University of Padova

Master Degree in ICT for Internet and Multimedia

Analysis of security at the Near-Real-Time RIC xApps based on O-RAN-defined use cases

Zeinab Shahbazi
Department of Information Engineering

Supervised by
Dr. Alessandro Brighente, University of Padova
Co-supervisors
Dr. Michele Polese, Northeastern University
Prof. Mauro Conti, University of Padova

July 18th, 2022

To:

My dear parents, for their undying support,
My beloved sisters and brother, who have been my source of inspiration and gave me strength when I thought of giving up, who continually provide their moral, spiritual, and emotional support,

And, thank you to my academic advisers, who guided me in this process.

Abstract

The Open Radio Access Network Alliance (O-RAN Alliance) is a group of industry and academic organizations that strive to realize the vision of next-generation cellular networks. Using standardized interfaces, telecommunications operators can operate multi-vendor infrastructure and deliver high-speed services to their mobile users. Additionally, the O-RAN Alliance has standardized an Open Radio Access Network (RAN) architecture based on the Third Generation Partnership Project (3GPP) and other standards. User planes and control planes are currently separate in RAN architecture. The separation makes it easier to accommodate network function virtualization methods required for 5G, enabling it to be more flexible. To help in the management of resources, the O-RAN standard proposes the use of xApps, i.e., dedicated applications that can be customly installed by the network operatior and that can be purchased from different vendors. For this reason, securely managing xApps represents a significant challenge for the security of the overall network.

In this thesis, we analyze the security of xApps and their proposed use cases. Based on the applications porposed by the O-RAN alliance, we provide an in depth analysis of the vulnerabilities and their impact on the network. We also discuss different features of attacks, such as reproducibility, stealthiness, exposure, and impact. Based on our analysis, we conclude that significant work is still to be made to guarantee the security of O-RAN and in particular of its xApps. This thesis hence provides a baseline for future research in the domain of security and privacy for next generation communication networks.

Contents

1 Chapter 1

1.1 Introduction

The architecture of the next 5G and 6G networks is changing from an inflexible, monolithic system to one that is flexible, agile, and disaggregated to accommodate service heterogeneity, coordination across various technologies, and quick on-demand deployments [1]. The upcoming Open Radio Access Network (O-RAN) framework makes these changes possible. Open RAN has attracted a large number of O-RAN groups. Among these groups, the Telecommunication Infra Project (TIP) is one of the first groups focusing on this area. It formed in 2016 to provide everyone with worldwide access, hosted by Vodafone. The other group is the O-RAN Alliance, formed to support open and intelligent Radio Access Network (RAN) interfaces. It started working in 2018 and consists of two groups C-RAN which includes Chinese vendors, and the XRAN Forum, which includes US, European, Japanese, and South Korean vendors [2]. O-RAN Alliance introduces nine working groups that explain different components and architecture in O-RAN, followed by nineteen use cases (UC) that follow its architecture. O-RAN proposes component-neutral disaggregation of RAN at both the hardware and software levels. This technology combines hardware- and software-defined functions for radio access parts of a network, increasing reliability and availability using modular software and cloud-scale economics [3]. O-RAN Alliance encloses two primary aims. The first is openness, which could promote service agility and cloud-scale economics allow smaller vendors and operators to launch their services or adapt the network to their needs. The second aim is to automate these more intricate networks to make operation and maintenance straightforward, leading to lower OPEX. It makes it possible by integrating intelligence utilizing cutting-edge deep learning techniques into each layer of the RAN architecture, both at the component and network levels. Combining AI-optimized closed-loop automation with standardized southbound interfaces will usher in a revolutionary era for network operations [4].
Despite the popularity of the O-RAN architecture among researchers, it is hard to implement open programmable and virtualized next-generation networks. There are still vital architectural questions to be answered, including functionalities and parameters that control by each component accessing data and analytics from the RAN while minimizing the overhead of moving them from the RAN to storage and inference locations [5].

O-RAN encloses three primary terms, Open RAN, O-RAN, and VRAN. Open RAN is an industry that focuses on open radio access network architecture. VRAN, on the other hand, refers to the virtualization of RAN functions, such as upper and lower layer functions of the baseband unit. O-RAN is the standard in O-RAN Alliance, referring to four missions, including open interfaces, virtualization, intelligence, and interoperability [6].
The O-RAN architecture consists of two sections, the Non-Real-Time Intelligent Controller (RIC) and the Near-Real-Time RIC. The former component's responsibility is to apply Machine Learning (ML) and Artificial Intelligence (AI),

train them and then send them to Near-Real-Time RIC using the A1 interface. Near-Real-Time RIC hosts xApps that control and optimize RAN functions and resources. It makes the xApp a critical component and increases the importance of the security inside it.
Furthermore, the number of machines, objects, and devices using O-RAN architecture is increasing rapidly, allowing third-party software developers to freely build xApps for the RIC to set up new services and innovate. Therefore, operators can offer tailored solutions, allowing their adaptation to specific applications or industries [7].

In this thesis, we assume that a malicious user gains remote access to the network by backdoor injection to control the other xApps. After gaining access to the xApp, an attacker can apply different attacks. These potential attacks explain in terms of their level of criteria. For this aim, we utilize use cases introduced by O-RAN Alliance working groups and analyze the amount of effectiveness of the attacks on each UC.

1.2 Research Goal and Scope

During the last few decades, the number of network operators has increased due to the dependency of people life on the wireless network. Although it has numerous benefits for society and industry, it leads the attackers to find a way to access the user's data and cause problems. These issues range from disrupting devices for a short time to stealing data. Consequently, devices become inaccessible. Security Focus Group (SFG) is collaborating with O-RAN working groups, and its goal is to provide information about the security of O-RAN components [8]. The other research in the area of O-RAN that done before is merely relying on analyzing the security inside the new interfaces introduced by O-RAN architecture. The xApp is the most crucial component inside the O-RAN, as its responsibility is to control RAN components. Making decisions according to the data received from Non-Real-Time RIC and sending them to the specific UE is the xApp goal. There will be an effect on all network functionality if xApp cannot generate the appropriate messages for the user. It increases the importance of finding the ways attackers may use to find vulnerability inside the xApp and apply their attack. The main focus of this thesis is analyzing the potential attacks inside each use case introduced by O-RAN Alliance which relies on xApp.

1.3 Thesis Structure

The thesis is organizing as follows:

- Chapter 2 discusses the fundamental concepts and architecture behind O-RAN and xApp,
- Chapter 3 explains the use cases and their architecture,
- Chapter 4 introduces the attack inside the Kubernetes and xApp,

- Chapter 5 present analyzes the use cases and their impact on the specific use cases,
- Chapter 6 presents a discussion of the thesis,
- Chapter 7 introduces the strategy,
- Chapter 8 shows related works,
- Chapter 9 concludes the thesis.

2 Chapter 2 - O-RAN

This chapter begins with an overview of the O-RAN architecture and its components. Then we give some details on Non-Real-Time RIC, followed by a detailed overview of the Near-Real-Time RIC components.

2.1 Background

Figure 1 showes the O-RAN architecture. It consists of Non-Real-Time RIC and Near-Real-Time RIC. The Service Management and Orchestration (SMO) includes Non-Real-Time RIC and is responsible for ML/AI, which is in charge of training the enrichment data. ML algorithms trained in the Non-Real-Time RIC are sent to the Near-Real-Time using the A1 interface. We briefly explain the other components and interfaces used in O-RAN.

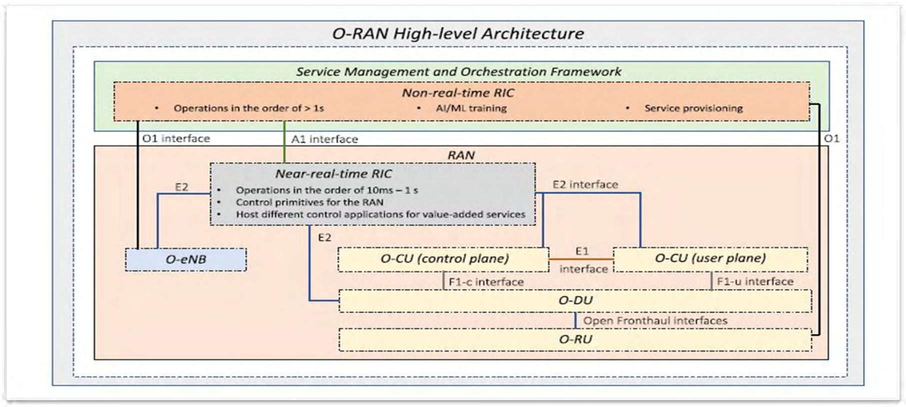

O-RAN High-level Architecture
Service Management and Orchestration Framework
Non-real-time RIC
Operations in the order of > 1s
AI/ML training
Service provisioning
O1 interface
A1 interface
O1
RAN
Near-real-time RIC
Operations in the order of 10ms – 1 s
Control primitives for the RAN
Host different control applications for value-added services
E2
E2 interface
E2
O-eNB
O-CU (control plane)
E1 interface
O-CU (user plane)
F1-c interface
F1-u interface
O-DU
Open Fronthaul interfaces
O-RU

Figure 1: O-RAN Architecture from [9]

2.2 E2 Nodes

E2 nodes are logical functions that support all the protocol layers and interfaces described by the Third Generation Partnership Project (3GPP) RAN. A single Near-Real-Time RIC can connect to one or more E2 nodes through transport functions, but each E2 node can only connect to one Near-Real-Time RIC. The E2 interface collects Near-Real-Time data from E2 nodes and performs fine-grained Radio Resource Management (RRM) activities over E2 nodes via the E2 interface [10].

2.3 Central Unit (CU)

The CU is run in the higher layer and hosts Radio Resource Control (RRC), Service Data Adaptation Protocol (SDAP), and Packet Data Convergence Protocol (PDCP) protocols. As Figure 1 shows, there are two different CU, the control plane and the user plane. The first one hosts RRC and PDCP protocol and uses it for the control plane. The user plane supports the PDCP protocol and the SDAP protocol. E1 Interface is in charge of connecting the user plane and control plane [9].

2.4 Distributed Unit (DU)

DU node hosts Radio Link Control (RLC)/Media Access Control (MAC)/ high-physical layers based on a lower layer functional division. DU is connected to the CU control plane using the F1-C interface and with the CU user plane by F1-U. In addition, an open fronthaul interface connects it to Radio Unit (RU) [11].

2.5 Radio Unit

A radio unit is a logical node that contains a low-physical layer and Radio Frequency (RF) processing based on a functional split at the lower layer [11]. The RU holds transceiver antennas and particular radio hardware that performs physical layer functions such as digital to analog conversion, filtering, and modulation. Furthermore, it is in charge of signal amplification and regeneration [12].

2.6 Non-Real-Time RIC

Non-Real-Time RIC logically terminates the A1 interface and offers Near-Real-Time RIC with policy-based guidance, enrichment data, and AI/ML model management. Moreover, it can access the other SMO using the O1 and O2 interfaces. Non-Real-Time RIC host rApp, i.e., is an application that uses the Non-Real-Time RIC framework and SMO features to deliver value-added services related to RAN operation and optimization [13].

2.7 Near-Real-Time RIC Components

In this section, we explain the different components of Near-Real-Time RIC. An illustration of the Near-Real-Time RIC architecture can see in Figure 2.

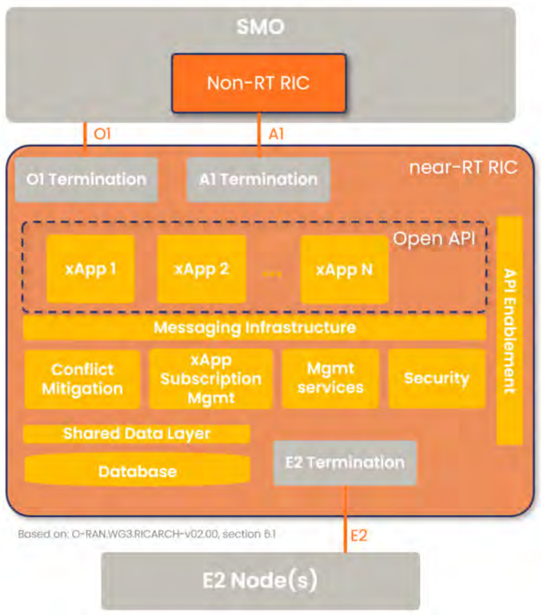


Figure 2: Near-Real-Time RIC Architecture from [14]

2.7.1 xApp subscription management

Subscription management follows some goals of managing subscriptions from xApp to E2 nodes. It ensures that policies affecting xApp messages are authorized and allows a combination of identical subscriptions from various xApps into a single E2 node subscription [14].

2.7.2 Database together with shared data layer

Database and shared data layer allow reading and writing of RAN and UE information. As Figure 3 shows database in Near-Real-Time RIC consists of two parts with different capabilities. Radio-Network Information Base (R-NIB) uses to store the data and configuration coming from Near-Real-Time RIC. The other part, User Equipment NIB (UE-NIB), manages the list of various UEs and their data in addition to their identity [14].

