Containerization Approach for Secure Internet of Medical Things (IoMT) Communication Protocols

OYEDEMI OLUYEMISI ADENIKE[1]
DEMOSTENES ZEGARRA RODRIGUEZ [2]
RENATA LOPES ROSA [3]
UGOCHUKWU OKWUDILLI MATHEW [4]

Department of Computer Science, Federal University of Lavras, Minas Gerais, Brazil.

[1]oyedemi.adenike@ufla.br
[2] demostenes.zegarra@ufla.br
[3] renata.rosa@ufla.br
[4] ugochukwu.mathew@estudante.ufla.br

Abstract. The need for secure communication protocol has become very paramount due to the rapid growth of IoMT devices. Lack of robust security measure for communication protocol in IoMT environment renders them vulnerable to cyber-attacks. Secure communication protocols also address a major aspect of IoT security. This paper presents containerization-based framework as an approach that can be deployed, managed, and orchestrated on various IoMT devices and platforms for securing communication protocols. Containerization of communication protocol involves packaging communication protocols into containers. This approach provides a lightweight, portable, and secure way to deploy and manage communication protocols. Containerizing the communication protocols can be executed efficiently, reliably, securely without compromising the integrity and functionality of devices. By implementing robust and secure communication protocol, IoT environment can be better equipped to defend against potential cyber threats.

Keywords: Internet of Medical Things (IoMT), Containerisation, Security, Communication Protocol.

(Received November 19th, 2024 / Accepted December 29th, 2024)

1 Introduction

The increase in the number of Internet of Things (IoT) devices that are being used in the 21st century has generated tons of data in different sectors such as finance, personal healthcare, social media, traffic management, and the smart city [14, 18, 5, 6, 21, 8, 23]. These IoT devices are capable of collecting an enormous amount of data each day [16]. Some of these applications are shown in Figure 1 where each sector sends and receives data to and from the central server (cloud server). This rapid development of computing technology and wearable devices such as smart phones, smart watches, and wristbands has greatly transformed all sectors especially the health sector which makes it easy to get access to peopleâs health information including activities, sleep, and sports. A lot of data is being generated across the globe and it has quite unique properties. Participants make use of their personal mobile devices, Internet of Medical Things (IoMT) to track their health status without using a publicly accessible devices [17].

This technology is referred to as Mobile Health in a health care setting [1]. Figure 2 shows the scenarios where health data gathered from different homes are transmitted to the health practitioner for decision mak-

ing. With this, the abundant user health data accessed by Internet of Medical Things (IoMT) devices and recent development in machine learning, smart healthcare has seen many successful stories. With such systems, patients can receive better care even from their own homes, particularly for those with rare or geographically uncommon diseases.

The Internet of Medical Things (IoMT) consist of the integrated system of sensors, central controllers and remote locations. The sensors are responsible for taking input from users, central controllers may make local decisions and/or forward data to the remote locations which may finally make centralized decisions based on the values of input parameters [10]. With the ubiquitous proliferation of such personalized IoMT devices, collaborative and distributed learning is now more possible than ever to help best utilize the behavioral information learnt from multiple devices. One major issue of this advancement is security concerns [11]. As medical data are highly sensitive and private, some individual sources may not be willing to share their data with a central data collector [22] due to security concerns from multiple participants. This therefore makes the issue of secured communication protocol for device management, data transfer, and connectivity very crucial. Devices communicate with each other, and with the server through various communication protocols creating a network that offers real-time data and insights [2]. The protocols such as Con-

