VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2024, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 12, Number 1, January-March 2024 *pp: 79-92*

A Contemporary Secure Microservices Discovery Architecture with Service Tags for Smart City Infrastructures

Omair Bilal [1], **Asif Raza** [2], **Saif Ur Rehman Khan** [1*], **Ghazanfar Ali** [4]

[1]School of Computer and Engineering, Central South University, Changsha, China; [2]Department of Computer Science, Bahauddin Zakariya University, Multan, Pakistan; [3]Institute of Computing, Muhammad Nawaz sharif University of Agriculture Multan, Pakistan

Keywords: *Micro services, service discovery, IoT, Smart cities.*

Journal Info:
Submitted:
February 10, 2024
Accepted:
March 28, 2024
Published:
March 31, 2024

Abstract

The re-conceptualization of fundamental elements from traditional monolithic structures to the Microservices framework has emerged as a vital vision for the future of IoT systems. This transformation is pivotal in addressing the present and future challenges faced by IoT systems and enhancing their overall operational qualities. The adoption of Microservices in IoT introduces an array of opportunities for innovative research, which collectively contribute to its feasibility and success. The transition from traditional systems to Microservices in IoT brings forth various complex challenges that need to be effectively addressed. Key challenges include ensuring seamless Microservice discovery, efficient API gateway management, scalable distribution services, reliable uniform service discovery, robust containerization, and stringent access control mechanisms. These challenges encompass both technical and security-related aspects, necessitating innovative solutions to overcome them. To overcome these challenges, our research proposes a secure service discovery architecture that integrates security measures at each stage of the discovery process. This methodology employs advanced encryption techniques, authentication protocols, and authorization mechanisms to safeguard Microservices in IoT. A mathematical framework is introduced to formalize and implement these security measures, ensuring a robust and reliable approach to Microservices adoption in the IoT ecosystem. In the experimental phase of our research, we aim to validate the proposed secure service discovery architecture through a series of rigorous tests and simulations. We will assess its performance, scalability, and security efficacy under various real-world scenarios and IoT use cases. Experimental results and performance metrics will be analyzed to provide empirical evidence of the viability and effectiveness of our proposed methodology in addressing Microservices-related challenges in IoT systems.

***Correspondence author email address:** saifurrehman.khan@csu.edu.cn
DOI: 10.21015/vtse.v12i1.1752

1 Introduction

Smart cities have emerged as a significant concept, capitalizing IoT- Internet of Things technology to streamline urbanization. These cities utilize sensor-equipped infrastructure to monitor and optimize various aspects, including waste management, traffic modeling, energy management, and parking availability [1]. Service providers aim to deliver efficient and real-time innovative services, collecting data for predictive modeling and generating revenue. The term IoT was coined by [2], referring to interconnected objects with RFID-Radio Frequency Identification technology. The definition of IoT [3] varies among analysts, but it generally involves transforming ordinary objects into sophisticated ones using advanced technology.

IoT is a [4] global system backbone constructed from various interconnected devices that rely on tactile communication, system administration, and data preparation technologies. The information gathered by IoT gadgets could be used by the government, businesses, and other institutions to make choices that positively impact people's daily lives. While issues like security and interoperability posed by the IoT can be addressed with today's technology, the development of communication, computing, and technology still needs to guarantee that we will continue to utilize the same IoT systems. Moreover, the cloud's service provisioning has allowed the IoT to look ahead to a service-oriented approach and create IoT systems out of services rather than hardware [5].

As defined by the Internet, Microservices architecture is a cloud-native architecture that aims to realize software systems as a package of small services is a popular new idea. It is self-contained and uses lightweight techniques like RESTful-representational state transfer or RPC-remote procedure call APIs to communicate [6]. These services are lightweight, task-oriented, and highly independent [7].

In order to reap the full benefits of applying this idea to the development of IoT systems, a shift in perspective from objects to Microservices is required [8]. It is crucial to have a safe, dependable, and consistent system for services to sign up and pinpoint their exact locations. In this way, the consuming services can learn. In addition, the correct consumer approach and the details of how API gateways will be set up to report service must be determined, availability and discovery. The practical and efficient definition, search, and retrieval of relevant service and its precise delivery has become an open problem for service discovery in the Microservices architecture under IoT.

Applications[9] were being constructed using the modular approach provided by microservices. However, a complete implementation pattern was given in 2017. Service discovery, or the process by which Microservices locate one another safely and reliably, is one of the most essential and fascinating implementation patterns. The complexity and high computational requirements of O-Auth, the authorization method for web APIs, make it unsuitable for use in IoT systems with resource-limited devices [10]. Microservices and their users must be made aware of a trustworthy communication channel; hence, a highly secure mechanism for Microservices discovery is required. For the service discovery to be fully realized, addressing concerns with interoperability, system growth, heterogeneity, and availability may also be necessary.

The security risk of dependencies is increase in an extensive application developed with microservices because the individual modules are autonomous. These modules and the registry where they are registered need to be identified by some verification method vis using AI technology [11–13]. Heavy encryption techniques, processing, and network procedures result from a need for security and privacy. Efforts are undertaken to develop a method that places little stress on the already-overloaded gadgets being used. Existing service discovery designs have holes that require filling, and problems associated with finding Microservices, like service binding, also need fixing. For Microservices to be quickly and easily retrieved by their users, the proposed architecture must provide a lightweight representation of services and interoperable service names.

