average [27, 28], ordered weighted average [28], and summation-based approaches [29]. The latter, instead, associates risks with a ranking of threats based on their severity.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT-4 in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] N. Alshuqayran, N. Ali, R. Evans, A systematic mapping study in microservice architecture, in: 2016 IEEE 9th international conference on service-oriented computing and applications (SOCA), IEEE, 2016, pp. 44–51.

[2] A. Pereira-Vale, E. B. Fernandez, R. Monge, H. Astudillo, G. Márquez, Security in microservice-based systems: A multivocal literature review, Computers & Security 103 (2021) 102200.

[3] P. Haindl, P. Kochberger, M. Sveggen, A systematic literature review of inter-service security threats and mitigation strategies in microservice architectures, IEEE Access 12 (2024) 90252–90286.

[4] D. Berardi, S. Giallorenzo, J. Mauro, A. Melis, F. Montesi, M. Prandini, Microservice security: a systematic literature review, PeerJ Computer Science 8 (2022) e779.

[5] P. Civile, Common Alerting Protocol - Italian Profile, Technical Report, Dipartimento della Protezione Civile, 2024. URL: https://www.protezionecivile.gov.it/static/60f148a4a63756446dfb15acab736dbe/all7-dpc-edxl-cap-it-10.pdf.

[6] OASIS, Common Alerting Protocol Version 1.2, Technical Report, 2010. URL: https://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html.

[7] R. Chandramouli, Z. Butcher, A. Chetal, et al., Attribute-based access control for microservices-based applications using a service mesh, NIST Special Publication 800 (2021).

[8] N. Sänger, S. Abeck, User authorization in microservice-based applications, Software 2 (2023) 400–426.

[9] N. Xi, J. Liu, Y. Li, B. Qin, Decentralized access control for secure microservices cooperation with blockchain, ISA transactions 141 (2023) 44–51.

[10] A. Venčkauskas, D. Kukta, Š. Grigaliūnas, R. Brūzgienė, Enhancing microservices security with token-based access control method, Sensors 23 (2023) 3363.
[11] A. Nehme, V. Jesus, K. Mahbub, A. Abdallah, Fine-grained access control for microservices, in: International Symposium on Foundations and Practice of Security, Springer, 2018, pp. 285–300.
[12] D. Preuveneers, W. Joosen, Towards multi-party policy-based access control in federations of cloud and edge microservices, in: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, 2019, pp. 29–38.
[13] S. Pallewatta, M. A. Babar, Towards secure management of edge-cloud iot microservices using policy as code, in: European Conference on Software Architecture, Springer, 2024, pp. 270–287.
[14] J. Tan, Ensuring component dependencies and facilitating documentation by applying open policy agent in a devsecops cloud environment (2022).
[15] M. Baldo, F. I. Ion, M. Miculan, M. Paier, V. Riccio, et al., Owsm: Empowering rego for stateful access control, in: CEUR WORKSHOP PROCEEDINGS, volume 3962, CEUR-WS, 2025.
[16] S. Xu, Q. Zhou, H. Huang, X. Jia, H. Du, Y. Chen, Y. Xie, Log2policy: An approach to generate fine-grained access control rules for microservices from scratch, in: Proceedings of the 39th Annual Computer Security Applications Conference, 2023, pp. 229–240.
[17] X. Li, Y. Chen, Z. Lin, X. Wang, J. H. Chen, Automatic policy generation for inter-service access control of microservices, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 3971–3988.
[18] H. Khambhammettu, S. Boulares, K. Adi, L. Logrippo, A framework for risk assessment in access control systems, Computers & Security 39 (2013) 86–103.
[19] A. A. Ganin, P. Quach, M. Panwar, Z. A. Collier, J. M. Keisler, D. Marchese, I. Linkov, Multicriteria decision framework for cybersecurity risk assessment and management, Risk Analysis 40 (2020) 183–199.
[20] H. M. Tusher, Z. H. Munim, T. E. Notteboom, T.-E. Kim, S. Nazir, Cyber security risk assessment in autonomous shipping, Maritime economics & logistics 24 (2022) 208–227.
[21] M. U. Aksu, M. H. Dilek, E. İ. Tatlı, K. Bicakci, H. I. Dirik, M. U. Demirezen, T. Aykır, A quantitative cvss-based cyber security risk assessment methodology for it systems, in: 2017 International Carnahan Conference on Security Technology (ICCST), IEEE, 2017, pp. 1–8.
[22] A. Alvarenga, G. Tanev, A cybersecurity risk assessment framework that integrates value-sensitive design, Technology Innovation Management Review 7 (2017).
[23] S.-C. Cha, K.-H. Yeh, A data-driven security risk assessment scheme for personal data protection, IEEE Access 6 (2018) 50510–50517.
[24] M. Kalinin, V. Krundyshev, P. Zegzhda, Cybersecurity risk assessment in smart city infrastructures, Machines 9 (2021) 78.
[25] A. A. Elmarady, K. Rahouma, Studying cybersecurity in civil aviation, including developing and applying aviation cybersecurity risk assessment, IEEE access 9 (2021) 143997–144016.
[26] M. Abdulsatar, H. Ahmad, D. Goel, F. Ullah, Towards deep learning enabled cybersecurity risk assessment for microservice architectures, Cluster Computing 28 (2025) 350.
[27] A. A. Ganin, P. Quach, M. Panwar, Z. A. Collier, J. M. Keisler, D. Marchese, I. Linkov, Multicriteria decision framework for cybersecurity risk assessment and management, Risk Analysis 40 (2020) 183–199.
[28] S. Miller, C. Wagner, U. Aickelin, J. M. Garibaldi, Modelling cyber-security experts' decision making processes using aggregation operators, computers & security 62 (2016) 229–245.
[29] D. Palko, T. Babenko, A. Bigdan, N. Kiktev, T. Hutsol, M. Kuboń, H. Hnatiienko, S. Tabor, O. Gorbovy, A. Borusiewicz, Cyber security risk modeling in distributed information systems, Applied Sciences 13 (2023) 2393.