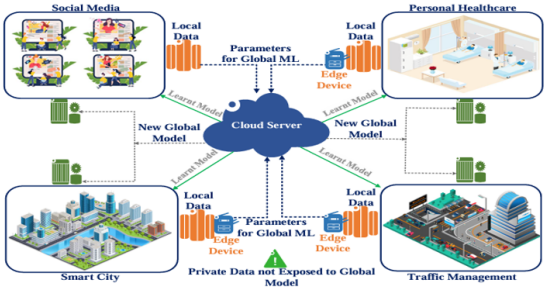

Figure 1: IoT Based Applications

Figure 2: Health Data Transfer

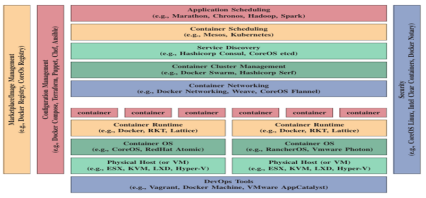

Figure 3: Containerâs stack and realisation Technologies

strained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT), Hyper Text Transfer Protocol Secure (HTTPS), and Lightweight Machine-to Machine (LWM2M) are set of rules and format that govern how data is transmitted and received between devices, networks, and systems [4]. Modern distributed systems rely heavily on communication protocols for both design and operation. They facilitate smooth coordination and communication by defining the norms and guidelines for message exchange between various components. Communication protocols must be secured to protect sensitive data and also prevent cyber attacks such as data breaches, unauthorized access, and Denial-of-Service (DOS). Secured communication protocol ensures maintenance of device integrity. As a result, this paper presents the concept of containerization as a framework towards securing IoT devices in mobile health. The rest of the paper is organized as follows: Section 2.0 discusses containerisation and IoT communication Protocol. Section 3.0 discusses the types of containerization platforms. Section 4.0 presents the framework for the implementation of containerization of communication protocols in IoMT devices. Section 5.0 presents the Step by step approach to implement containerisation of communication protocol in IoMT devices. Finally, concluding remarks are provided in section 6.0.

2 Containerization and IoT Communication Protocol

This section describes containerization and how it is applied to IoT platforms and scenarios. Containerization is a software deployment process in which all the components of an application are bundled into a single container image and can be run in isolated user space on the same shared operating system [3]. The process bundles an applicationâs code with all the files and libraries it needs to run on any infrastructure Containers enable developers to define and build their software environments and then run them on top of various resources in a portable, reproducible way. Containerization of communication protocol provides a secure environment for

communication protocols, protecting them from unauthorized access and malicious attacks. It enables easy deployment and management of communication protocols across different environments and devices. It allows efficient scaling of communication protocol to meet the demands of large-scale IoT deployments and simplifies the maintenance and updating of communication protocols, reducing the risk of errors and downtime. Containers are being used for different applications such as IoT services, smart cars, fog computing, and service meshes [15] Figure 3 shows a container as a building block for a larger technology stack that are used to facilitate microservice deployment. Developers have used containers to define and build their software environment, and then run them on various IoT resources in a portable, secured and reproducible way [3]. The deployment allows the development of a distributed platform as a set of independent platforms that work together. It increases the reliability of communication and data transfer when applied to IoT scenarios [9].A modular and scalable architecture based on lightweight for IoT scenarios was proposed by [19]. Each component of the architecture has embedded docker, with the application divided in small services, and implemented inside containers. Reliability was achieved through this application. [12] proposed an IoT platform based on the microservice models. This with the aim of making inherent application to be distributed, secure, and have support for heterogeneity. This platform is leveraged on a SAVI cloud, a two-layer academic cloud, including a core in Toronto and seven smart edges across Canada.

3 Containerization Platforms

These are software platforms that enable the creation, deployment, and management of containers in communication scenario in IoT context. These platforms improve security of communication protocols through isolation and sandboxing, simplified deployment and management, increased efficiency in communication and support for a wide range of container runtimes.

3.1 Docker

Docker is a lightweight container-based virtualization platform that enables developers to isolate applications and their dependencies, reducing the attack surface and enhancing security. It is a popular containerization platform for different operating systems (Linux and Windows) that enables developers to run applications in containers [7]. By running each application in its container, developers can limit the potential impact of security vulnerabilities and maintain better control over the

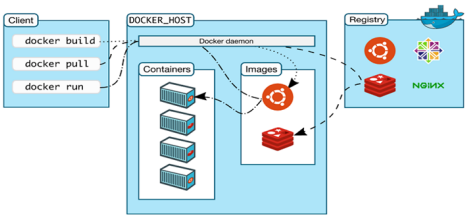

Figure 4: The Docker System

application's environment. Figure 4 shows the docker system consisting of the Docker server and daemon, images, registries, and containers.Docker container provides methods for security level, by using namespaces, and Cgroups mechanisms, to achieve process hardware, and isolation mechanisms. Docker allows an easy way for running and managing containers among users and data centers. [13] perform a measurement study on Linux container security using real exploits that can break the isolation launched to attack containers, by proposing a defense mechanism to defeat all identified privilege escalation attacks. [20] enhanced container security by providing a secure container mechanism to protect the container processes from outside attacks using the SGX trusted execution support of Intel CPUs. Core tools of Docker container technology includes Docker Engine which processes Dockerfile manifests or runs pre-built container images, Docker Registry that stores and provides access to numerous public and private images that are intended for deployment within Docker Engine; Docker Compose which helps assemble applications consisting of multiple components with all the required configurations declared in a single compose file, and Docker Swarm which represents several independent Docker nodes interconnected into a cluster.