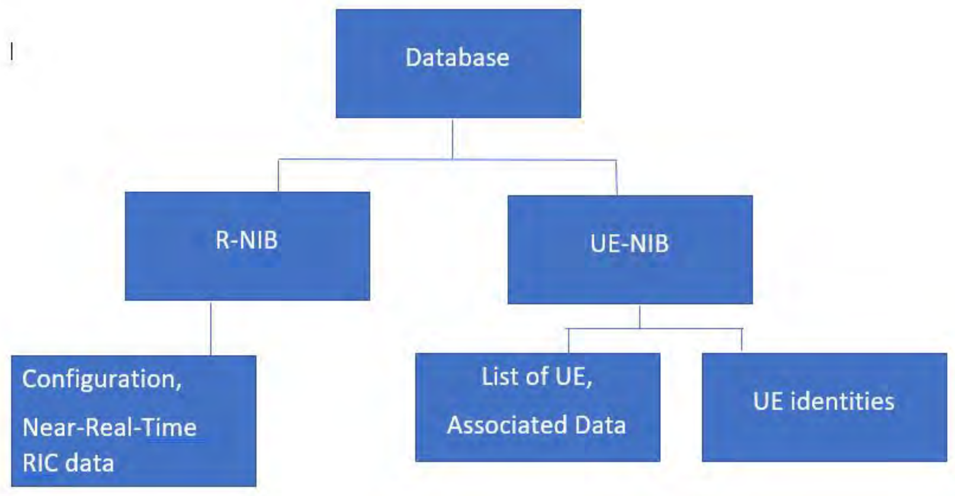


Figure 3: Database

2.7.3 Conflict mitigation

Conflict is trying to solve the mitigation between different xApp in the context of Near-Real-Time RIC.

Conflict mitigation can divide into three different parts:

Direct Conflicts: In this type of mitigation, conflict depicts by conflict mitigation. In this case, three conditions may happen:

- The first one occurs when for the same configuration of one or more parameters of a control target, two or more xApps request different settings so that conflict is analyzed and make a decision.
- In the second one, running configuration resulting from a prior request from another or the same xApp may conflict with the current request from a xApp.
- The last mitigation in direct conflict is the total requested resources from separate xApps may exceed the RAN system's resource limitation. Pre-action cooperation may usually reduce direct conflicts.

Indirect conflict: Although the conflicts cannot observe, some dependency in parameters between xApps can catch. In this mitigation, the system must make decisions based on the observations, such as rolling back one of the xApp operations.

Implicit conflicts: Conflicts cannot see directly, and dependencies between xApps are invisible. Implicit conflicts are the most difficult mitigation to resolve because these dependencies are difficult or impossible to observe, making any mitigation scheme difficult to model [14].

2.7.4 Messaging infrastructure function

The other component in Near-Real-Time RIC is the messaging infrastructure which provides a low-latency message delivery service between internal Near-Real-Time RIC endpoints. Endpoints are physical devices communicating with the network.
Table 1 indicates the different endpoints inside messaging infrastructure, followed by their impact. The first endpoint is registration, meaning that endpoints register themselves to the messaging infrastructure. Discovery is the other endpoint in messaging infrastructure. Endpoints are typically detected by messaging infrastructure. The last one is the deletion endpoint, meaning that messaging infrastructure removes it if that is of no use anymore. In addition, message infrastructure can provide two APIs for sending and receiving a message. The other characteristics of message infrastructure can refer to supporting multiple message modes, message routing, and message robustness to avoid data loss [14].

Table 1: Messaging Infrastructure Endpoint

Endpoint	**Impact**
Registration	Register endpoint
Discovery	Detect endpoint by message infrastructure
Deletion	Remove useless endpoint

2.7.5 Security function

The main goal of the security function is to prevent malicious xApps from exploiting radio network information, exporting it to unauthorized remote computers, and controlling RAN functionality [14].

2.7.6 Management services function

It controls the life-cycle management of xApp and Fault, Configuration, Accounting, Performance, and Security (FCAPS) management of Near-Real-Time RIC. Life-cycle management provides onboarding xApps, deploying, resource management, and termination of xApps. In addition, it is in charge of the collection transmission of logging, tracing, and metrics to an external system for analysis.

2.8 Near-Real-Time RIC Interfaces

This section provides information about the new interfaces introduced in O-RAN.

2.8.1 E2

The E2 interface acts as a bridge between the E2 node and the Near-Real-Time RIC. In addition, it can send data from E2 nodes to the Near-Real-Time RIC. This data includes supported slices, cell configuration, and supported slices. With this information, Near-Real-Time RIC can control functions inside E2 Node. Figure 4 shows the E2 functions [15]. These functions are Near-Real-Time RIC service and Near-Real-Time RIC support. The first function explains about report, control, insert, and policy. The support function includes interface management and service updates. Interface management indicates information such as E2 configuration, E2 setup, and reports of possible errors. Service update informs Near-Real-Time RIC about supported lists and service-to-function mapping.

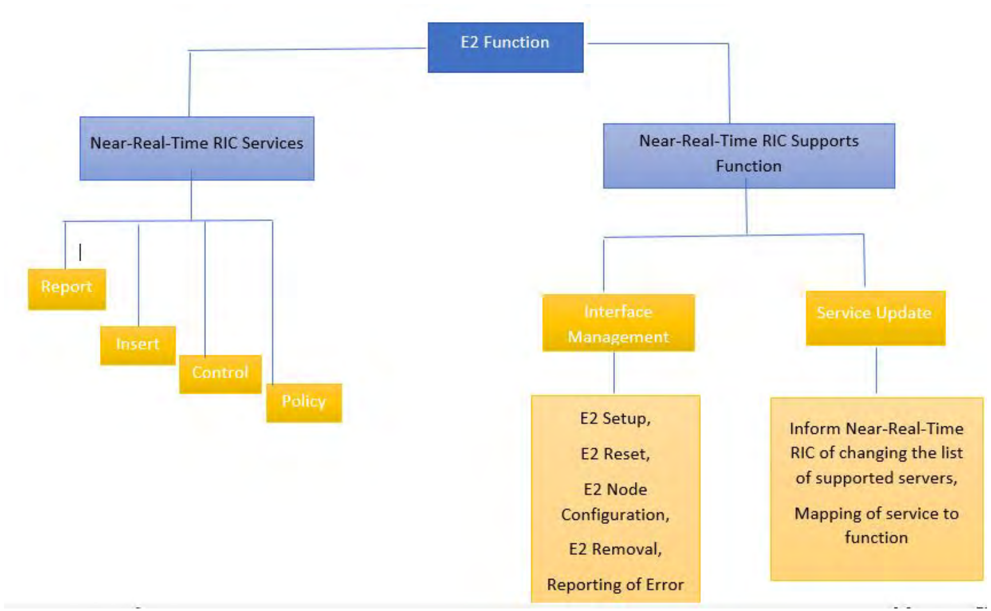

E2 Function
Near-Real-Time RIC Services
Near-Real-Time RIC Supports Function
Report
Insert
Control
Policy
Interface Management
Service Update
E2 Setup,
E2 Reset,
E2 Node Configuration,
E2 Removal,
Reporting of Error
Inform Near-Real-Time RIC of changing the list of supported servers,
Mapping of service to function

Figure 4: E2 Function

2.8.2 A1

The main functionality of the A1 interface is the connection between Non-Real-Time RIC and Near-Real-Time RIC. The SMO hosts the Non-Real-Time RIC function. These functions include AI/ML and ML model deployment. Moreover, it can have access to the data outside the RAN to improve the control and optimization functions of the RAN. The A1 interface's goal is to allow the Non-Real-Time RIC function to offer policy-based advice, ML model administration, and enrichment data to the Near-Real-Time RIC function [16].

2.8.3 O1

The O1 interface uses in operation and administration between management entities in the SMO framework, and O-RAN controlled elements, FCAPS management, software management, and file management [17]. The O1 interface is a bridge between Non-Real-Time RIC and Near-Real-Time RIC, exchanging information and data. For AI model training, the SMO gets data from the managed components via the O1 interface [18].

2.9 xApp

The xApp is an application that is created specifically for the Near-Real-Time RIC. Such an application is likely to contain one or more micro services and specifies which data it consumes and gives throughout the onboarding process. The application is not limited to the Near-Real-Time RIC and may deliver by any third party. The E2 allows for a direct link between the xApp and RAN capabilities [15].

2.9.1 Pod

Kubernetes creates a pod or xApp based on the deployment file. A pod is a kubernetes abstraction that performs a collection of one or more application containers. A pod's containers have the same IP address and port space, are always co-located and co-scheduled, and execute in the same shared context on the same node. On the kubernetes platform, pods are the atomic unit. Each pod belongs to the node on which it is scheduled and stays there until it terminates or deletes [19]. As Figure 5 shows, each pod has a unique IP address for communication. Each pod consists of one or several containers which share the same volume inside the pod.

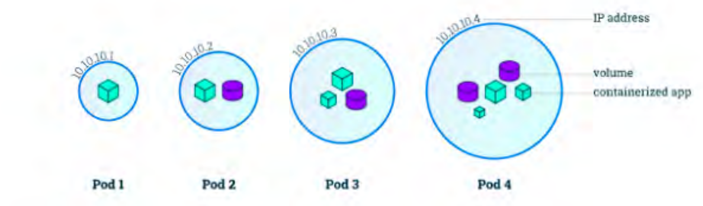


Figure 5: Pod Overview from [20]

2.9.2 Node

As Figure 6 shows, the node contains one or several pods along with at least one kubelet and container run time like docker [21]. The Control plane is in charge of managing the data inside the nodes by communication with kubelet [20]. Docker stands in charge of getting the container image from a registry, unpacking it, and executing the program.

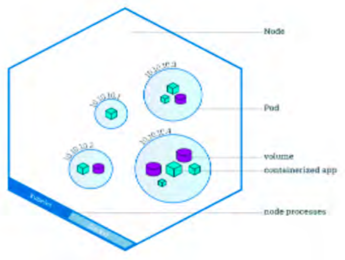


Figure 6: Node Overview from [20]

2.9.3 Master Node

The master node supervises the cluster and acts as the entry point for all administrative activities. It gets operating the Command Line Interface (CLI), Graphical User Interface (GUI), or Application Programming Interface (API). Several master nodes determine in the cluster to achieve fault tolerance, and every time one is active. A master node consists of four components, as Figure 7 shows, that we briefly explain.

API server handles all administrative functions. The API server receives the rest instructions from the user, verifies them, then processes and executes them. The scheduler assigns tasks to various worker nodes. It contains information about each worker node's resource utilization. The scheduler also uses the quality of service need and data location.

The control manager or controller is in charge of non-terminating control loops that regulate the status of the kubernetes cluster. The control manager communicates with the API server to compare desired and actual states. The desired state is the number of pods, while the actual one is the number of real pods inside each node. If the desired does not match the actual state, the control loop takes corrective action to make it equal the desired. As a result, the controller manager ensures that the actual and desired state are identical.

The cluster status saves in the etcd, which is a distributed key-value store [22]. Figure 7 shows the master node along with its components.

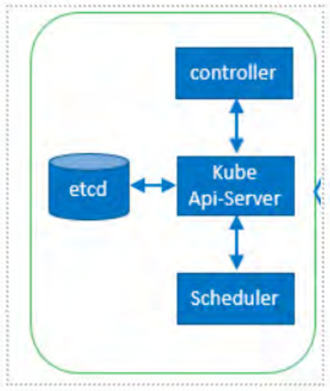


Figure 7: Master Node Overview from [23]

2.9.4 Worker Node

According to Figure 8, every worker node contains one or several pods with a unique kubelet and Kube-proxy. If there are ten worker nodes, kubelet runs on each of them. Kubelet also checks the health of each pod and sends a report to the API server. Kube-proxy keeps track of network rules. These rules allow network sessions inside or outside the cluster to communicate pods [22].

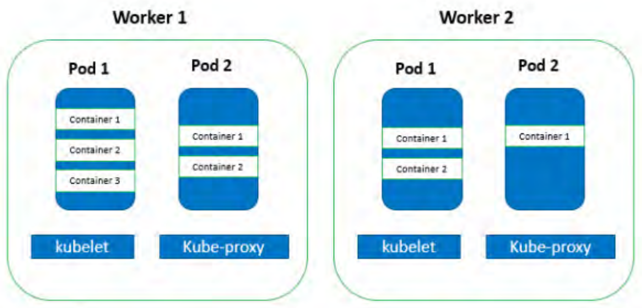


Figure 8: Worker Node Overview from [23]

2.9.5 Creation of Pods

Master and worker nodes communicate to create a pod. Figure 9 briefly explains how these nodes communicate to create a pod. The first step is when a kubelet command sends, goes into the kubernetes cluster, and hits the API-server. The first thing the API-server does is authenticate and validate the command line. The next step is that the API-server writes that pod in the etcd or cluster store. Etcd returns the request when it has made a successful write. At that point, the API-server returns the request to the person who runs the kubelet command. At the same time, the API server sends a configuration to kubelet inside the pod and forces it to create the new one.