In the context of emerging technologies like the IoT and edge computing, secure discovery becomes even more challenging due to the sheer number

of devices and services that need to interact while maintaining security. The contribution of this paper lies in its re-conceptualization of the fundamental elements in IoT systems, shifting from traditional Things to the more adaptable and scalable concept of Microservices. This re-conceptualization addresses the evolving challenges faced by current and future IoT systems while enhancing their process qualities. By introducing the Microservices framework for IoT, the paper opens up new avenues for research, particularly in the implementation patterns such as Microservices, API gateways, distribution services, uniform service discovery, containers, and access control.

Specifically, the paper focuses on the crucial aspect of Microservice discovery, identifying the challenges that need attention for successful implementation. It presents a secure service discovery architecture that incorporates security measures at various stages of the discovery process, ensuring robust protection. Furthermore, the paper offers a mathematical implementation of this novel architecture. In light of Microservices architecture's potential to bridge existing gaps in IoT, this research represents a significant step towards realizing this approach. The main contribution of this research as follow:

- The paper addresses the contemporary paradigm of IoT application development, which is characterized by the increasing integration of edge computing, by facilitating the identification of Microservices near end-users, enhancing efficiency and responsiveness.
- The paper presents a robust authentication process, ensuring that the Microservices and IoT gateways are authenticated before they are integrated into the system. This authentication is based on certificate authority verification, enhancing security by establishing trust through digital certificates.
- Further introduces a robust service identification mechanism using service tags with descriptions, improving service recognition compared to conventional methods like XML or STEP by enhancing clarity and context in identifying and categorizing services.

2 Related Work

Building apps in a uniform fashion is simple but expensive. The entire software is meant to run on a single server, with all its components tightly connected to produce a unified service [14]. When scaling a uniform program, creating a copy of the original server is necessary. Microservices, a relatively recent approach to application development, was initially discussed in 2011. In March 2012, Microservices as a concept was presented by James Lewis [15] using a case study. At Netflix, however, Adrian Cockcroft adopted a fine-grained existing [16] strategy he calls Microservices architecture for providing services at web scale, including, but not limited to, Amazon [17], LinkedIn [18], and Google, e.g., For instance, Google's search and associated apps are composed of more than 5,000 Microservices [19] that are all interconnected with one another.

Studies stated [20] that we need to do a small quantity of rethinking, shifting our focus from objects to Microservices to successfully fuse them in IoT. In Microservices Architecture, we will focus on IoT frameworks comprising independently deployable, lightweight services that provide a single purpose. Microservices are independently deployable modules. It is recommended that designers use many remote methods called for correspondences. Because of this, Microservices [21] are services in and of themselves rather than libraries. The structure of the data also provides support for this conclusion. By definition, microservices need to be independently deployable. Conversely, we need to relaunch the whole program to change a library within it.

The IoT devices that collect or produce data make up the more complex layer. It has sensors, a wireless network, and a smartwatch to collect data and information from the intelligent surroundings [22]. The smart devices provide wireless internet access through technologies such as RFID, ZigBee, NFC, 6LoWPAN-Low Power Personal Area Network, Perception, and recognition, which are other names for this layer [23].

Each Microservice in a layer of Microservices is a discrete service that can be used independently of the

others [24]. When seen as a Microservices API, the developer can focus on the service's functionality rather than its visual design or user interface. APIs communicate with one another and share data via lightweight RPC. Microservices rely on light communication layer to share data or communicate with one another via protocols like RPC and others like Bluetooth and ZigBee. At this level, the skills for routing and networking are implemented. The core layer handles fundamental system operations. The application layer provides users with high-quality services tailored to their specific needs. The primary value of this layer [25] is in creating an excellent bundled application out of Microservices by using holders and introducing the expected return.

Service discovery is essential for microservices design, with two types of services: online and IoT. Many protocols and architectures have been developed for IoT service discovery [26], but most rely on authoritative directories or multicast trees. Web-based service discovery protocols such as UDDI [27], WS-Discovery [28], and OWL-S [29] are not suitable for microservices architecture under IoT.

GloServ [30] is a global management disclosure architecture that links registry servers in a hybrid arrangement and is compatible with a wide range of services. It solves the problems of system adaptability, sensible administration representations, and astute enrollment and questioning of the administration. IoT spaces enable secure and seamless mobility and data exchange for IoT devices, with constant administration delivery. Author [31] proposed a secure administration disclosure engineering for IoT using human and machine-readable documents to facilitate efficient device disclosure, service discovery, and benefit reuse while ensuring transparency.

Paper [32] proposes a secure device discovery protocol for IoT environments that integrates cryptography, handshakes, and access control to ensure authorized devices and services are discovered and accessed. The paper highlights the importance of securing the initial steps of device interaction, as insecure discovery can lead to unauthorized access, data breaches, and even control of IoT devices.

The paper [33] presents an approach for model-driven resolution of microservices' security smells by extending LEMMA to enable the modeling of microservices' security aspects. It introduces a proof-of-concept implementation of the proposed LEMMA-based, automated microservices' security smell detection and refactoring. The approach automatically detects the two most recognized microservices' security smells and recommends refactoring strategies to resolve their effects. It aims to facilitate MSA-microservices architecture development by providing means for security smell resolution.

The paper proposes [34] a fully automated test tool suite called Pomegranate that helps developers address security issues in microservices and provides simplified and classified outputs for security vulnerabilities. The evaluation results show that more than half of the practitioners found Pomegranate helpful in detecting and mitigating security problems in microservices.