3.2 Kubernetes

Kubernetes, often called K8s, is an open-source platform for orchestrating containers, first made by Google. Itâs built to automate deploying, scaling, and managing containerized applications. Kubernetes gives a strong and flexible setup for handling containers in complex setups. It is a container orchestration platform that automates the deployment, scaling, and management of containers as shown in the architecture in Figure 5. The features of Kubernetes includes Scalability which let applications to automatically add or remove instances (containers) based on changing load. This ensures applications are available and perform well; Load balancing which spreads traffic across containers or pods. This evens out requests and ensures fault tolerance.

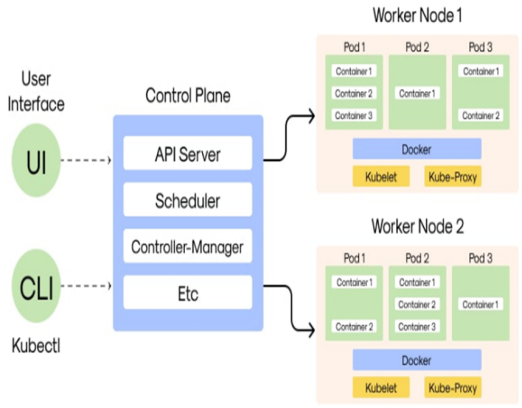

Figure 5: Architecture of Kubernetes

3.3 Rancher

Rancher is a complete software stack for teams adopting containers. It addresses the operational and security challenges of managing multiple Kubernetes clusters, while providing DevOps teams with integrated tools for running containerized workloads. It is a container management platform that enables developers to deploy, manage, and orchestrate containers in a multi-cloud environment. Under Rancher's network, a container will be assigned both a Docker bridge IP (172.17. 0.0/16) and a Rancher managed IP (10.42.0.0/16) on the default docker 0 bridge. Containers within the same environment are then routable and reachable via the managed network. Rancher infrastructure services include networking, storage, load balancer, DNS, and security. Rancher infrastructure services are typically deployed as containers themselves, so that the same Rancher infrastructure service can run on any Linux hosts from any cloud. Rancher Desktop offers greater flexibility and power.

3.4 Red Hat Open Shift

It is a container application platform that enables developers to deploy and manage containers in a hybrid environment. It has built in security features in Red Hat for risk analysis and security protection. These features protect the cluster infrastructure and network communication, isolate computer resources, and ensure security compliance across the infrastructure components and container deployments as shown in the infrastructure. The architecture is shown in Figure 6.

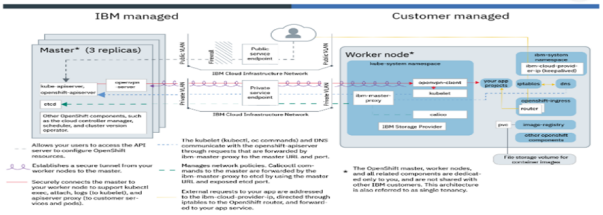

Figure 6: Architecture of Red Hat Open Shift

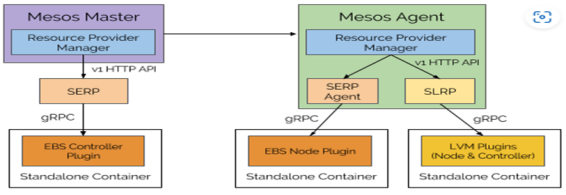

Figure 7: Architecture of Apache Mesos

3.5 Apache Mesos

This platform enables developers to manage and orchestrate containers in a distributed environment. Apache Mesos is a powerful cluster manager that offers effective resource isolation and sharing across distributed application. Figure 7 shows the architecture of how containerisation is supported in Apache Mesos. Mesos simplifies the complexities of running tasks in a shared pool of servers, ultimately rendering more efficient and scalable system operations. Apache Mesos is a powerful cluster manager that provides efficient resource isolation and sharing across distributed applications or frameworks. It acts as the âkernelâ for your data center, abstracting CPU, memory, storage, and other compute resources away from machines (physical or virtual). Mesos allows for the dynamic sharing of these resources between applications, which can significantly improve the scalability and efficiency of large-scale systems. Mesos runs on Linux (64 Bit) and Mac OS X (64 Bit). Apache Mesos provides several security features, such as SSL-based communication, Container Image verification for Docker and application images, and Access Control Lists (ACLs) for controlling access to Mesos APIs. Apache Mesos is known for its outstanding performance with the ability to scale and manage tens of thousands of nodes efficiently. It also minimizes the resource footprint through containerization.