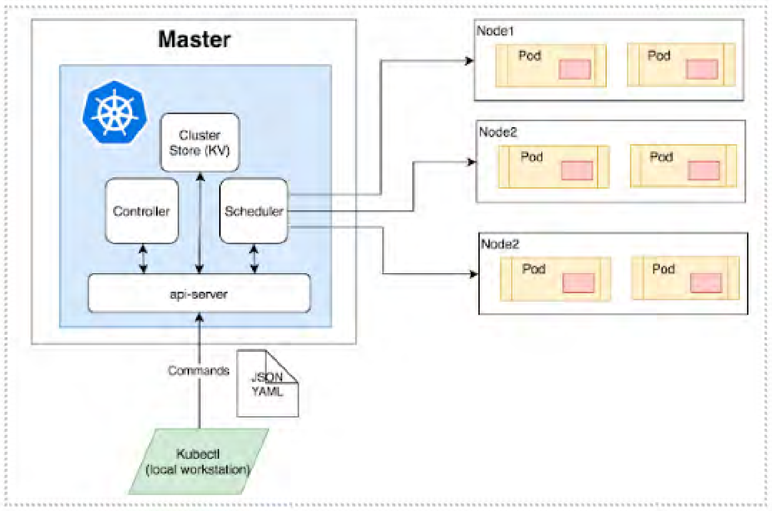


Figure 9: Kubernetes Component

2.9.6 Role-Based Access Control permission (RBAC)

RBAC is a way of controlling access to a computer or network resources based on the responsibilities of individual users in the company. RBAC permission defines the level of access of each user inside the cluster. There are four types of kubernetes objects accessed via the RBAC API. These objects are Role, ClusterRole, RoleBinding, and ClusterRoleBinding [24] and [25].

2.9.7 Role and ClusterRole

The Role and ClusterRole represent a collection of permissions. The role uses to set permissions to access a specific namespace. ClusterRole, on the other hand, defines permissions across several namespaces [25].

2.9.8 RoleBinding and ClusterRoleBinding

A rolebinding assigns a user or group of users the permissions define in a role. It contains a list of users, groups, or service accounts. A rolebinding offers access to a namespace, but a clusterrolebinding grants access to the whole cluster. Once the clusterrole is creating is almost impossible to modify it [25].

2.9.9 Default Roles and Rolebindings

API servers generate clusterrole and clusterrolebinding objects by default. Kubernetes.io/bootstrapping=rbac-defaults is used to name all of the default

clusterroles and clusterrolebinding. Default cluster roles update with any missing permissions, and default clusterrolebindings are updated with any subjects [25]. There are different clusterroles inside the kubernetes. Some of them have a system prefix which refers to the role that the cluster control plane manages the resource directly. The other type of clusterrole is the user role, including:
Cluster-admin, intend to be a super user and have access to and modify all clusters and namespaces,
Admin, access to all workloads pod, deployment replicaset, and permission to set up Role and RoleBinding in namespaces,
Edit, access to all workloads pod and deployment replicaset,
View, having access to reading deployment.

3 Chapter 3 - Use Cases

In this chapter, the UC introduced by O-RAN Alliance discusses. First, we provide some background information about each UC. Second, we focus on the architecture indicated for each UC according to O-RAN working groups.

3.1 Use case 1- Context-Based Dynamic HO Management for V2X

3.1.1 Background Information

The main goal of this UC is to improve radio resources and reduce latency through O-RAN architecture [26]. Several types of V2X introduce in [27] namely, Vehicle-To-Vehicle (V2V), Vehicle-To-Network (V2N), Vehicle-To-Infrastructure (V2I), Vehicle-To-Pedestrian (V2P), Vehicle-To-Self (V2S), and Vehicle-To-Road (V2R). These types of V2X, with communication, can help to increase road safety, reduce emissions, save time, and improve traffic management [28]. It accomplishes by sending a message known as a Basic Safety Message (BSM) and a Cooperative Awareness Message (CAM). To avoid accidents, V2V and V2P exchange information. V2I and V2N contain vehicle connectivity to traffic controllers or servers over networks. Since V2X cannot record all the messages on the road, a new concept, Collective Perception Service (CPS), has been created to use Collective Perception Message (CPM) [28]. CAMs are distributed in the Intelligent Transport Systems (ITS-G5) network, providing information about the presence, positions, and status of communicating ITS stations to neighboring ITS stations. All the ITS stations must send and receive the data from CAMs. By receiving CAMs, the ITS station is aware of other stations in its close area, their locations, movements, properties, and sensor data [29]. Figure 10 shows different types of V2X.

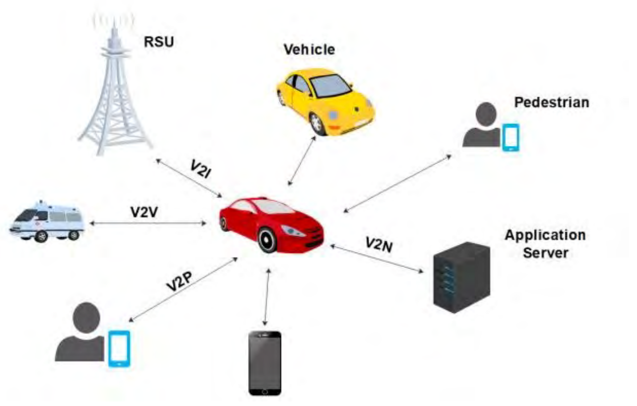


Figure 10: V2X

3.1.2 Architecture

V2X consists of V2X User Equipment (UE), sim and device attached to the vehicle, and V2X Application Server (AS). In this UC, one xApp with two functionality deploys to avoid anomalous HO sequences from degrading the performance of the V2Xs. The first xApp applies long-term analytics using AI/ML algorithms inside Non-Real-Time RIC. The other one is to use a trained ML model on the Near-Real-Time RIC to achieve real-time optimization. V2X AS keeps track of UE events and mobility data and communicates with Non-Real-Time RIC over the O1 interface. Near-Real-Time RIC monitor UE based on the data from V2X AS. The trained model sent from Non-Real-Time RIC to Near-Real-Time RIC can predict the anomaly. As a result, the Near-Real-Time RIC to develop and update UE-specific Near-Real-Time RIC to enhance V2X user performance. Input comes from Xnb and V2X AS. These inputs consist of CAMs, navigation data, geographic data, radio cell ID, connection ID, basic radio measurements, and handover events [26].

Non-Real-Time RIC, Near-Real-Time RIC, RAN, and V2X AS, collaborate in this UC. After an event is detected, Non-Real-Time RIC deploys an ML/AI model and sends it either to Near-Real-Time or xApp by O1 and A1 interface, respectively. The enrichment data from the Non-Real-Time RIC may include V2X UE location, trajectory, navigation information, GPS data, CAMs, and Decentralized Environmental Notification Message (DENMs). These data send to RAN, and it deploys a xApp according to it. This xApp detects and monitors UE and sends an appropriate alarm to it [26].
For this UC, the following architecture suggested.

- Option A involves deploying the Open Central Unit (O-CU), Open Distributed Unit (O-DU), and Near-Real-Time RIC on a vehicle. Non-Real-Time RIC and core network are in the central cloud. Fiber optics link the control vehicle to the core network and Non-Real-Time RIC.

- Option B is a private network in which all modules are in the control vehicle, including the gNB, Near-Real-Time RIC, Non-Real-Time RIC, and the essential core network function modules. Radio resource allocation for Unmanned aerial vehicles (UAV) applications involves some of the O-RAN components' core functionality in both deployment modes. To produce UE-based radio resource allocation rules, Non-Real-Time RIC obtains UE-level radio resource needs from the AS. Near-Real-Time RIC is used in this scenario to assist the execution and interpretation of retrieve policies for configuration parameters for the E2 node. It gives information about UE registration or UE status change to Near-Real-Time RIC [26]. The V2X architecture illustrates in Figure 11.

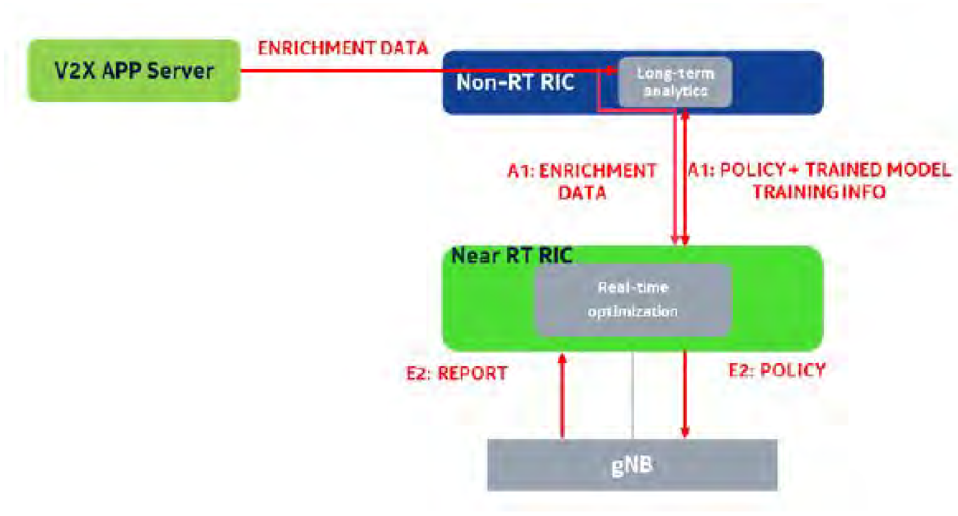


Figure 11: V2X Architecture from [26]

3.2 Use case 2 - Massive MIMO Beamforming Optimization

3.2.1 Background Information

Massive multiple-input and multiple-output (mMIMO) is one of the primary ways to improve 5G network performance and Quality of the System (QoS). For both Single-User (SU) and Multi-User (MU) MIMO, capacity improvement achieves by beamforming transmitted signals and multiplexing data streams [26]. Beamforming is a signal processing approach that uses multiple antenna arrays at the transmitter and receiver to broadcast or detect numerous signals from multiple desired terminals concurrently to boost system capacity and performance [30]. Grid of Beam (GoB) makes beam production of control channels simpler during initial access, especially for high-frequency MIMO. The code book and the GoB determine the beam aperture, the horizontal and vertical apertures in which the beam formation is supported, along with the coverage area and shape of the cell. MIMOs may be employed in 5G large cells, heterogeneous networks, where macro cells and tiny Three-dimensional MIMO (3D MIMO) cells coexist and complement each other to increase storage capacity and coverage [26].

3.2.2 Architecture

Using an appropriate beam pattern is the key to increasing performance in this UC. Moreover, the control of beam failures is a fundamental component of mMIMO. For this reason, AI/ML-based beam mobility optimization techniques prevent connectivity and user experience degradation. The two optimization loops can be created and deployed separately or in a hierarchical configuration [26]. The architecture of Non-real-Time and Near-Real-Time Optimization loops show in Figure 12.

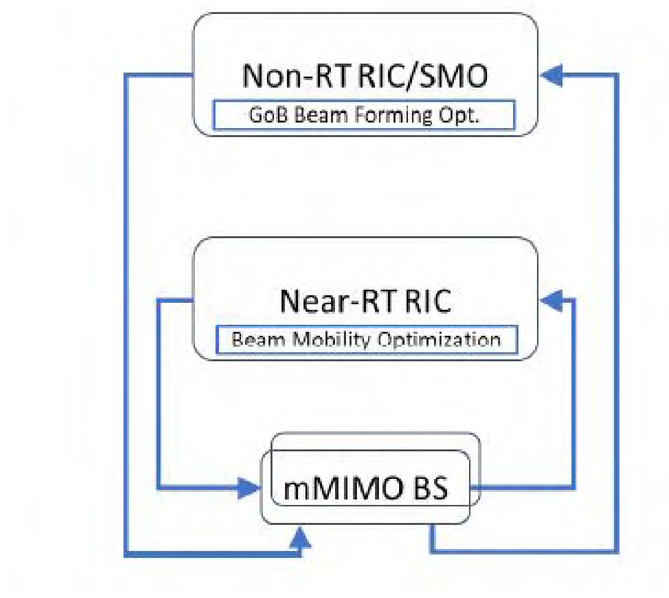


Figure 12: Non-Real-Time and Near-Real-Time Optimization loops from [26]

3.2.3 Non-Real-Time Massive MIMO GoB Beam Forming Optimization

The data send to Non-Real-Time RIC for ML training includes process and evaluation antenna array characteristics, cell performance KPIs, UE mobility, spatial density data, traffic density data, interference data, and minimization of Drive Test (MDT) measurement data. The output of the Non-Real-Time is optimized BF configuration, the number of beams, beam elevation, beam horizontal, vertical widths, and power allocation of beams [26]. Figure 13 indicates the Non-Real-Time BF Optimization architecture.