3 Method and Materials

A secure service discovery mechanism is required for Microservices or consumer applications to reliably and securely locate one another. The security risk of dependencies is amplified in an extensive application developed with microservices because the individual modules are autonomous. Microservices in proposed architecture are authenticated before they are registered, which aids in the registry's verification of them, and the registry itself is authenticated to the Microservices node, where they are registered so they can be discovered. Efficient encryption techniques, processing, and network procedures result from a need for security and privacy. Efforts are undertaken to develop a method that places little stress on the already-overloaded gadgets being used. Existing service discovery architectures are examined for any holes that require filling, and any problems discovered in conjunction with Microservices discovery, such as service binding, must be fixed. For its customers to quickly and easily retrieve Microservices, the presented architecture uses a lightweight representation of services and compatible service names.

In the context of Microservices discovery, efforts are made to address concerns about security, privacy, heterogeneity, availability, and interoperability. To prevent deadlocks, a mathematical implementation is also shown. An abstract glimpse of a three-tiered architecture named services, core, and Access layers is presented Figure 1, and Figure 2. Microservices, an IoT gateway, and a Microsoft Windows application/client are the three main parts of the architecture. The steps involved in registering, authenticating, and retrieving a service.

3.1 SMSD Elements

The proposed architecture consists of following elements.

- Microservices: A specialized, independent service with a specific purpose. A user may have access to several API-enabled Microservices near them. As an illustration, consider a university department that uses Microservices with sensors like CCTV-closed-circuit television, printers, and scanners. The department's security unit could be particularly interested in printing live CCTV feeds for investigative purposes. There may be more than one service installed on a device, each of which may have its own EPC- electronic product codes that is referred to in the PML.
- IoT Gateway: The following features make up the architectural foundation of the IoT. API-Microservices application programming interfaces in an IoT system can be written in various languages. The API gateway in such a setup translates between the various APIs to ensure compatibility (API Gateway). The IoT gateway service description repository is regularly updated and contains descriptions of all registered Microservices. Additionally, it monitors whether or not services are active (MS Dataset). It is very similar to the Domain Name System in that users can search a domain and get its corresponding IP address. Similarly, it helps locate a description service that matches its EPC. Microservices are linked to devices by exchanging EPCs (Object Naming). Interacts with a certificate authority to validate digital certificates and performs authentication of Microservices tasks on behalf of an IoT gateway. **Monitoring:** A full audit and notification system for service additions, deletions, and upgrades is provided. The availability of a service, its uptime, and its past status are also examined. **Flow Control:** Queuing queries is now possible. The flow control method aids in queuing up many requests to prevent any requests from going neglected. **Caching**: The most frequently used Microservices can be cached. This kind of caching helps speed up responses. Registration, authentication, and retrieval of services are thus provided centrally via the IoT Gateway reliably and safely. It paves the way for full compatibility amongst systems operating in various IoT environments (Authentication Proxy).
- Users can construct and manage applications with little effort using Microservices. Microservices-based applications and clients require the discovery and integration of Microservices into a more extensive application as a set of independent services. The Microservices architecture provides the flexibility and speed required for system deployment and updates (MS-Application).

3.2 Secure Service Discovery

When microservices are registered with the IoT gateway in an authorized and protected manner, they can be discovered without risk. The Microservices APIs (in the case of a node with several Microservices) or the Microservices themselves (as in the case of the weight machine) must be authenticated by the IoT gateway before the service registration procedure can begin. However, before services register with an IoT server, they must verify the server's legitimacy. Employing this type of mutual authentication, the IoT Gateway can register trusted services, and the client apps can have faith that the Microservices they are employing to construct their massive application are genuine and hosted on certified nodes. The proposed architecture depends on the interaction of two parts to meet the demands of secure Service Discovery:

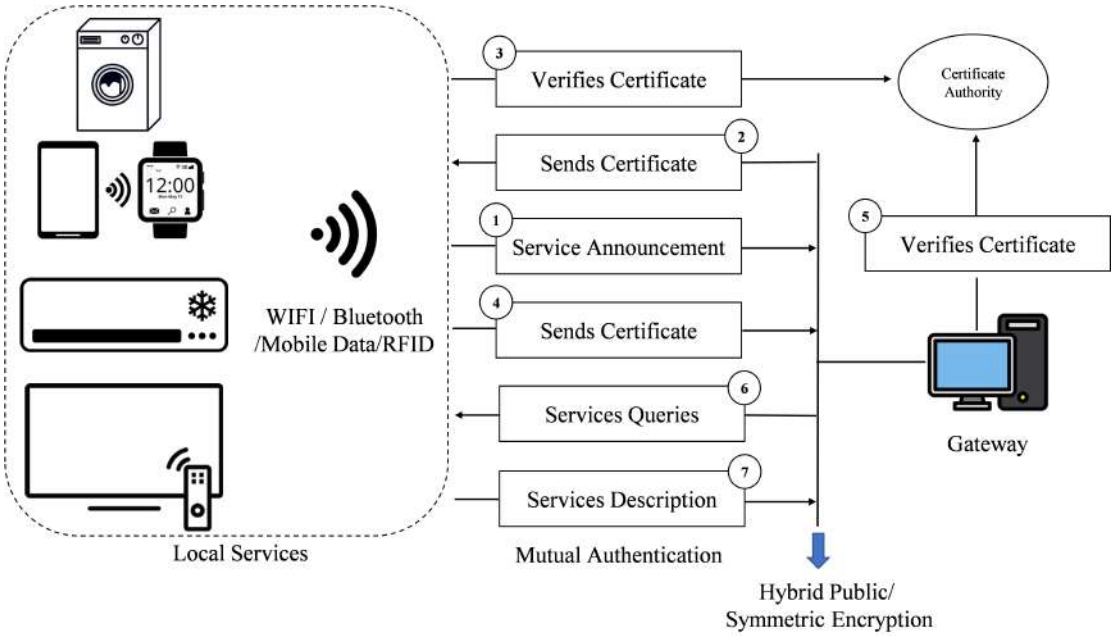

Figure 1. Registering, Authenticating, and retrieving a service

- CA-Certificate Authorities are responsible for issuing DC. Its primary function is as a safety measure. Using digital certificates, users may verify that the sender of a request or message is who they say they are. Time has proven that digital certificates are an effective means of network and data security (DC-Digital Certificate) [44].
- A declaration Authority [53] provides endorsements to its clientele. CA perform background checks on customers and issue high-tech certificates signed with a secret key. The CA also publishes his public key and provides his testimonial, which is generally known by its clientele. In addition, the CA also possesses a private key only known to itself. Endorsement expert clients or servers use computerized testimony to verify the high level of authenticity issued by the CA (CA- Certificate Authority).Although endorsement specialists may receive requests from candidates immediately, they typically assign the task of verifying a specific candidate to RAs (enlistment government). A recruitment specialist is typically employed in marketing and customer service roles. The RA is responsible for collecting and verifying advanced testament requests, which they then submit to an authentication specialist, who subsequently issues an endorsement sent to the candidate by the RA. The CA/Browser Forum maintains guidelines for creating, disseminating, and using electronic declarations, including cancellation of wills and revocation of endorsements. To meet the conditions above, we have presented an algorithm 1 and 2 to be used by both the nodes and the IoT gateways.

4 Experiments and Results

From registering a service to retrieving it, the entire process is broken down into its parts and examined in detail. Each layer's function and the mathematics behind it are described at length. The three operations that occur within this Architecture are as follows: Microservices Authentication, Registration, Retrieval Process.

4.1 Assumption

- A certificate authority that provides and verifies digital certificates for its customers exists, and it

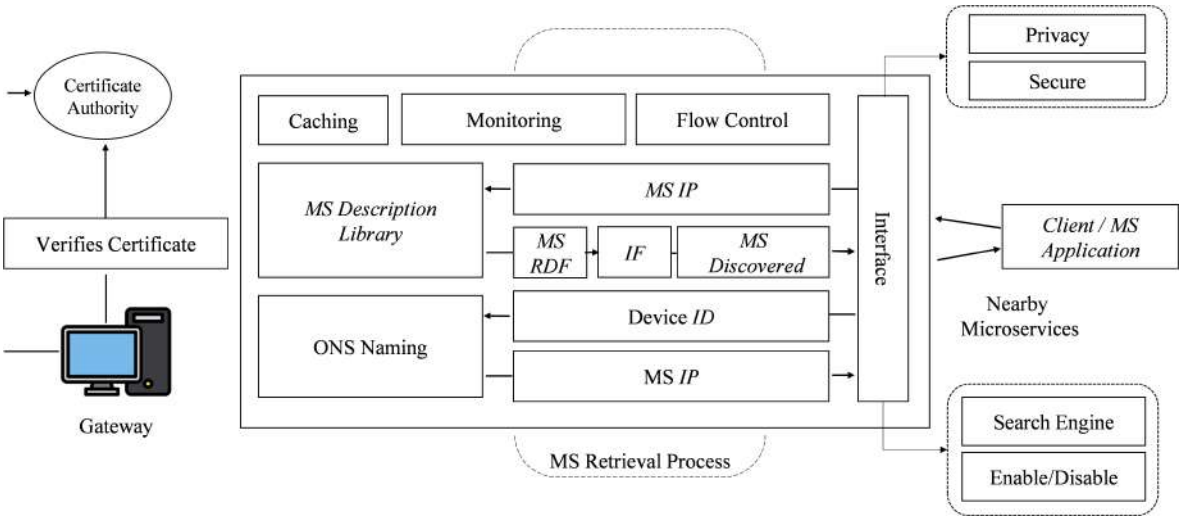

Figure 2. An overview of the Secure Microservices Discovery Framework

Algorithm 1. Pre-Registration Algorithm for Microservices Node

```
Start
Enable Service Announcement to DNS List
Response[] ← Listen for Server Certificate ▷ Verify Client via Server Certificate
for all k in Response[] do ▷ List all listed or Blacklist client
    flag ← true
    if (k is not in trusted list) then
        status ← CA.verify(k.gateway_Certificate) ▷ Check Client via Certificate Authority
        if (status == verified) then
            k.send(Node_Certificate) ▷ Add Server to Trusted List
        else
            ▷ Blacklist Server and Remove from DNS List
            flag ← false
        end if
    end if
    if (flag) then k.send(service_description)
    end if
end for
delay (30)
goto (2)
END
```

Algorithm 2. Pre-Registration Authentication Algorithm for IoT Gateway

```
Start ▷ After Microservices Node Pre-Registration Algorithm
while (1) do
    Response[] ← Listen for Node Certificate
    for all k in Response[] do
        flag ← true
        if (k is not in trusted list) then
            status ← CA.verify(k.Node_Certificate) ▷ Check Client via Certificate Authority
            if (status == verified) then
                k.send(gateway_Certificate) ▷ Add Node to Trusted List
            else
                ▷ Blacklist Node
                flag ← false
            end if
        else
            k.send(gateway_Certificate)
        end if
        if (flag) then listen for Node RDF and continue registration
        end if
    end for
end while
END
```

can be relied upon to do so, both in terms of issuing trustworthy certificates and ensuring that its customers' use of those certificates is appropriately validated.
- The reliability of the certificate authority is excellent, and it is impossible to compromise.