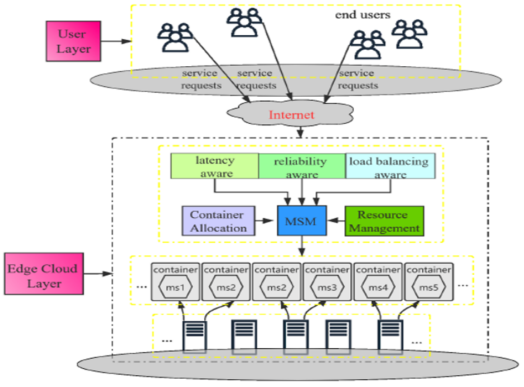

Figure 8: Framework for the Containerized Communication Protocol in IoT Context

4 Framework for the Implementation of Containerization of Communication Protocols in IoT Devices

This involves packaging communication protocols into containers, which can be deployed and managed independently as shown in the framework in Figure 8. The user layer plays a key role in the implementation framework by providing an interface for users to interact with the containerized communication protocol. This layer provides security and access control mechanism. A request initiated at the user layer triggers the service request to initiate communication between the user layer and the containerized communication protocol. Latency aware ensures that the most suitable protocol is chosen for a specific application, reliability-aware mechanism enables error detection and correction, ensures that communication protocol continues to function even in the event of any failure or error. Container allocation enables containers to be allocated and deallocated dynamically, Memory and Storage Management (MSM) ensures that containers use the minimum amount of memory required, while resource management enables improved scalability and provides a dedicated environment for each container. The cloud edge layer is the infrastructure and services that enable cloud computing capabilities at the edge of the network. It enables real-time processing and analytics, leveraging containerization to deploy and manage applications. Latency aware ensures that the most suitable protocol is chosen for a specific application, reliability-aware The containers provide isolation and security for the communication protocol, ensuring that each runs independently and securely. Multiple communication protocols can be enabled to run on the same host without conflicting each other.

5 Step by step approach to implement containerisation of communication protocol in IoMT devices

This section discusses the steps to implementing the containerisation of communication protocol in IoMT devices.

1. **Identification of the type of Communication Protocol**: Communication Protocols are the machine to machine communication language that ensure smooth communication between different IoT devices. There are different types such as MQTT, CoAP and HTTP. Protocol type, protocol version, and protocol configuration such as encryption and authentication determines the containerization requirement. Different protocols have varying requirement and dependencies, distinct security feature and vulnerabilities, and specific configuration to function effectively and efficiently. All these will ensure optimal performance optimization within the containerized environment.

2. **Selection of suitable containerization platform**: Platform compatibility and security features are two key selection criteria. The containerization platform must be compatible with the IoMT deviceâs operating system, deviceâs hardware architecture, and the communication protocols and software required by the IoMT device. The selected platform must provide data encryption, access control mechanism for the communication protocol, and support secure communication protocols such as TLS or IPsec.

3. **Containerisation of communication protocol**: This stage involves creating a container image using the selected communication protocol and its dependencies. For example a Dockerfile or a Kubernestes YAML are used to define the container image. In the case of HTTP/HTTPS protocol, the Docker will be used to containerize an HTTP/HTTPS server, such as Apache or Nginx. After this, a Dockerfile will be created that installs the server software and configures it to run on a specific port. Finally, A Docker image will be built from the Docker file and run as a container. Kubernestes is used in the case of MQTT protocol to containerize and MQTT broker, such as Mosquitto. A Kuberneste deployment YAML file that defines the container and its configuration is created, this file is finally used to create the deployment and run the container.

4. **Configuring the container network and security**: At this stage, the container is configured to optimize its security and performance. This stage also involves setting up environment variables, configuring network settings, and installation of dependencies.