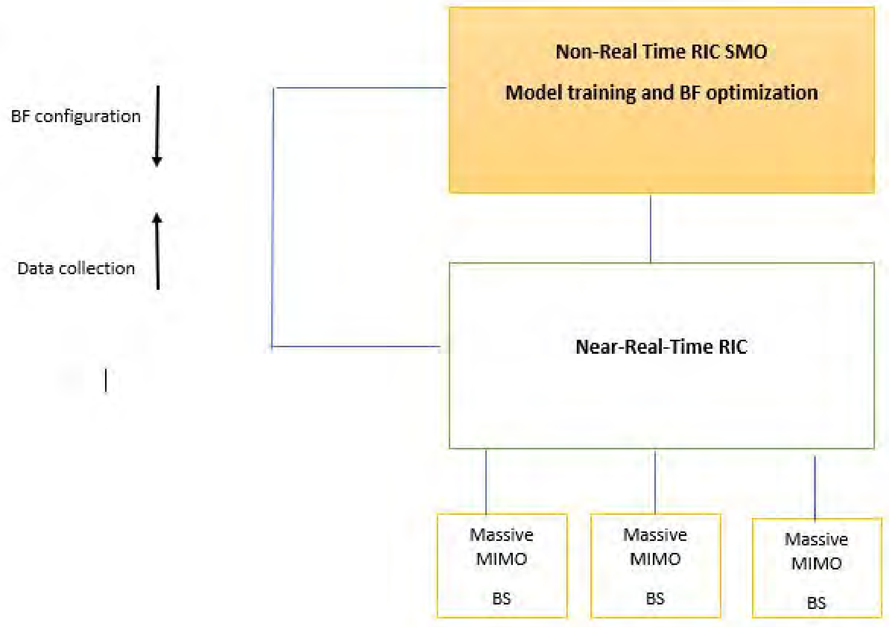


Figure 13: Non-Real-Time Beam Forming (BF) Optimization

3.2.4 Near-Real-Time Massive MIMO BeamBased Mobility Robustness Optimization (bMRO)

The Near-Real-Time RIC contains a xApp with a BF optimization function. It is in charge of monitoring UE/traffic density measurements, UE location information, beam-based mobility and beam failure KPIs, and selecting/deploying optimized mMIMO BF parameters to E2 nodes. In addition, there would be a xApp to optimize inter-cell beam mobility that deploys in a Non-Real-Time BF optimization loop [26]. As can be seen in Figure 14, Non-Real-Time RIC hosts an application for the long-term analytic function to collect and analyze underlying GoB configuration. The Near-Real-Time RIC has a xApp with a Beam Selection Optimization (BSO) function that monitors prospective source-target beam pairings and manages user-beam pairing to improve beam mobility for scheduling. Per-user measurements, beam failure statistics, per-user potential source-target beam pairings, and neighboring cells' beams/interference information may all use as input data for BSO training and inference. For beam mobility, the output of the BSO optimization function uses to alter offsets for possible source-target beam pairings [26].

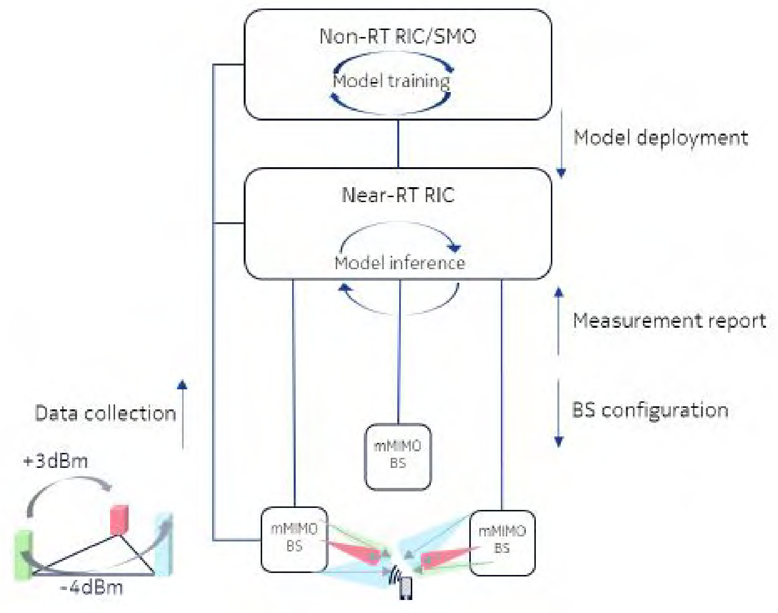


Figure 14: BeamBased Mobility Robustness Optimization (bMRO) from [26]

3.3 Use case 3 - O-RAN signaling storm protection

3.3.1 Background Information

This use case protects the mobility network against signaling storms from the other devices. These days, the number of mobile users increased significantly, leading to a community dependence on the network's connectivity. As a result, the mobility network is vulnerable to unintentional or intentional attacks that might interrupt regular network operations. O-RAN signaling storm protection introduces to predict and avoid the possible attack inside the network [26].

3.3.2 Architecture

In this UC, xApp deploys two different functionalities, detecting and mitigating DDoS. As can be seen in Figure 15, there are two xApps detections deployed in the Non-Real-Time RIC and Near-Real-Time RIC. The reason is some attack needs specific information available inside each of them. However, the others detect without this information. The other difference is that xApp responds to aggressive device detection quicker. The detection algorithm may change based on factors, network environment, dynamic usage trends, and available devices. In addition, depending on attack output would be different. These outputs can consist of a list of specific devices or only point to problematic geographic locations, specific gNBs, or cell towers. The Near-Real-Time RIC's measurement counters and detection depend on evaluating data events per UE connection setup. They include the following data [26].

Some parameters for a primary registration event, as Table 2 shows, can include a time stamp, cell ID, and temporary ID. The other measurement counter is RAN parameters for determining the relationship between a UE and registration events. Tracking the status of ongoing attacks by monitoring statistics of active filters is the third one which contains the number of UE in the filter. The last one is overloaded regions in which data can be above regular.

Table 2: Near-Real-Time RIC's measurement counters

Measurement Counters	**Data**
Basic registration	Timestamp, cell ID, Temporary ID
RAN parameters	RSRP/RSRQ, Timing Advance, Beam ID
Status of ongoing attack	Number of UEs in the filter, number of requests blocked
Overloaded regions, overloaded sites, severity	Percent above normal

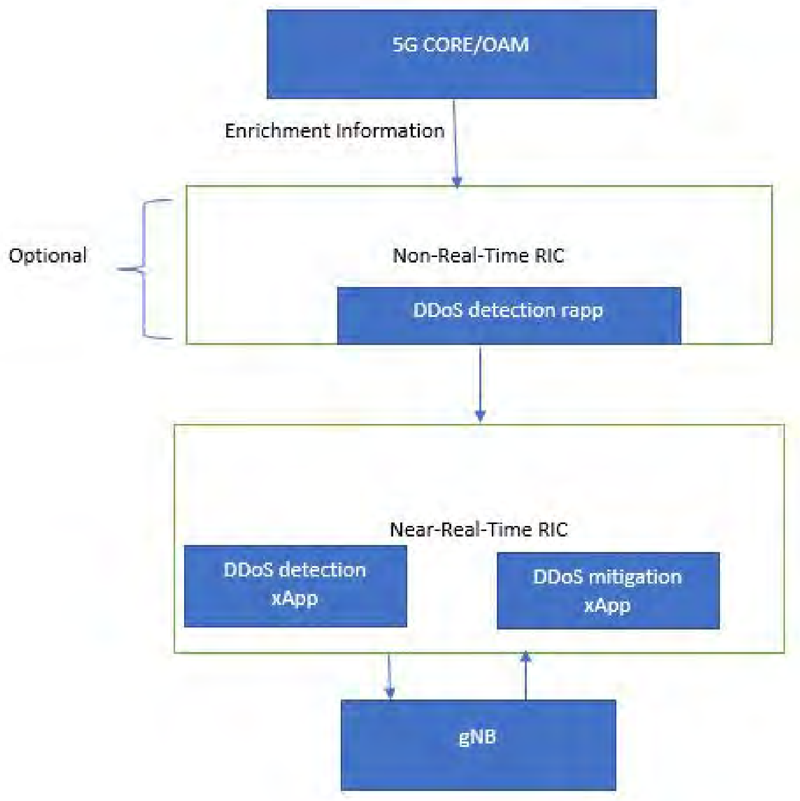


Figure 15: RIC DDoS Capability for 5G Stand Alone Setup

3.4 Use case 4 - Congestion Prediction and Management (CPM)

3.4.1 Background Information

This use case takes a proactive approach to base station congestion management by evaluating radio resource consumption and taking timely corrective action to mitigate any possible system congestion. Network congestion is a critical issue for telecommunication providers since it directly impacts consumers' QoS. The main goal of this use case is to utilize embedded ORAN's intelligence to forecast congestion ahead of time so that operators may have their cell traffic management solution in place before congestion occurs. [26].

3.4.2 Architecture

CPM architecture suggests detecting and mitigating congestion proactively. In the CPM architecture, E2 nodes gather by SMO's data collector. Pre-processing consists of adding cell names/numbers and ids to the data and converting counters to KPIs using KPI logic. Data exchange with Non-Real-Time RIC deployed in the SMO via a data-sharing entity after preprocessing. The training model and application in an AI server invoke by Non-Real-Time RIC. The trained data and the projected KPIs are delivered back to CPM rApp in Non-Real-Time RIC format. The inference form by CPM rApp in Non-Real-Time RIC. After the interface creates, it includes data about the cell. These data include cell ids, information about whether the cell is congested or not, a timestamp of cell congestion, and a predicted KPI value.

There are two ways to mitigate cell congestion. The first way is to transfer data from rApp to xApp through the A1 interface. In this way, Near-Real-Time RIC can decide on a mitigation solution. These solutions could be switching to dual connectivity mode, debarring user access, and load sharing.

The second solution, Non-Real-Time, offers solutions for mitigation consisting of splitting a cell, adding more carriers, switching to higher-order MIMO, and some of the users to Wi-Fi [26].

3.5 Use case 5 - Dynamic Spectrum Sharing (DSS)

3.5.1 Background

DSS is a new antenna technology that allows Long Term Evolution (LTE) and 5G to coexist in the same frequency range. The technology forecasts 5G and LTE demand in real-time. Under the assumption that the two networks coexist, DSS is compelling because it allows operators to dynamically share deployed spectral resources between LTE and New Radio (NR) devices without degrading the Quality of Experience (QoE) of current 4G subscribers. This use case presents DSS as an application in the RIC framework, especially in the context of the ORAN architecture. Over various 4G and 5G spectrum sharing cells, DSS based on RIC may also increase resource management efficiency [26].

3.5.2 Architecture

Figure 16 shows the architecture introduced for the DSS use case. As we explained, both Non-Real-Time RIC and Near-Real-Time RIC collaborate in this UC. DSS App in Non-Real-Time RIC is responsible for providing policy and configuration to 4G and 5G according to criteria such as user, spatial and temporal workload. In addition, the RAT App in Non-Real-Time RIC is in charge of transferring this data to the RAT App in Near-Real-Time RIC. DSS-App in Near-Real-Time RIC translates the data and policy received from Non-Real-Time RIC and sends them to the RAT App in Near-Real-Time RIC. The other purpose of the Near-Real-Time RAT-App is to execute RAT-specific configuration, control, and data subscription via the E2 interface with the RAN [26].

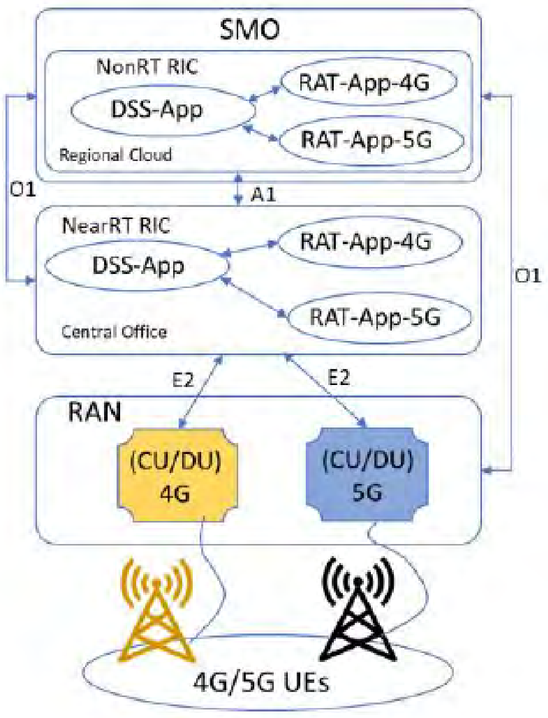


Figure 16: RIC Based DSS Architecture from [26]

3.6 Use case 6 - Integrated SON Function

3.6.1 Background

Self-Organizing Networks (SON) is a set of functions that allows cellular networks to be automatically configured, optimized, diagnosed, and healed. Because of the growing costs, it is crucial in the future mobile networks and operations. SON also aids in improving network performance and customer experience, improving the Opex-to-revenue ratio, and avoiding Capex. Controlling the function inside SON is according to an algorithm based on collecting Management Data Analysis Service (MDAS) monitor network. This algorithm in 3GPP introduces some solutions, as Table 3 shows. These solutions are Centralized SON (C-SON) runs algorithms in a 3GPP management system. The other call Distributed SON (D-SON), which refers to implementing the algorithm in the network function layer. The last one is Hybrid SON (H-SON), where the execution of the SON algorithm distributes throughout the network function layers and management layers.

The main goal of introducing this use case is that the interoperability concerns in multi-vendor network installations might arise when the interfaces between multiple organizations hosting the SON algorithms are not standardized. Hence, O-RAN uses an architecture that xApp and rapp are trying to tackle this issue.

Table 3: Specifications of Different Implemented

SON Solution	**Explanation**
C-SON	In the 3GPP management system, the SON algorithms are run.
D-SON	The SON algorithms are implemented at the Network Function layer
H-SON	SON algorithm is distributed throughout the network function layers

3.6.2 Architecture

This section provides some information about SON architecture and the different components used in this use case. Figure 17 shows SON architecture and the components used in this UC. The A1 and E2 interfaces, the optional O1 interface, are utilized to implement the SON functionalities. Different SON functionality such as initial Physical Cell ID (PCI), Automatic Neighbor Relations (ANR), and energy-saving can implement by both management entities like EMS or SMO and Non-Rear-Time RIC by rApp. The information related to SON exchange using the O1, A1, and E2 interfaces. The xApps of Near-Real-Time RIC implement dynamic or time-sensitive SON functionalities.

To deploy a cell, information such as PCI need which uses to get location information in the E2 node.