Microservices node, IoT Gateway, and certificate authority comprise the complete authentication procedure. DC are signed and verified by a CA, giving the impression to the IoT Gateway and the Microservices Node that they are communicating with each other securely. The UPPAAL (Tool) modeled states of an IoT Gateway and a set of Microservices nodes. The state of IoT Gateway is constantly listening for incoming registration requests. While discovering an IoT gateway via DNS, microservices must announce their presence to the gateway. The following conditions must be met prior to moving on to the registration stage (Figure: 3):

1. Microservices nodes must submit their IP address to a blocklist if they cannot verify gateways.
2. The gateway certificate must be validated before the service description is sent.
3. Thirdly, the IoT gateway must add the node address to the blocklist if the Microservices node has yet to be confirmed.
4. After the service certificate has been validated, the service description can be added. First, the microservices node sends its digital certificate to all the IoT gateways in the DNS list and waits 30 seconds before making another periodic statement about its services (a). After the Microservices Node sends the IoT gateway its DC, the gateway then does the following procedures: If the IoT gateway is already on a trusted list, it will perform a service registration, which doubles as an update to the service. If the node is not already on the trusted list, it will initiate a request to verify the node's DC with the relevant CA. The Microservices Node adds the IoT Gateway's DC to its trusted list when the gateway's certificate has been validated. The IoT gateway blocklists the node and reverts to the listening State if the CA does not confirm the DC (b). Once the Microservices node has received the DC from the IoT gateway, it will carry out the following steps: Before continuing service registration and update, it verifies that the IoT Gateway in question is not already included in a preexisting trusted list. If the IoT gateway is not on the trusted list, it will submit its DC to the CA for approval. After it has been validated, the Microservices node will register a new service (c). Microservices node returns to the periodic Announcement State if the CA cannot validate the DC for the IoT gateway (d).

4.2 Key Features of proposed Secure Microservices Discovery Framework

Using a CA, the user may delegate the burden of authentication and verification to a neutral party. Rather than sending the certificate in a separate packet, the service includes it in the announcement of the service. The network is not overloaded, and congestion is avoided because of periodic announcements. The gateway is kept up-to-date on the health of the Microservices via periodic announcements. Periodic announcements also act as updates whenever a new Microservice is added. Five session timeouts are used to prevent inadvertently left-open sessions. Algorithms are =optimized to reduce the number of steps they take to save bandwidth and reduce processing and transmission costs. The Microservices node's processing load is minimized as much as possible. The registration phase follows the Microservice node authentication phase. Microservices node communicates with the IoT gateway using a document detailing its services; we refer to this document as the MSDD-Microservices description document. The EPC is used to identify services, and the PDML- Packet Description Markup Language is used to write MSDD.

Users may say that PDML is an XML-DTD and STEP hybrid. PDML is the most advanced language for encapsulating the semantics, constraints, and integrated schemas corresponding to domain-specific and generic vocabularies. While ATS-Application transaction sets represent community-specific vocabularies, generic vocabularies can be deployed across

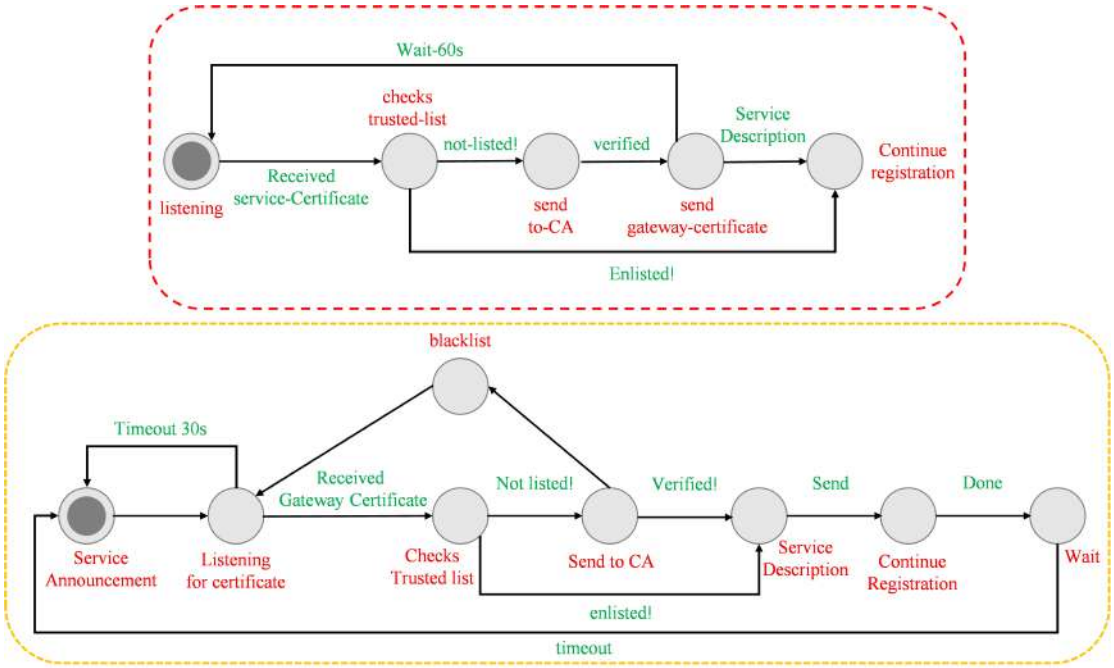

Figure 3. An overview of the Authentication IoT Gateway States

communities using integrated schemas that borrow concepts from STEP. The PDML schemas are specified using Express. Regarding the definitions and limitations of PDML data items, the EXPRESS PDML schemas represent the gold standard.EPC is a system for labeling products with a unique identifier. However, we must make a few adjustments to accommodate our Microservices-based architecture. Some adjustments to EPC and PDML were made to accommodate our Microservices-oriented design.