5. **Deployment of the Container**: This can be achieved in IoT devices using a container orchestra-

tion tool like Kubernestes or Docker Swarm. After this stage, the containerâs performance and security are monitored using tools like Promenthus and Grafana. The containerized communication protocol is deployed and monitored in the desired environment.

6 Conclusion

Containerisation provides a high level of isolation between communication protocols, reducing the risk of a security breach. It enables secure communication between IoT devices and the cloud or other devices. Containerisation makes it easier to manage vulnerabilities in communication protocols.

References

[1] Adibi. Mobile health: A technology road map. 2015.

[2] Anusha, S. et al. Performance analysis of data protocols of internet of things: A qualitative review. *International Journal of Pure and Applied Mathematics*, 115(6):37–47, 2017.

[3] Bentaleb, B. et al. Containerization technologies: Taxonomies, applications and challenges. *The Journal of. Supercomputing*, 78(1):1144–1181, 2021.

[4] Bormann, C. et al. Coap: An application protocol for billions of tiny internet node. *IEEE Internet Computing*, 1(2):62–67, 2012.

[5] Carvalho Barbosa, R., Shoaib Ayub, M., Lopes Rosa, R., Zegarra Rodríguez, D., and Wuttisittikulkij, L. Lightweight PVIDNet: A priority vehicles detection network model based on deep learning for intelligent traffic lights. *Sensors*, 20(21):6218, 2020.

[6] de Sousa, A. L., OKey, O. D., Rosa, R. L., Saadi, M., and Rodriguez, D. Z. A novel resource allocation in software-defined networks for iot application. In *2023 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–5. IEEE, 2023.

[7] Docker. [online] available april 2020: https://docs.docker.com.

[8] dos Santos, M. R., Batista, A. P., Rosa, R. L., Saadi, M., Melgarejo, D. C., and Rodríguez, D. Z. Asqm: Audio streaming quality metric based on network impairments and user preferences. *IEEE Transactions on Consumer Electronics*, 2023.

[9] Fazio, C. et al. Open issues in scheduling microservices in the cloud. *IEEE Cloud Computing*, 3(5):81–88, 2016.

[10] Garcia-Constantino, K. et al. Ambient and wearable sensor fusion for abnormal behaviour detection in activities of daily living. *IEEE Access*, pages 1–6, 2020.

[11] Ji, L. et al. Differential privacy and machine learning: A survey and review. 2014.

[12] Khazei, B. et al. A self-managing containerized iot platform. In *Proceedings of the IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, 21 - 23 August*, 2017.

[13] Li, J. et al. Application research of docker based on mesos application container cluster. In *International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 476 – 479, 2020.

[14] Marjani, N. et al. Yahoo,big iot data analytics: Architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261, 2017.

[15] Morabito, F. et al. Evaluating performance of containerized iot services for clustered devices at the network edge. *IEEE Internet of Things Journal*, 4(4):1019–1030, 2017.

[16] Mourtzis, V. et al. Industrial big data as a result of iot adoption in manufacturing. *Procedia CIRP 55*, pages 290–295, 2016.

[17] Qadri, N. et al. The future of healthcare internet of things: a survey of emerging technologies. *IEEE Commun Surv Tutorial*, pages 1121–1167, 2020.

[18] Rosa, R. L., Rodriguez, D. Z., and Bressan, G. Sentimeter-br: A social web analysis tool to discover consumers' sentiment. In *2013 IEEE 14th international conference on mobile data management*, volume 2, pages 122–124. IEEE, 2013.

[19] Rufino, A. et al. Orchestration of containerized microservices for iiot using docker. In *Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT), Toronto, ON, Canada, 22-25 March*, 2017.

[20] Sergei, B. et al. Scone: secure linux containers with intel sgx. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16). USENIX Association, USA,*, 2016.

[21] Teodoro, A. A., Silva, D. H., Rosa, R. L., Saadi, M., Wuttisittikulkij, L., Mumtaz, R. A., and Rodriguez, D. Z. A skin cancer classification approach using gan and roi-based attention mechanism. *Journal of Signal Processing Systems*, 95(2-3):211–224, 2023.

[22] Van-Zoonen, L. Privacy concerns in smart cities, government information quarterly. pages 472–480, 2016.

[23] Vieira, S. T., Rosa, R. L., and Rodríguez, D. Z. A speech quality classifier based on tree-cnn algorithm that considers network degradations. *Journal of Communications Software and Systems*, 16(2):180–187, 2020.