The location information can send directly to the management entity by the E2 node. A management entity can share this data to rApp, position data such as azimuth, elevation, and altitude use by the first PCI rApp, to determine the potential neighbor nodes already deployed in and around the present node's location, as well as their related PCIs. This derived PCI serves as the initial PCI for the deployed E2 node. After receiving this data from the management entity to E2, it can be operational, and PCI SON can implement in the management entity. Hence, it can access the location of the E2 node in the management entity database. Third-party SON algorithms such as rApp, xApp, or management entity functions easily integrate into the O-RAN framework to provide maximum performance in a multi-vendor deployment condition.

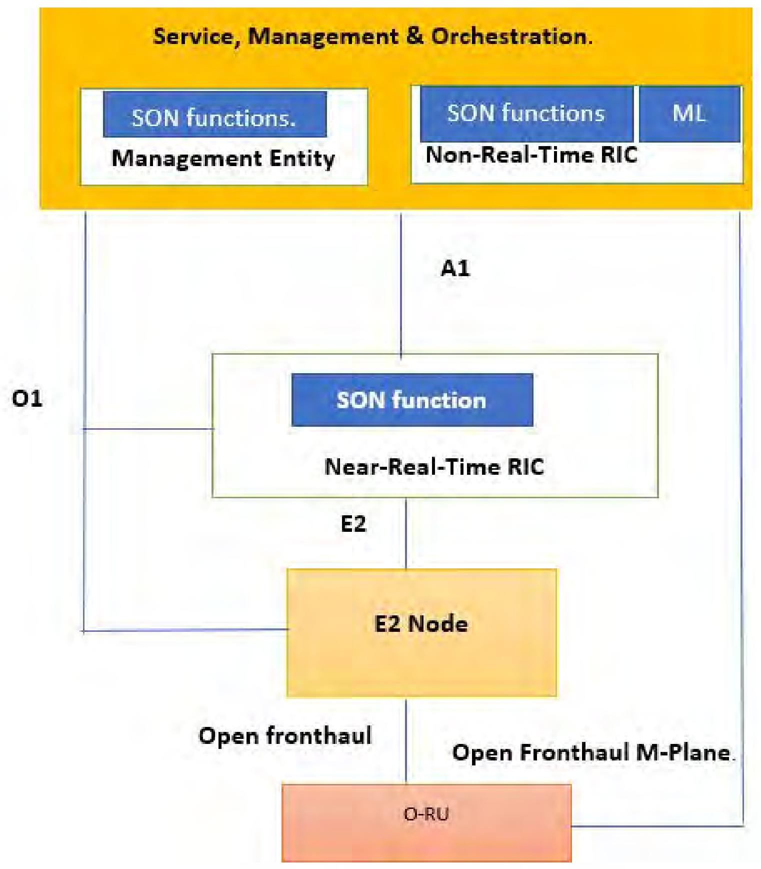


Figure 17: SON Functions

3.7 Use case 7-Local Indoor Positioning in RAN

3.7.1 Background

An Indoor Positioning System (IPS) is a network of sensors used to find people or things that GPS and other satellite systems fail or cannot recognize. These places can be inside buildings, airports, alleyways, and parking garages. It sends a real-time alarm to people to alert them to be far from a dangerous location. In Near-Real-Time RIC, a xApp deploys to measure the positioning of UE and optional velocity according to the data from the E2 node [26].

3.7.2 Architecture

In this UC, a different component in RIC is involved in its goal. The responsibility of Non- Real-Time RIC is ML/AI computational. It uses historical positioning measurements and labeled user location to train the model. E2 nodes can periodically send the position for Near-Real-Time RIC. To ensure the measure requirements, Near-Real-Time RIC alters the relevant positioning measurement configurations in E2 nodes. Near-Real-Time RIC, based on subscription, decides the positioning measurement report to the E2 node. The Near-Real-Time RIC positioning xApp may send the positioning findings to the SMO for further visibility. Additionally, Near-Real-Time RIC securely exposes location findings to adjacent edge applications [26].

4 Chapter 4- Kubernetes Attack

Near-Real-Time RIC xApps need to deploy in the Kubernetes. In this chapter, we briefly explain the kubernetes followed by the backdooring docker image that gives remote access to the attackers. The last section illustrates the possible attack that attackers apply to xApp.

4.1 Background

Kubernetes provides automatic deployment, scaling, and management of virtual machines and containers. It represents the system condition through a series of objects and describes Virtual Network Functions (VNF) and applications running on the kubernetes-managed cluster [31]. All software parts of the Near-RT RIC, including the xApps, are deployed as microservices in a Kubernetes cluster using helm [32]. Helm is a tool for managing packages. Helm uses to add packages, services, and other components to the Kubernetes-deployed APPs. There is container run time in Kubernetes to run and control the services required to operate containers. As figure 18 shows, one of these run times is docker, which is necessary for each node to manage the containers.
In Kubernetes, programs and how they can execute run by images. Containers are image instances that allow for the execution of several containers of the same image, each in a unique condition. Each xApp can include one or several containers that run different images [31].

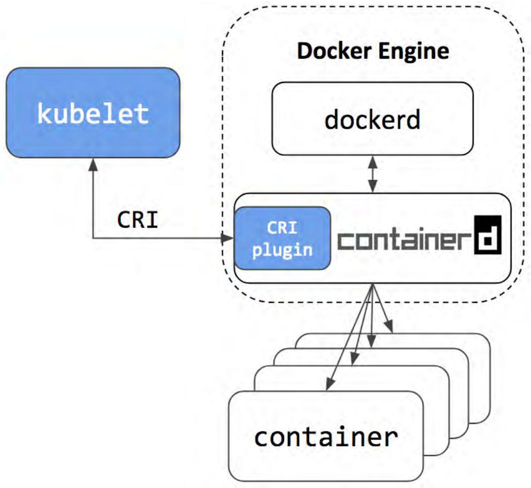
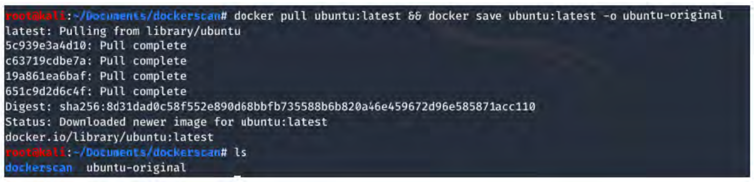


Figure 18: Docker

4.2 Backdooring Docker Image

The attack we consider in our study is backdooring existing docker images. Backdooring means giving authenticated to user or attacker to gain access to the system and change data and configuration. Typically, the docker hub uses to download the docker images so that this docker hub could be an appropriate place for an attacker to apply their modification. To run container an image which includes some policy and structure needs. These images, such as Centos, Ubuntu, and Jenkins, are freely available to everyone [33]. Researchers report that attackers have successfully infected public Docker repositories with malware by using open-source Docker repositories. After downloading this modified docker by a user and installing it in their system, attackers gain access to the network. The illegal image can apply different attacks on the xApp, such as running Bash Reverse Shell and adding the SSH key.

A reverse shell, also known as a connect-back, exploits vulnerabilities in the target system to start a shell session. Subsequently, obtain access to the victim's computer. The purpose is to connect to a distant computer and redirect the target system's shell's input and output connections so that the attacker may access it remotely. An attacker gets administrator access and executes commands directly on a victim's operating system after a shell connection is established [34].
SSH keys attacks are credentials that allow you to access servers without needing to put in a password [35]. The hackers have access to the system and can act as admin. In some cases, even if victims stop using or uninstall the malicious docker images, the attacker might have achieved persistence on their computers through other mechanisms, potentially enabling them access to the system at a later date [36].
In [37] shows how a Kubernetes image can give a reverse shell to an attacker after installing in Ubuntu. First, the image stores locally after being downloaded from the docker hub. In the next step, the image has injected a backdoor into the original image using dockers scan. After running the container, it gives the reverse shell to the attacker (Figure 19).



```
root@kali:~/Documents/dockerscan# docker pull ubuntu:latest && docker save ubuntu:latest -o ubuntu-original
latest: Pulling from library/ubuntu
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@kali:~/Documents/dockerscan# ls
dockerscan  ubuntu-original
```

Figure 19: Reverse shell from [37]

4.3 Threats against xApps

In [38] refer to some possible attacks inside the xApps. The first one is that attackers can use the misconfigured xApp to have access inside it. It can modify and catch the sensitive transmitted data over A1 and E2 interfaces. This attacker influences a cell, a group of UEs, and a specific UE. In addition, it tracks a specific subscriber or influences a subscriber or a certain area. The other attack discusses in this article is, conflicting xApps decrease performance or cause a DoS by inadvertently or deliberately interfering with O-RAN system functionalities. Since there is no specific functional split between Near-Real-Time RIC and O-gNB and all depends on the accessible xApps and the capabilities of O-gNB, it leads to a conflict decision between Near-Real-Time RIC and the O-gNB. The last threat is compromise isolation of xApps, meaning that attackers can use the vulnerable xApps to access it. An attacker can extract information from co-hosted xApps by exploiting side effects. Sharing resource utilization leads to abuse of components for intercepting, spoofing network communications, and element for degrading services.

4.4 Attack Model

This work presents the possible attack inside UCs, based on Spoofing, Tampering, Repudiation, Information disclosure, DoS, and Elevation of privilege. We briefly explain each of these attacks.
The act of adopting the identity of another person or entity and utilizing that identity to commit fraud is known as identity spoofing. Spoofers use password attacks and credential capture methods to steal credentials from people or organizations.
Tampering attack is the purposeful alteration of items that become dangerous to users. Tampering attacks have consequences, stealing crucial information, deleting the files, illegal conversations being eavesdropped on, and altering vital communications.
The repudiation attack occurs when an application or system fails to appropriately track and log users' actions, allowing for malicious modification or forgery.
If a user is not supposed to be able to access sensitive and confidential information within an application, information disclosure occurs.
DoS or Denial-of-service attack is a cyberattack in which the offender tries to render a computer or network resource inaccessible to its intended users by temporarily or indefinitely interrupting the services of a network host.
Elevation of privilege exploits a bug or a configuration oversight in an operating system or software program to acquire elevated access to resources.

5 Chapter 5 - Analysis of the attacks on the use cases

5.1 Criteria model

In our model, we use different criteria to analyze attacks.

Reproducibility: This criterion indicates the ability to recreate the attack. If the attack executes in the system several times, it presumes high. Instead, if it duplicates under some conditions, it assumes medium. Low criteria occur when an attacker cannot recreate the attack due to its sophistication.

Stealthiness: This criterion shows how simple for a potential victim to detect an ongoing attack. We define high stealthiness when the victim needs specific and dedicated devices to identify the attack. While, medium indicates that to recognize an attack, various misbehavior needs. In contrast, in the low criteria attack is easy to be recognized since the victim gets informed about the attack.

Exposure: It represents the number of affected users. If the attack involves all the users, we consider this criterion High. When a significant number of users are affected, it assumes as a medium. Low indicates that a particular number of devices are involved.

Impact: We consider it in terms of the three basic security principles:

Confidentiality indicates that any attack provides information about the xApps. Integrity means If the attacker can modify data inside xApps. While availability implies that an attack causes the operator or developer loses control of UC.

Table 4 summarizes the information about the different criteria discussed previously.

Table 4: Criteria

Criteria	**Type**
Reproducibility	**High**: Recreate attack is easy, **Medium**: Recreate attack under some condition, **Low**: Due to some condition in use case attack cannot be recreated.
Stealthiness	**High**: some specific device is needed, **Medium**: Some misbehavior needs to detect attack, **Low**: Attack can be recognized clearly.
Exposure	**High**: All users are affected by attack, **Medium**: Significant number of users are involved attack, **Low**: Small number of users are affected.
Impact	**Confidentiality**: Attack provide some information about xApp, **Integrity**: Attacker is able to modify data inside xApp, **Availability**: Lose control of UC.

5.2 V2X attack

A V2X use case can help increase safety, improve traffic, and reduce emissions. A xApp in this UC consists of some vital data to achieve this goal. The data is used in xApp to produce some messages accordingly and send a message to the driver [26]. The xApp in this UC is in charge of sending messages among vehicles, and it deploys inside Kubernetes. It assumes that backdooring the image permits attacker access to the kubernetes. As a result, each vulnerability inside it gives a chance to malicious users to have access to the Kubernetes and modify the xApp data or apply a different attack.

The first possible attack that we consider inside V2X is tampering. In this case, the compromised xApp sends bogus messages throughout the network by creating or changing an existing message. It means that this attack impacts the integrity of the communication. In this attack, the malicious user needs to show some misbehavior to detect. This misbehavior can be a delay in sending the message to a destination. There must be misconduct to recognize tampering, so a user may be affected before the attack is detected and blocked. Therefore according to the mentioned criteria, stealthiness assumes medium, and reproducibility and exposure consider high.

In both circumstances, it deceives the cars by providing false information and places them in peril. A rogue vehicle can send out fake traffic and safety alerts or inaccurate traffic estimation information that differs from actual data, interrupting traffic or causing an accident. Another attack is replay which occurs when an attacker inside xApp saves event information and sends it later, even if it is no longer valid. In this scenario, a vehicle needs the periodical message from the xApp. However, the attacker ignores sending data at a vital time to the drivers.
In location-based replay attacks, for example, the attacker records an authenticated message at point A and immediately transfers it to place B. Similarly, the attacker captures a valid message at time t1 and sends it again at time t2 at the same place. As seen in Figure 20, attackers can send a message to the victim and provide incorrect information about the position by manipulating data.
An assault recognizes once the driver finds out the data include inappropriate attributes about the other car's position or pedestrian.