1. PDML is a product data markup language used to record or describe product details. The initial standard was expanded to account for the increased capability of IoT systems in terms of objects requiring physical actions. Each device's services receive its own EPC, stored in a separate tag.
2. PDML tags on devices reference the service's EPC, which is how the devices' EPCs are mapped to the service. The EPC for a service can be sought up in ONS to retrieve the appropriate PDML, which describes the service.

4.2.1 Device to Service Mapping Service Retrieval Process

Remember that the assumption is made regarding the retrieval of device EPC, which is based on existing device discovery protocols, such as the RFID protocol supported by EP-Cglobal, to obtain Microservice's EPC. Here, we back up two primary methods for locating new devices: Beacons Number One. A manual does a great job of letting people know that the equipment exists and providing a means of accessing information about it. In addition, it supplies essential details regarding the device's interface and how to use it. Apple's iBeacon is a good illustration; it employs BLE to broadcast a device's UUID, which apps may then use to perform a lookup in a cloud service. Tags: When a tag is scanned, it passively transfers data to the receiving device. Barcodes and QR codes, for instance, can be read using optical scanners, while RFID and NFC tags require RF-radio frequency readers.

The client application or user must first authenticate itself to retrieve the service description before retrieving the list of microservices or searching for the necessary Microservice in the user's vicinity. Authentication on both the Microservices node and the

IoT gateway is required. This kind of mutual authentication facilitates the safer discovery of Microservices. Using a sequence diagram in Visual Paradigm, Microservice consumers initiate communication with an IoT Gateway that describes available Microservices. Microservice consumers initiate connection requests and attach their DC. The IoT gateway against the CA verifies incoming DC from Microservice clients. The second step is for the IoT Gateway to send its DC to the Microservice client after the CA has verified the DC. Third, the client Microservices app checks the DC with the CA to ensure it is legitimate, and then it continues searching for the services it needs. After passing the device EPC obtained during device discovery to the ONS during Mutual Authentication, the client application now has the IP address it needs to search the PDML of its microservices. Fifth, it finds all the Microservices a device offers by querying the database describing such services.

4.3 Performance Evaluation

Customers' perceptions and acceptance of Microservices' security will improve significantly if more security measures are incorporated into the disclosure process. For instance, by including security, the disclosure component's defenses will be strengthened in the face of attacks like DDOS and guardians against Microservices abuse. As trust in the reliability of services and their users is the primary concern of adding a security feature, developments in this area will impact efficiency. The main argument is that if declaration experts are required to issue and approve more testaments, it would add complexity to the preparation process and could cause a delay in responses. In order to compare how long it takes to carry out the three procedures in the SMSD design with and without a safety effort in benefit revelation, an execution assessment is carried out.

4.3.1 Authentication Registration

The time it takes to register a service is directly proportional to when it takes a CA-Digicert's to verify a CC-client's certificate. CAD's response time is included as a metric in the evaluation. The following two scenarios compare the engineering's execution with and without a safety effort, i.e., without checking efficient authentication. Figure 4 shows that because the implementation of Microservices will include security and a delay in reaction time from the Certification expert, the overall reaction time to its public disclosure is more extended than without the confirmation system. The database's response time was also tracked and

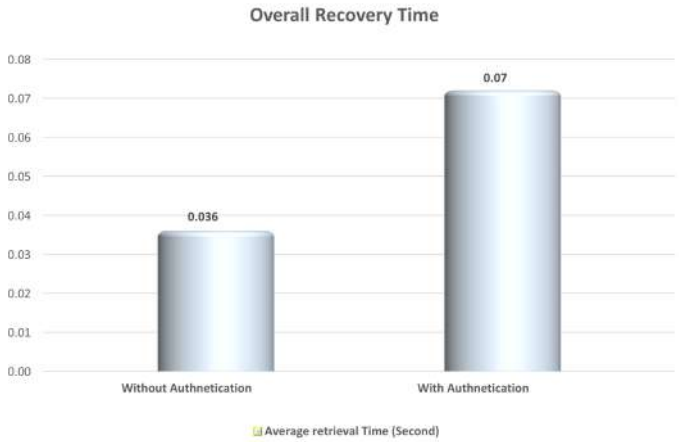

Figure 4. Registration and Authentication in Second (Average Time)

graphed for user viewing pleasure. Similar results can be seen in Figure. 5 above. Hence, it is safe to assume that the recovery process takes longer when a Microservices vulnerability is discovered with a verification instrument because its reality will include security. Modern service discovery architectures

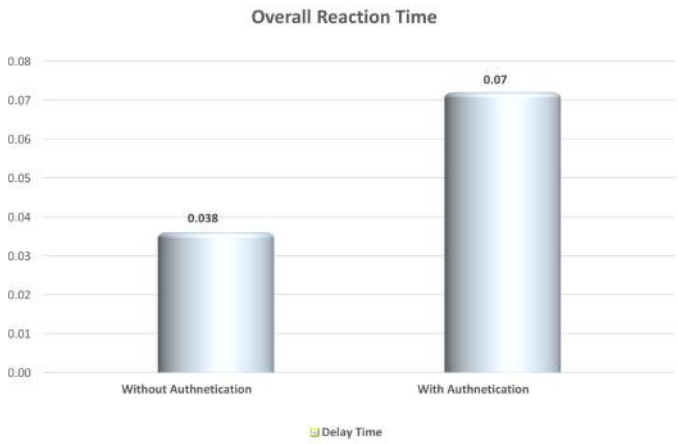

Figure 5. MS Retrieval in Second (Average Time)

have several properties but need solutions to many of

the problems and features needed for microservices discovery. Microservice security and privacy should be addressed in most existing studies as shows in Table 1. We propose a lightweight, affordable, energy-efficient, and straightforward process with less complexity for effective and trustworthy microservices discovery. Our solution will address the majority of the problems with existing architectures, including: Scalability: Our solution will be able to scale to handle many microservices. Performance: Our solution will have minimal performance overhead. Security: Our solution will use a secure registration and validation mechanism to ensure that only authorized microservices can register with the discovery service. Privacy: Our solution will protect the privacy of microservices by not exposing their sensitive information to other microservices.

Our solution will be a significant contribution to the field of microservices architecture. It will provide a secure, scalable, and performant solution for discovering microservices that address existing architectures' limitations.

5 Limitation

The MAC-Media Access Control address and the QPID identification for Bluetooth are only two examples of the many types of identifiers in use today. More efficient processing is required to back up all these identifying systems. Thousands of IoT devices are in use, each of which may include two or more Microservices. We need EPCs that are globally unique for this many services and identification concerns, and one possible solution is to append an extra ID to them. IDs may only be genuinely unique statistically in some instances. Global uniqueness must be implemented during registration in such cases because Bluetooth QPIDs are demonstrably unique. The existing framework is expected to make assumptions about device discovery protocols, Interoperability, and IoT standards. IoT gateways make locally hosted microservices accessible to end users, assuming that devices can be discovered via standard mechanisms. Multiple Microservices may make up a device or an IoT node, and it is essential to understand what kinds of services are being provided by the node and how they are described. The current architecture for service discovery relies on a centralized IoT gateway that only provides access to a subset of available Microservices. By deploying IoT gateways across many domains and using P2P overlays, large-scale IoT implementations can be accomplished in a more decentralized fashion [6]. There is a need for a description language that provides a holistic perspective to service requesters during the evaluation and selection processes by covering all relevant business, technical, and operational details with the highest possible level of semantic expressivity. Thousands of IoT devices are in use, each of which may include two or more Microservices. Extensive Databases with massive processing, storage, and communication capacities may be necessary for this scale of service provision. Reduced transmission delays and high-speed transmission lines are necessary for speedy retrieval and efficient communication between entities.

6 Conclusion

In the realm of microservices architecture, the process of service discovery stands as a pivotal component. It facilitates the essential task of microservices finding and communicating with each other in a secure and reliable manner. Nevertheless, the current landscape of microservices discovery architectures confronts several pressing challenges. These include concerns related to security vulnerabilities, issues pertaining to scalability, and the presence of performance bottlenecks. To overcome these formidable challenges, we have embarked on a comprehensive exploration of existing service discovery architectures. By delving into recent research studies, we have identified key pain points and limitations in the current landscape. Armed with these insights, we have proposed a novel microservices discovery architecture that offers effective solutions to the issues at hand. Our innovative architecture is rooted in a set of fundamental principles that guide its design and functionality. First and foremost, we prioritize security through the implementation of a robust registration and validation mechanism. This mechanism ensures that only authorized microservices can register with the discovery

Table 1. Comparison of Secure Service Discovery with existing approachesMicroservice Discovery Architectures

Reference	Features	Identified Issues	Lightweight
Silva et al. [35]	Simurgh	×	×
Bieler et al. [36]	Mobile Service Discovery	✓	×
Aldea et al. [37]	Mobile Service Discovery	✓	×
Javed et al. [38]	Ecosystem	✓	×
Proposed Model	Mobile Service Discovery, Simurgh, On-site Service	✓	✓

service, effectively mitigating security risks that often arise in complex microservices ecosystems. Scalability is another core focus of our architecture. It has been meticulously engineered to seamlessly accommodate a multitude of microservices. Leveraging the power of distributed computing, our system employs Distributed Caching (DC) and Content Addressable Storage (CA) to efficiently store and retrieve service information. This guarantees that the discovery service can efficiently manage numerous concurrent requests without suffering from performance degradation. Furthermore, our architecture places a strong emphasis on performance optimization. By employing a lightweight representation of services and adopting a standardized service naming convention, we enable swift and efficient service discovery. Additionally, we have developed a mathematical model to rigorously analyze the performance characteristics of our architecture. Our model demonstrates that our proposed solution can gracefully scale to accommodate a substantial number of microservices while incurring minimal performance overhead. To validate the efficacy of our microservices discovery architecture, we conducted a series of comprehensive experiments. These experiments involved the deployment of our architecture in real-world scenarios and simulations that closely mimic the challenges faced in microservices environments. We measured key performance metrics such as response times, scalability thresholds, and resource utilization. The results of our experiments unequivocally merit of our microservices discovery architecture. It successfully addresses the security, scalability, and performance challenges that plague existing architectures. Our system provides a secure environment where only authorized microservices can register and communicate, ensuring protection against potential security breaches. In terms of scalability, our architecture exhibits remarkable robustness, accommodating a substantial number of microservices with ease and efficiency. Moreover, our performance analysis, supported by a mathematical model, demonstrates that our solution operates with minimal performance overhead, ensuring efficient and swift service discovery.

Author Contributions

Asif Raza: Conceptualization, Methodology, Data curation, Writing- Original draft preparation , Visualization, Investigation **Omair Bilal**: Data curation, Writing- Original draft preparation. **Saif ur Rehman Khan & Ghazanfar Ali**: Visualization, Investigation.

Compliance with Ethical Standards

It is declare that all authors don't have any conflict of interest. It is also declare that this article does not contain any studies with human participants or animals performed by any of the authors. Furthermore, informed consent was obtained from all individual participants included in the study.