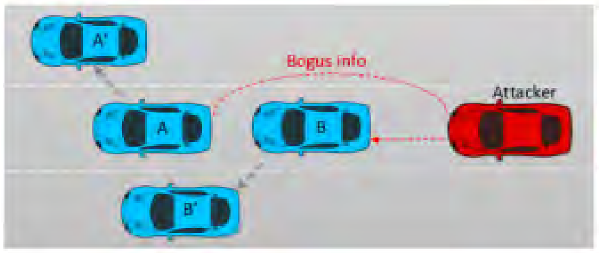
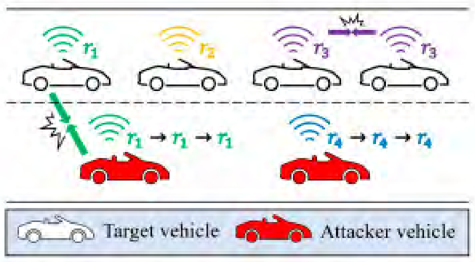


Figure 20: Tampering Attack

One of the data send from the V2X application server to xApp is GPS information, including the location of the vehicles. With this data, xApp can compute the position of the other cars, pedestrians, and infrastructure and send the data accordingly. Attackers by GPS spoofing can inject false position data and send it to the victim's vehicles. It sends a message with a fake location or after time has passed. In general, GPS is in charge of transmitting both position and time to the near vehicle. Incorrect data, such as false location information, might reduce message delivery efficiency dramatically. GPS spoofing requires specific devices to be detected as in [39] and [40] is explained. It means that the stealthiness of this attack is high. The absence of these devices leads to repeated attacks on the network, resulting in accidents, and crashes among the vehicles. Thus, the invasion affects users and gives them the wrong location, resulting in effects on a large number of vehicles, creating high exposure. The attack can regenerate within the network, so reproducibility is also considered high. The impact is the other criterion we consider in our study. According to the explanation of impact type, since the attacker modifies the vehicle's current location, it assumes integrity.

Another attack inside V2X is the DoS attack. Such assaults can raise the latency of V2X connections and decrease the network's reliability by sending the traffic to xApp. The attacker sends the traffic to xApp in an attempt to consume enough server resources to make it unresponsive to legitimate traffic. DoS attacks have become one of the most dangerous attacks against vehicular networks due to their ability to congest the Radio Frequency (RF) spectrum and prevent vehicles from accessing necessary RF resources, as well as their ability to block the flow of safety-critical information between cars and deny vehicles access to Road-side Units (RSUs).

It is easy to recognize this attack since it comes from one source, and the aggressive node can be blocked. For this reason, the stealthiness is low in Dos attack. In our example, if attackers can collaborate and send information about their target, they do not attack one vehicle simultaneously. In this way, they can impact a significant number of cars. The exposure criterion is assumed high. This attack, in addition, needs to listen to the messages that come from the victim and apply its attack accordingly. It means that attackers can collect various information about their target to recreate, and reproducibility assumes high the attack. Since the hacker can access the data of the other users by listening to their data, the impact considers confidentiality. Figure 21 shows a specific example in which one attacker picks a resource block randomly, making the attack successful until the victim updates its resource block. The other attacker picks a resource block that is not employed. No target vehicle picks that resource block as long as the attacker is utilizing it, meaning that the attacker can not succeed in his attack.



r_1
r_2
r_3
r_3
$r_1 \rightarrow r_1 \rightarrow r_1$
$r_4 \rightarrow r_4 \rightarrow r_4$
Target vehicle
Attacker vehicle

Figure 21: DoS Attack from [41]

The second type of DoS attack happens when the attacker vehicles search for resource blocks that are only used by one vehicle and choose one of them. The opponents must listen to signals broadcast by target vehicles over a while to compile a list of lonely cars' resource blocks from which they must select their own at random. Attacker vehicles do not collaborate, which means they may attack the same lonely car many times throughout one assault phase, lowering the attack's efficacy.
The last possible DoS attack, as Figure 22 indicates, selects different cars by cooperating among attackers. With this attack, the attacker listens to all messages from lonely vehicles. During this type of attack, attackers work together to select different victims. In this type of DoS attack, two xApps hack by malicious users, and they exchange information together about their victims [41].

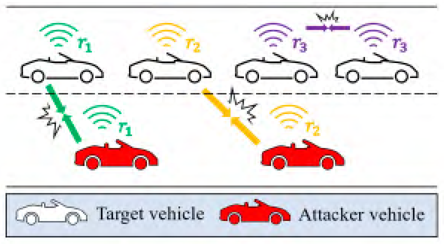


Figure 22: DoS Attack from [41]

During a Non-Repudiation attack, the periodic message and alert message sign with the sender certificate. The location-based services, as well as any sensitive data, are encrypted. As a result, when a node engages in harmful conduct, its identity may be determined.
This attack needs a signed certificate for sending messages to the victims and can be detected quickly. Stealthiness assumes low due to this behavior. As a result of the possibility of recognizing the identity of senders and blocking it immediately, the power of recreating it considers low and affects the minimum number of users. Therefore, the reproducibility and exposure level assumes low

in this attack. Additionally, the message can be modified and sent to the vehicles so that the impact criteria are considered integrity.

Another attack is called information disclosure, in which attackers have access to the data inside the xApps and begin misusing them. The obtaining information might also reveal the location of backup/temporary files and ID server. Consequently, attacker access to it causes unavailability of the whole network. It also shows sensitive content such as passwords and signatures to other users and permits them to act even as a cluster admin.
This type of attack involves having access to sensitive data and sharing or modifying them with others and causing impacts on confidentiality and integrity, respectively. Every time an attacker attempts information disclosure, it can involve real users. In some cases, the attacker pretends to be a cluster administrator, recognizing that the attack could be almost impossible and needs some specific app to distinguish between attacker and real admin. This behavior considers stealthiness in the mentioned criteria, and this use case assumes high stealthiness. As a result, attackers can gain access to the cluster-admin and replicate multiple attacks simultaneously, leading to high reproducibility and exposure.

Elevation of privilege attack could be another attack inside V2X. Attacker, after access to the xApp by image docker backdooring, can act as cluster-admin and change the RBAC permission of the users. The user, for example, by RBAC permission of view, can be promoted to admin RBAC and therefore be able to alter the xApps data. The attacker seizes administrator rights for the transceiver and uses it to access other devices connected to the onboard network by remotely providing malicious codes and commands. The attacker's usage of devices and functions causes the accident evasion system to fail and affects driving safety.
Similar to the previous, tries to impersonate an administrator to obtain access to the vehicle's data and modify or share it with others. So, the impact of this attack is on confidentiality, integrity, and availability. Some tools need to recognize these attacks and avoid running malicious messages so that stealthiness is high. Access to cluster admin and modifying xApps data allow attackers to impact the high range of users and collect sensitive information about that users. These sensitive data apply to the other attempt inside the network. It causes to consider reproducibility and exposure criteria as high.

These attacks and misbehavior in sending the message to the driver and pedestrian have some consequences such as accidents between vehicles, crash pedestrian increased the traffic jam and emissions.

Table 5 summarizes the possible attack inside the V2X use case followed by their criteria.

Table 5: V2X attack analyses

Attack	Stealthiness	Reproducibility	Exposure	Impact
Spoofing	High	High	High	Integrity
Tampering	Medium	High	High	Integrity
Non-repudiation	Low	Low	Low	Integrity
Information disclosure	High	High	High	Confidentiality Integrity
DoS	Low	High	High	Confidentiality
Elevation of privilege	High	High	High	Integrity Availability Confidentiality

5.3 Massive MIMO Beamforming Optimization

Massive MIMO is one of the keys to the next-generation wireless network. We expect cellular 5G BS includes a wide antenna array, such as hundreds or thousands of antennas, enhancing spectral efficiency by magnitudes over a conventional MIMO system. The trained data transfer to RAN consists of horizontal beam width, vertical beam width, beam azimuth maximum, and average transmitted power per beam/direction. A xApp with the goal of managing beam pairing deploy in this use case. The performance of BF is heavily dependent on the selected beam pattern. A UE must perform cell search, select operation, and collect initial cell synchronization and system information before connecting with the network.
Identity spoofing is one of the possible attacks on this UC. Here we consider a use case involving a BS, a user, and a spoofing attacker who uses a false identity to spoof another node. Beam pairing uses to increases facilitate communication between the user and the BS. The attacker can use spoofing to masquerade as a verified user and subsequently change arbitrary fields and even modify the authentication key after sniffing the traffic between the BS and valid users for a long enough time.
Some algorithms and applications need to recognize the spoofing attack, as explained in [42]. It means that the stealthiness of this attack is high. Spoofing attacks can affect integrity and availability because they can steal the identity of legitimate users. Reproducibility and exposure assume high since when the attack occurs, it can affect all the users connected to the network and start executing its attack.
The other possible attack is tampering with the data inside the xApps. As we mentioned, xApps include information about the location of the appropriate BS to beam pairing. Hackers modify the position of the BS and send the data to an arbitrary destination. It leads to losing control of the use case and availability and integrity.
To recognize assault, victims need to show inappropriate behavior, including the

inability to find the beam pairing. The need for detecting attacks leads to its assumption of medium stealthiness. Aside from that, it can affect all the devices and recreate the attack over an extended period, leading to high reproducibility and exposure level.
The other attack is information disclosure. In this attack, malicious users can share the xApp's data with other users. After revealing information to an unauthorized user, she has the opportunity to have access to sensitive data and control the whole system. The data send in an illogical direction and leaves the system inaccessible. Therefore, the impact of the attacks considers confidentiality, integrity, and availability. In an information disclosure attack, some algorithms should define to recognize this attack. It also can reveal information about the significant number of users to illegal users. It means that these users can use this information for other attempts in the network and recreate the attack. Therefore reproducibility, stealthiness, and exposure assume high in this attack.
A repudiation attack is the third attack we analyze in this UC. In this attack, the attackers start modifying the message and data inside xApps. In this attack, the system can not trace the malicious activity to identify a hacker leads to high stealthiness. If assaults are not detectable due to a lack of data, they may last longer and cause severe damage. These damages range from losing control of the UC to system unavailability, leading to integrity and availability impact. Therefore, the attacks can stay inside the system for a long time and affect all its users by creating a new assault. It conducts to high reproducibility and exposure.
DoS attacks can decrease performance by transmitting large amounts of illicit traffic and utilizing system resources that the system becomes unusable, denying authorized users access. We mentioned that docker backdooring permits attackers to access the xApps. It means that the xApp can receive traffic of requests from the illegal user and cause the unavailability of the xApps. It can receive manipulated data about the beam and causes the availability of the UCs. An intrusion detection system uses to mitigate and detect the attack. Since aggressive behavior can be identified by users, we consider high stealthiness. If even one malicious user can access the system, it can take advantage of the network and impact all the users. It means that the exposure and the reproducibility consider high.
With an elevation privilege attack, the attacker can reveal xApp sensitive data to other users or even change their RBAC permission. It means that a specific user may be able to access sensitive information and modify them, identical to a tampering attack. They can swipe the ID of the other users and avoid providing data to the legitimate user. It leads to the integrity, availability, and confidentiality of massive MIMO. Since a pirate can expose the sensitive data and save it for the next attempt, we consider the reproducibility high. Every attempt involve users connected to the network, which leads to a high exposure level. Because this attack can detect through specific algorithms, it assumes a high level of stealthiness. Table 6 summarizes the possible invasions in the MMIMO UC and the criteria for each attack.

Table 6: Massive MIMO attack analyses

Attack	Stealthiness	Reproducibility	Exposure	Impact
Spoofing	High	High	High	Integrity, Availability
Tampering	Medium	High	High	Integrity, Availability
Repudiation	High	High	High	Integrity, Availability
Information disclosure	High	High	High	Confidentiality Availability, Integrity
DoS	High	High	High	Availability
Elevation of privilege	High	High	High	Integrity, Availability, Confidentiality

5.4 O-RAN Signaling Storm Protection

As mentioned before, O-RAN signaling storm protection UC is used to avoid and predict DDoS attacks inside the network by analyzing Channel Quality Indication (CQI) reports or volume of Negative-Acknowledgment (NACK) based on signaling messages. In this use case, one xApp with two functionalities uses to detect and mitigate the attack. The core defensive mechanism established against threats originating from devices toward the network depends on device configuration and faith that would legitimately follow mobility standards limitations. One way to limit access to the network by different devices is a back-off timer which restricts the repeated devices to the network and leads to blocking the device from sending too many attachments to the network. If the network cannot limit access, it ignores malicious and real devices trying to access the system. It causes a DDoS attack on the system.
We mentioned that the xApp goal is to detect the aggressive node and then block it. It means if the repudiation attack occurs in the network can recognize easily. If the attacker, tampering attack, changes the ID of the malicious node, the blocked message send to the legal user or an arbitrary node. Consequently, the invasion continues inside the network and impacts the system quality. In addition, the attacker can apply other attacks by changing the RBAC permission of the users. For example, they can limit the access of the legal user. The user with the view authority is improved to admin and starts managing the information inside the O-RAN. It leads to an elevation of privilege attack. It also can provide information to the illegal users about the vital data of the xApp, such as the user ID and their cluster key, and information disclosure attack. It causes impacts confidentiality. After this data reveals to unauthorized users, it can use for spoofing attacks by users' IDs. The attacker who gains access to the network

can generate an attacking storm which might cause the network to go down for a long time. As a result, the attacker can continue this attack for hours and select a significant number of devices. Exposure level and reproducibility could be high since it affects the devices connected to a network. These malicious devices owing to their behavior, are easily recognizable. So their stealthiness is low. In terms of impact criteria, there is integrity, availability, and hackers can alter the xApp. This UC and possible attack, and their criteria, are outlined in Table 7.

Table 7: O-RAN Signaling Storm Protection attack analyses

Attack	**Stealthiness**	**Reproducibility**	**Exposure**	**Impact**
DDoS	Low	High	High	Availability Integrity
Repudiation	Low	High	High	Availability Integrity
Tampering	Low	High	High	Availability Integrity
Spoofing	Low	High	High	Availability Integrity
Information disclosure	Low	High	High	Availability Integrity
Elevation of privilege	Low	High	High	Availability Integrity Confidentiality

5.5 Congestion Prediction and Management

Congestion prediction and management offer two ways to predict congested cells. One way is that Non-Real-Time RIC by the O1 interface directly helps to reduce it and the other way is that CPM rApp using the A1 interface sends the created interface to CPM xApp. This interface includes cell ID, the time stamp of cell congestion, predicted KPI, and some data about whether the cell is congested or not. After receiving this interface, CPM xApps offer three solutions to predict and avoid congestion inside the network. These ways include directing some traffic to one server and some traffic to another (load sharing), preventing user access, and shifting to a dual-mode network. Since hackers can take the identity of a legitimate user and transfer data with their authorization, they are undetectable.

As we discussed earlier, cell ID and KPI is one of the input data in xApp. In the scenario that an attacker modifies the xApp, by inserting malicious files and information about the cell ID, it leads to sending more data to the cell and carries data more than it can handle. Moreover, by changing the cell ID, the congested cell cannot be determined by xApp, resulting in data loss, packet loss, queuing delays, and blocking of existing connections. This attack considers tampering. Since the attackers alter the xApp data, they do not need to sign the information, resulting in an inconceivable method of identifying the malicious users, known as repudiation. Hackers by information disclosure, in addition, can share sensitive data with other users and cause misusing the xApp data. They also can change the level of access users and permit them to apply their manipulation by elevation of privilege attack. This misusing can be changing data and even their certificate. It also leads to a DoS attack in which hackers attempt to send several messages by spoofing the ID address to the network and causing traffic. As a result, the network is shut down or slowed down and cannot predict the congested cell. Since hackers can take the identity of a legitimate user and transfer data with their authorization, they are undetectable. Hackers can smoothly share the xApp information with the illegal user with sensitive information such as user ID and signature misuse by malicious users to gain access to the system. This attack is known as information disclosure. As a result of these attacks, users get infected. It cannot be detected easily, so the algorithm is required to recognize it, which leads to high stealthiness. Attackers using a novel attack inside the network can start recreating the novel attack to congest the cell. It means that the reproducibility and exposure are assumed high. Accordingly, illegal users cause the system to go down, share xApp data with other users, and even modify it, so its impact considers availability, integrity, and confidentiality.

Table 8 briefly explains each of the assaults discussed during this section.

Table 8: Congestion prediction and management attack analysis

Attack	**Stealthiness**	**Reproducibility**	**Exposure**	**Impact**
Tampering	High	High	High	Integrity Availability Confidentiality
Repudiation	High	High	High	Integrity Availability Confidentiality
Information disclosure	High	High	High	Integrity Availability Confidentiality
DoS	High	High	High	Integrity Confidentiality Availability
Elevation of privilege	High	High	High	Integrity Confidentiality Availability

5.6 Dynamic Spectrum Sharing (DSS)

The other UC that analyze is DSS which allows operators to dynamically share spectrum in existing 4G deployments with 5G systems depending on network loads and resource sharing regulations. In this UC, the main functionality of the xApp is to translate the policy and configuration received from Non-Real-Time RIC and then send them to the RAT-App 4G and RAT-App 5G.
The Near-Real-Time DSS app serves as a central office, receiving data from the DSS-APP in Non-Real-Time and interpreting it for the RAT-APP. An attacker can spoof a spectrum xApp, and deny sending the translated data to the proper RAT-APP 4G and RAT-APP 5G. Hackers can use tampering attacks to manipulate translated data in xApp before sending it to the RAT-APP. The traffic of a valid user may be redirected to a bogus server by malicious parties. Similarly, assaulting the user with a high number of fake inquiries may result in the actual user denying service or a DoS attack. Hackers can change the level of access to users inside a network and reveal sensitive data, such as certificates and passwords, to unauthorized users by elevation of privilege. This attack happens on the condition that the infected user has cluster-admin permission. If it happens in the network, the malicious user can misapply the user identity for sending data, causing a repudiation attack. The attacker is a super user and can assess and modify all network policies. It is a dangerous attack that alters, modifies, replaces, or deletes information before reaching its intended destination. It deceives the xApp and forces it to make the wrong decision to send the data to the appropriate RAT-APP 4G and RAT-APP 5G.
Attacks inside this UC recreate easily and affect users since they can access the sensitive data inside the network and use them for the other attempt. It leads to high reproducibility and exposure level criteria. Infected devices can demonstrate unusual behavior during these assaults and detect easily, leading to low stealthiness. Moreover, these attacks impact availability, confidentiality, and integrity since the attacker can tamper the sensitive data. In this section, we discuss attacks on Dynamic Spectrum and Sharing, as shown in Table 9.

Table 9: Dynamic Spectrum Sharing attack analysis

Attack	Stealthiness	Reproducibility	Exposure	Impact
Spoofing	Low	High	High	Integrity Availability Confidentiality
Tampering	Low	High	High	Integrity Availability Confidentiality
Repudiation	Low	High	High	Integrity Availability Confidentiality
DoS	Low	High	High	Availability Integrity Confidentiality
Elevation of Privilege	Low	High	High	Availability Integrity Confidentiality

5.7 Self-Organizing Network or SON

SON is the other use case we consider in this study. It provides the network functionality for 5G by introducing some characteristics such as automated configuration, optimization, management, and healing of mobile network faster. Self-management characteristics of this UC give the network opportunity to improve its performance. A xApp with the sensitive SON function deploy to derive the possible neighbor nodes and associated PCIs. This sensitive data includes location information such as azimuth, elevation, and altitude. Data collected from the UE allows xApps to make decisions regarding network management. If xApp hacks, the malicious user can control the information inside the xApps. Resulting of tampering with the data is network downtime and decreased user experience over private cellular networks. Attackers can spoof the neighbor's IP address and send the wrong data to the network about confidential information. Since data transmitted by the legal node IP lead to a repudiation attack. In this scenario, xApp receives incorrect data from the user and makes decisions based on that information, since for managing the network, inaccurate data use makes the system unavailable. Information disclosure and elevation of privilege attacks can be the other possible attacks in this UC and lead to confidentiality impact. In the scenario of hacked xApp having an RBAC permission with admin or cluster-admin, a malicious user can misuse this characteristic and change the other user's permission. An attacker can also create xApp with their configuration. Therefore, the information to manage the network avoids deriving the possible neighbor node. These attacks lead to a decline the system performance. Identifying the attacks is simple due to some misbehavior from the illegal users, so stealthiness is medium. Hackers can affect the users by stealing their IP, so the high exposure assumes in these attacks. The xApp's data manipulation causes the system to become unavailable and lose control of the UC, and changing information implies system integrity. These attacks summarize in Table 10, along with the goal of UC and its criteria.

Table 10: Self-Organizing Network(SON) attack Analysis

Attack	Stealthiness	Reproducibility	Exposure	Impact
Tampering	Medium	High	High	Integrity Availability
Repudiation	Medium	High	High	Integrity Availability
Spoofing	Medium	High	High	Integrity Availability
Information disclosure	Medium	High	High	Integrity Availability
Elevation of privilege	Medium	High	High	Integrity Availability Confidentiality

5.8 Local indoor positioning

Local indoor positioning includes UC in the industrial manufacturing business. It applies real-time safe to warn operators to stay away from dangerous areas. In this UC, a xApp is in charge of computing user positioning and velocity. Once hackers have access to the xApp through the backdoor image, they can perform their attack. One of these attacks is tampering. The hacked xApp allows them to manipulate the user's location, send this message to the victim and give them inaccurate information about a dangerous place or store recommendation. Imagine that a user wants to go to the nearest pharmacy. Since the user location has changed, the xApp cannot propose the closest destination for the users. This misbehavior is more vital if this UC uses to find unstable areas in industrial places. An attacker can also deploy their xApp inside the network. After the message sends to the user with malicious xApp, their identity could be spoofed and used for other attempts. Additionally, it can force the xApp to send multiple modified messages to the interface generator, preventing them from communicating within the wireless networks, a DoS attack, and repudiation. By access to the API, the malicious user could also share exposed xApps data with other users with an information disclosure attack. It has an impact on the confidentiality of the system. Similar to the other UCs, illegal users can act as cluster admins and change their level of access to the users.

These attacks can only identify after the attacker exhibits some suspicious behavior, meaning that stealthiness can sees as a medium. Every attempt can also affect a large number of users. Because illegal users can change xApp data and share it with others, their impact on integrity and confidentiality consider.

Having access to the API, an attacker can recreate the attack for the period and cause high reproducibility.

In Table 11, you can find a brief description of this UC, followed by some possible attacks inside.

Table 11: Local indoor positioning attack analysis

Attack	**Stealthiness**	**Reproducibility**	**Exposure**	**Impact**
Tampering	Medium	High	High	Integrity Confidentiality
Repudiation	Medium	High	High	Integrity Confidentiality
Information disclosure	Medium	High	High	Confidentiality Integrity Confidentiality
Spoofing	Medium	High	High	Confidentiality Integrity Confidentiality
DoS	Medium	High	High	Integrity Confidentiality
Elevation of privilege	Medium	High	High	Integrity Confidentiality

6 Chapter 6

6.1 Discussion

The analysis performed in the previous section shows that different attacks of the proposed model bring various consequences to the UCs. In the V2X use case, attacks conduct by sending fake traffic queues and crash the drivers. It also avoids sending the message in proper time to vehicles and consequently declines the network stability. In the event of a hacker admission the admin ID and acts with the admin permission, it can modify the whole network without being recognized. Due to a xApp attack, there is an increase in traffic and decreased protection for drivers. In the massive MIMO use case, a malicious user shifts the beam's position and directs it to an unintended area. In this way, users cannot access the system and exchange their data, which results in a decline in the performance of beam pairing. The other use case we analyze is O-RAN Signaling Storm Protection which use to predict the DDoS attack. Hackers can produce a storm attack and allow a thousand devices to aggressively connect to the network per hour, leading to system unavailability.
Nevertheless, this attack impacts a significant number of users, and it can identify thanks to the behavior of the aggressive devices. CPM use case offers the ability to predict the congested cell. Illegal users steal the user identification and start to alter or send the xApp data by that ID. These malicious users sent the data to the congested cells more than their capacity, leading to cell unavailable and packet loss. The attacks in the DSS use case influence the delivering the information to the proper RAT-App 4G and RAT-App 5G. If xApp cannot send this information or send the wrong data to RAT, it causes completely inaccessible systems. Similarly, in the SON use case, inaccurate information causes network downtime that impacts user experience. In the last use case, local indoor positioning, messages are based on user location. If this location manipulates or calculates inaccurately, it cannot give suggestions about the dangerous places or destinations they seek.

In conclusion, in terms of the importance of the use cases, we should mention that V2X and local indoor UCs can be the most crucial use case, and improving security among them comes first. These UCs directly impact the safety of the users.
Consider that a driver and user rely on messages from the server and use instructions accordingly. In this case, the wrong message directs the users to dangerous places and causes an accident in V2X and local indoor place users. Users' experience may be negatively affected by attacks on the other mentioned UCs, but they do not affect people's safety. They only have an impact on the communication of the network. In a massive MIMO UC, for example, the attack can avoid accessing the user to the network by changing the location of the BF. Regardless of the ability to duplicate attacks in all use cases, it should note that all the attacks cited can reproduce several times and affect the system's performance.

Based on the ability to recognize the attack, we can conclude that it would be challenging in almost all cases. However, Identifying attacks in the O-RAN signal storm protection UC is simpler since aggressive devices behave differently. In the V2X use case, similarly, DoS attack and Non-repudiation are simple to detect, corresponding to the other possible invasion inside. Therefore, it enables users to detect and block aggressive behavior before the system breaks down.

Elevation of privilege attacks could be dangerous in massive MIMO, V2X, CPM, signal storm protection, SON, and DSS use cases that impact all criteria mentioned, including confidentiality, integrity, and availability (CIA) of all use cases. In contrast, in local indoor positioning use cases, it merely affects the integrity and confidentiality of the use case. It means it does not cause losing control of the use case and leads to severe conditions.

We acknowledge that there are other ways besides docker image backdooring to give attackers a chance to access the kubernetes and apply their modification on xApps. One area of future work is combining the knowledge gained in this thesis and suggesting other ways of accessing illegal users to the xApps to enhance the security of the suggested UC.