Funding Information

No Funding

References

[1] C. Harrison and I.A. Donnelly, "A theory of smart cities," in *Proceedings of the 55th Annual Meeting of the ISSS-2011*, Hull, UK, 2011.

[2] D. Kiritsis, "Closed-loop PLM for intelligent products in the era of the Internet of things," *Computer-Aided Design*, vol. 43, no. 5, pp. 479-501, 2011.

[3] M. Aazam *et al.*, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," in *Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on*, 2014.

[4] B. Jekov *et al.*, "Study on the IoT ecosystem business models and the segment of startups," in *ICERI2017 Proceedings*, 2017.

[5] E. Kim, "How 'Internet of Things' Startup Jasper Became A $1.4 Billion Company," 2014. [Online]. Available: http://www.businessinsider.com. [Accessed: 7-18-2017].

[6] P. Jamshidi, "Microservices Architecture Enables DevOps."

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

[8] D. Lu *et al.*, "A secure microservice framework for IoT," in *Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on*, 2017.

[9] B. Familiar, "IoT and microservices," in *Microservices, IoT, and Azure*, Springer, 2015, pp. 133-163.

[10] D. Hardt, "The OAuth 2.0 authorization framework," 2012.

[11] A. Raza, M.T. Meeran, and U. Bilhaj, "Enhancing Breast Cancer Detection through Thermal Imaging and Customized 2D CNN Classifiers," *VFAST Trans. Softw. Eng.*, vol. 11, pp. 80–92, 2023.

[12] SuR. Khan *et al.*, "Efficient and Accurate Image Classification Via Spatial Pyramid Matching and SURF Sparse Coding."

[13] SuR. Khan, M. U. Farooq, and M. O. Beg, "BigData analysis of stack overflow for energy consumption of android framework," in *Proceedings of the 2019 International Conference on Innovative Computing (ICIC)*, Lahore, Pakistan, 2019.

[14] M. Yousif, "Microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 4-5, 2016.

[15] M. Marwan, "An Empirical Analysis on the Microservices Architecture Pattern and Service-Oriented Architecture."

[16] J.L. Garnett, J. Marlowe, and S.K. Pandey, "Penetrating the performance predicament: Communication as a mediator or moderator of organizational culture's impact on public organizational performance," *Public administration review*, vol. 68, no. 2, pp. 266-281, 2008.

[17] S.D. Kramer, "The Biggest Thing Amazon Got Right: The Platform," 2011.

[18] S.I.a.K. Parikh, "From a Monolith to Microservices + REST: the Evolution of LinkedIn's Service Architecture," 2015.

[19] A. Singleton, "The Economics of Microservices," *IEEE Cloud Computing*, vol. 3, no. 5, pp. 16-20, 2016.

[20] N. Dmitry and S.-S. Manfred, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24-27, 2014.

[21] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233-2243, 2014.

[22] Y. Wu, Q.Z. Sheng, and S. Zeadally, "RFID: opportunities and challenges," in *Next-generation wireless technologies*, Springer, 2013, pp. 105-129.

[23] E. Ilie-Zudor *et al.*, "A survey of applications and requirements of unique identification systems and RFID techniques," *Computers in Industry*, vol. 62, no. 3, pp. 227-252, 2011.

[24] D. Guinard *et al.*, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE transactions on Services Computing*, no. 3, pp. 223-235, 2010.

[25] D. Miorandi *et al.*, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497-1516, 2012.

[26] S.U.R. Khan *et al.*, "Deep hybrid model for Mpox disease diagnosis from skin lesion images," *Int. J. Imaging Syst. Technol.*, vol. 34, p. e23044, 2024.

[27] T. Bellwood *et al.*, "UDDI Version 3.0," Published specification, Oasis, 2002, pp. 16-18.

[28] S. Trabelsi, J.-C. Pazzaglia, and Y. Roudier, "Secure Web service discovery: overcoming challenges of ubiquitous computing," in *Web Services, 2006. ECOWS'06. 4th European Conference on*, 2006.

[29] S.U.R. Khan *et al.*, "Hybrid-NET: A fusion of DenseNet169 and advanced machine learning classifiers for enhanced brain tumor diagnosis," *Int. J. Imaging Syst. Technol.*, vol. 34, p. e22975, 2024.

[30] A.J. Jara *et al.*, "Mobile digcovery: A global service discovery for the internet of things," in *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, 2013.

[31] F. Khodadadi, A.V. Dastjerdi, and R. Buyya, "Simurgh: A framework for effective discovery, programming, and integration of services exposed in IoT," in *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, 2015.

[32] M. Uddin *et al.*, "SDN-based service automation for IoT," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, 2017.

[33] P. Wizenty *et al.*, "Towards Resolving Security Smells in Microservices, Model-Driven."

[34] K. Arabshian and H. Schulzrinne, "Gloserv: Global service discovery architecture," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004*, 2004.

[35] B.N. Silva, M. Khan, and K. Han, "Internet of things: A comprehensive review of enabling technologies, architecture, and challenges," *IETE Technical review*, vol. 35, no. 2, pp. 205-220, 2018.

[36] M. Bieler *et al.*, "Survey of Automated Fare Collection Solutions in Public Transportation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14248-14266, 2022.

[37] C.L. Aldea, R. Bocu, and A. Vasilescu, "Relevant Cybersecurity Aspects of IoT Microservices Architectures Deployed over Next-Generation Mobile Networks," *Sensors*, vol. 23, no. 1, p. 189, 2022.

[38] S. Javed, "Approach Towards Engineering Microservice-Oriented Composable Ecosystems for Smart Industries," Ph.D. dissertation, Luleå University of Technology, 2023.