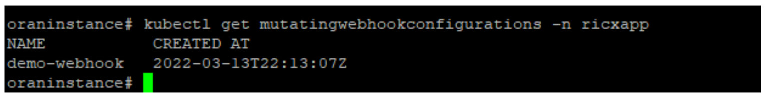
7 Chapter 7

7.1 Mitigation Strategy

In this section, we try to cover the possible ways to avoid backdooring the docker image in the Kubernetes. Every user in the kubernetes has a different level of access. This access can range from just viewing the information to cluster-admin as a super user in the Kubernetes. one solution to restrict the backdoor attack is the principle of least privileges, meaning that you should not use admin users to execute containers. In this way, even if an attacker gains access to the xApp, it cannot act as an admin and modify the information of that xApp.
The other method use official verified and signed docker images. Docker is a collection of software development tools for creating, sharing, and running individual containers. One way to verify the trusted docker image is Docker Content Trust (DCT). It is possible to verify publishers during runtime using digital signatures. In addition, you can create an additional security layer and use images from protected registries. These protected registries can be Harbor and Quay. Harbour is an open-source registry that secures artifacts by enforcing policies and role-based access control, ensuring that images scan and are free from vulnerabilities, and signing those images as trusted. Quay, a RedHat-powered image registry, identifies and fixes security vulnerabilities in your images.

The other way is to limit the CPU in the docker container. Docker containers come without memory and CPU limits by default, so you should set them up. It reduces the risk of a DoS attack. You can set up a memory limit, for example, to prevent your container from consuming all available memory.

The other solution is the admission controller webhook. As we mentioned, the assaulter can modify the xApps data and put their image and configuration inside xApps. This way, even if hackers modify xApp metadata after passing through the admission controller webhook, they convert to the default configuration. As shown in Figure 23, we deploy a webhook admission controller in our Kubernetes and tries to change the xApp images. However, the command shows that the change applies, but in xApp's configuration indicates the default image. Once malicious users gain access to xApps, this solution prevents tampering attacks. It is possible to deploy several admission controllers simultaneously in every system.



```
oraninstance# kubectl get mutatingwebhookconfigurations -n ricxapp
NAME           CREATED AT
demo-webhook   2022-03-13T22:13:07Z
oraninstance# 
```

Figure 23: Admission Controller

8 Chapter 8

8.1 Related work

O-RAN Alliance Security Task Group (STG) collaborates with O-RAN Alliance working groups to find the security of the O-RAN interface and components [43]. STG is currently analyzing the potential new attack surface, merging by separate of the O-DU and O-RU in the RAN. The analysis conducts on the other O-RAN-defined interfaces such as A1, E2, O2, and Open Fronthaul CUS-plane. Management interfaces of the O-RAN are secure via TLS/SSH with mutual authentication using access controls that can integrate into an operator's identity management platform, centralized logging, and input validation. Furthermore, the STG leverages existing O-RU security capabilities and investigates additional capabilities for securing the open fronthaul interface between an SMO and an O-RU.
In [19], Dudu Mimran et al.,2022, also analyze the O-RAN security components. It describes relevant risk areas, and threat actors identify the components of the ORAN targeted by threats and explain the operational range necessary to execute a given threat. The risk area in O-RAN is considered five various components in this article. They include cellular infrastructure, architecture openness, cloud and virtualization, machine learning, and 5G architecture. The actors are hardware manufacturers and suppliers, UE, external users, ISV infrastructure and application, and operator insiders. The other one is a threat which indicates the places that actors deploy assault inside the mentioned components. It analyses the amount of vulnerability on the O-RAN components according to the mentioned criteria. The security aspect of the article covers the analysis of the interface in O-RAN. However, our work is trying to interpret the security inside the xApp and the use cases based on it.

9 Chapter 9

9.1 Conclusion

The last few years have seen some of the most rapid technological advancements in networks, and the O-RAN has the potential to be the next major distribution technology. The traditional network design with a closed architecture means that operators had to use the same manufacturer for radio and baseband equipment. On the other hand, O-RAN helps operators with its emphasis on intelligence and open interfaces and standards, which supports multi-vendor installations and allows for customization to meet the demands of particular operators. According to the O-RAN Alliance, its primary ideas are openness and intelligence.
This idea is the key to increasing the number of operators using the O-RAN architecture. Consequently, it increases the number of attackers to create unique ways to have access inside these devices and apply their modifications. Therefore, all aspects must weigh to increase security in this architecture. One way is to improve the safety inside the xApps focus in our work.
In this thesis, we analyze the architecture in each use case-based xApps and then define their possible attack. Moreover, in this study, we use four criteria to interpret the power of recreating the attack, detecting the attack, the number of infected users, and eventually, their impact on each use case. In two of the use cases, v2x, and local indoor positioning, that we analyze in this thesis, since having a direct impact on the safety of the users, the security inside them should be a vital task. For example, in the V2x use case, the driver decides according to the messages from xApp. Hence, if these messages are manipulated and give incorrect information about the other V2X like pedestrians or traffic lights cause accidents. In terms of identifying the attack inside the use cases we analyze, O-RAN signal storm protection, and in V2X when DoS and Non-repudiation attack occurs, recognizing attack would be simply due to the different behavior from the malicious devices. Elevation of privilege attack is the most dangerous attack inside the use cases. Based on our analysis, it impacts all use cases we consider and threatens the confidentiality, integrity, and availability of use cases. As a result, this attack causes the network to go down.

What we know about upcoming years, they will be dominated by smartphones and IoT gadgets. Therefore, increasing security among these devices is one of the major concerns in today's world. Focus on improving safety in xApps based on O-RAN architecture can be a key to reaching this goal.

References

[1] Pratheek S Upadhyaya, Aly S Abdalla, Vuk Marojevic, Jeffrey H Reed, and Vijay K Shah. Prototyping next-generation o-ran research testbeds with sdrs. *arXiv preprint arXiv:2205.13178*, 2022.

[2] Parallel Wireless community. Every things you need to know about oran. *https://www.parallelwireless.com/wp-content/uploads/Parallel-Wireless-e-Book-Everything-You-Need-to-Know-about-Open-RAN.pdf*, 2020.

[3] Nasim Kazemifard and Vahid Shah-Mansouri. Minimum delay function placement and resource allocation for open ran (o-ran) 5g networks. *Computer Networks*, 188:107809, 2021.

[4] Dariusz Wypiór, Mirosław Klinkowski, and Igor Michalski. Open ran—radio access network evolution, benefits and market trends. *Applied Sciences*, 12(1):408, 2022.

[5] Leonardo Bonati, Salvatore D'Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. Intelligence and learning in o-ran for data-driven nextg cellular networks. *IEEE Communications Magazine*, 59(10):21–27, 2021.

[6] Jason Boswell and Scott Poretsky. Security considerations of open ran. *Stockholm: Ericsson*, 2020.

[7] Michelle Lo David Martin. key benefits of open ran. *https://stlpartners.com/articles/telco-cloud/four-benefits-of-open-ran-and-whether-open-matters/.*, 2017.

[8] Andres Garcia-Saavedra and Xavier Costa-Perez. O-ran: Disrupting the virtualized ran ecosystem. *IEEE Communications Standards Magazine*, 5(4):96–103, 2021.

[9] Leonardo Bonati, Michele Polese, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Open, programmable, and virtualized 5g networks: State-of-the-art and the road ahead. *Computer Networks*, 182:107516, 2020.

[10] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Understanding o-ran: Architecture, interfaces, algorithms, security, and research challenges. *arXiv preprint arXiv:2202.01032*, 2022.

[11] O-RAN Alliance. O-ran architecture overview. *https://docs.o-ran-sc.org/en/latest/architecture/architecture.html*, 2021.

[12] Sameer Kumar Singh, Rohit Singh, and Brijesh Kumbhani. The evolution of radio access network towards open-ran: Challenges and opportunities. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, 2020.

[13] O-RAN Alliance. O-ran.wg2.non-rt-ric-arch-ts-v01.00. *https://orandownloadsweb.azurewebsites.net/specifications*, 2022.

[14] O-RAN Alliance. O-ran.wg3.ricarch-v02.01 o-ran working group 3 near-real-time ran intelligent controller near-rt ric architecture. *https://orandownloadsweb.azurewebsites.net/specifications*, 2022.

[15] O-RAN Alliance. O-ran.wg3.e2gap-v02.01 o-ran working group 3 near real time ran intelligent controller architecture and e2 general aspects and principles. *https://orandownloadsweb.azurewebsites.net/specifications*, 2022.

[16] O-RAN Alliance. O-ran-wg2.a1.gap, "o-ran working group 2, a1 interface: General aspects and principles. *https://orandownloadsweb.azurewebsites.net/specifications*, 2022.

[17] O-RAN Alliance. O-ran architecture overview. *https://docs.o-ran-sc.org/en/latest/architecture/architecture.html*, 2021.

[18] Connor Craven. What are the open ran standards? *https://www.sdxcentral.com/5g/ran/definitions/what-are-open-ran-standards/*, 2021.

[19] Dudu Mimran, Ron Bitton, Yehonatan Kfir, Eitan Klevansky, Oleg Brodt, Heiko Lehmann, Yuval Elovici, and Asaf Shabtai. Evaluating the security of open radio access networks. *arXiv preprint arXiv:2201.06080*, 2022.

[20] Kubernetes. Viewing pods and nodes. *https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/*, 2021.

[21] Kuberentes community. Kuberenetes documents. *https://kubernetes.io/docs/concepts/architecture/nodes/*, 2022.

[22] Kuberentes community. Kuberenetes documents. *https://kubernetes.io/docs/concepts/overview/components/*, 2022.

[23] avi. Understanding kubernetes architecture. *https://geekflare.com/kubernetes-architecture/*, 2020.

[24] Andy Magnusson. Kubernetes role-based access control (rbac). *https://www.strongdm.com/blog/kubernetes-rbac-role-based-access-control: :text=The2022.*

[25] Kubernetes Community. Rbac authorization. https://kubernetes.io/docs/reference/access-authn-authz/rbac/.*, 2022.*

[26] O-RAN Alliance. O-ran.wg1.use-cases-analysis-report v07.00. https://orandownloadsweb.azurewebsites.net/specifications*, 2022.*

[27] *Ahmad Alalewi, Iyad Dayoub, and Soumaya Cherkaoui. On 5g-v2x use cases and enabling technologies: A comprehensive survey.* IEEE Access, *9:107710–107737, 2021.*

[28] *Mohammad Raashid Ansari, Jean-Philippe Monteuuis, Jonathan Petit, and Cong Chen. V2x misbehavior and collective perception service: Considerations for standardization.* arXiv preprint arXiv:2112.02184*, 2021.*

[29] *TS ETSI. 102 637-2 v1. 2.1 (2011-03)-intelligent transport systems (its).* Vehicular Communications.

[30] *Ehab Ali, Mahamod Ismail, Rosdiadee Nordin, and Nor Fadzilah Abdulah. Beamforming techniques for massive mimo systems in 5g: overview, classification, and trends for future research.* Frontiers of Information Technology - Electronic Engineering*, 18(6):753–772, 2017.*

[31] *Kubernetes Community. What is kubernetes.* https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.*, 2022.*

[32] *Hoejoo Lee, Jiwon Cha, Daeken Kwon, Myeonggi Jeong, and Intaik Park. Hosting ai/ml workflows on o-ran ric platform. In* 2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE, 2020.*

[33] *Mayank Shah. Docker security : Backdooring images with dockerscan.* https://medium.com/@mayankshah1607/docker-security-backdooring-images-with-dockerscan-ace5ff65bd39*, 2019.*

[34] *Andrew Johnson and RamiJ Haddad. Evading signature-based antivirus software using custom reverse shell exploit. In* SoutheastCon 2021*, pages 1–6. IEEE, 2021.*

[35] *Peter Gutmann. Do users verify ssh keys.* Login*, 36:35–36, 2011.*

[36] *Catalin Cimpanu. 17 backdoored docker images removed from docker hub.* https://www.bleepingcomputer.com/news/security/17-backdoored-docker-images-removed-from-docker-hub/.*, 2018.*

[37] *Panagiotis Mytilinakis. Attack methods and defenses on kubernetes. Master's thesis, 2020.*

[38] *O-RAN Alliance. O-ran security threat modeling and remediation analysis 2.01.* https://orandownloadsweb.azurewebsites.net/specifications*, 2022.*

[39] *Feilong Wang, Yuan Hong, and Jeff Ban. Infrastructure-enabled gps spoofing detection and correction.* arXiv preprint arXiv:2202.05463*, 2022.*

[40] *Christian Vitale, Nikos Piperigkos, Christos Laoudias, Georgios Ellinas, Jordi Casademont, Josep Escrig, Andreas Kloukiniotis, Aris S Lalos, Konstantinos Moustakas, Rodrigo Diaz Rodriguez, et al. Caramel: results on*

a secure architecture for connected and autonomous vehicles detecting gps spoofing attacks. EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–28, 2021.*

[41] Nataša Trkulja, David Starobinski, and Randall A Berry. Denial-of-service attacks on c-v2x networks. arXiv preprint arXiv:2010.13725*, 2020.*

[42] Ning Wang, Long Jiao, Pu Wang, Weiwei Li, and Kai Zeng. Exploiting beam features for spoofing attack detection in mmwave 60-ghz ieee 802.11ad networks. IEEE Transactions on Wireless Communications*, 20(5):3321–3335, 2021.*

[43] O-RAN Alliance. O-ran alliance security focus group. https://www.o-ran.org/announcements*, 2020.*