THE DEFINITIVE GUIDE TO

Open RAN Security

OCTOBER 2022

Rakuten Symphony

TABLE OF CONTENTS

1 | INTRODUCTION

1.1 | Executive summary

Is Open RAN secure? That depends on who you ask. Based on Rakuten Symphony's experience securing the world's largest fully virtualized Open RAN mobile network with Rakuten Mobile in Japan, our answer is unequivocally yes.

To put this recurring question to rest and, more importantly, help guide the industry toward enduring security strategies for next-gen networks being deployed around the world, we have created "The Definitive Guide to Open RAN Security." In this paper we explain the changes from legacy RANs, what they mean and how to solve for a different open security landscape.

1.2 | Open RAN security motivation

Mobile telecom networks are increasingly becoming disaggregated. Operators are using cloud infrastructure and operations to disaggregate software from the hardware it runs upon. Open RAN standards are architecturally disaggregating the radio access network (RAN) by standardizing growing numbers of what were previously proprietary protocols and interfaces. Increased disaggregation allows for more granular control systems, increasing efficiency of execution and diversity of vendor supply chain.

While increasing the accessible programmable surface area of the total system leads to more transparency and opportunity for introspection, it also increases the potential attack surface from a cyber-security perspective.

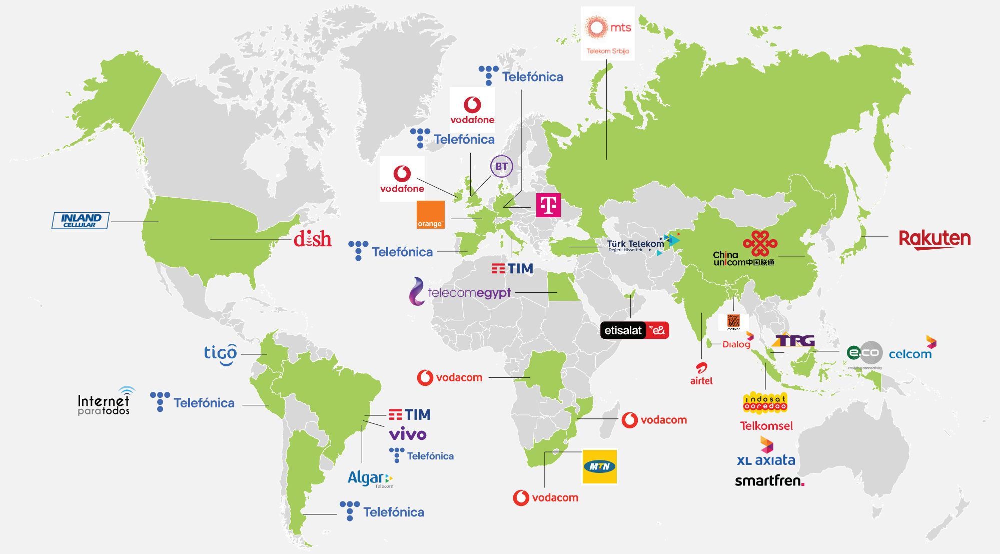
This guide addresses the potential planes of attack and recommended strategies to mitigate risk from a total software system, configuration, and operational point of view. It does not address physical security concerns.

Security doubts shouldn't stand in the way of innovation and transformation. Risk will always be present and demands management, but not avoidance. To compete safely at the speed of cloud, telecom operators should evaluate industry best practices, collaborate with vendors and peers, and innovate, to set the best security and privacy strategies based on individual regulatory and market context.

1.3 | Open RAN market status

Open RAN and associated cloudification of telecom networks are trends that are experiencing accelerating adoption. Both trends mirror those seen in the broader technology transformation journey being seen in all industries. In July 2022, Dell'Oro revised upward near-term Open RAN and vRAN (radio cloud) projections to reflect the higher starting point improving momentum. Global cumulative Open RAN revenues – radio and baseband excluding services – are now projected to approach $20 billion over the next five years. and account for around 15 percent of the 2026 RAN market [1]. As of September 19th 2022, the Open RAN specification has been approved as an ETSI specification[2]. Activity exists on all continents, and all major vendors have announced that their portfolios are Open RAN ready.

The adoption of Open RAN and cloud principles appear to be inevitable for both existing public mobile network operators and emerging private network entrants. All industries are digitalizing and moving to higher automation, software-based cloud solutions and operations, where components are being disaggregated to create more optionality and opportunity to control. Because telecom is considered national critical infrastructure, the only question that remains is when the full transition happens. From the security perspective, the task is to ensure the networks are cyber security ready at launch, from a continuous threat prevention and threat detection point of view. It's also important that the industry is ready and prepared to continuously improve its threat management capabilities to match the evolution of the threats and associated actors who are always growing in sophistication and capability.



TIP Insights 2021: Accelerating Commercial Realities [3].

Global cumulative Open RAN revenues – radio and baseband excluding services – are now projected to approach $20 B over the next five years and account for around 15 percent of the 2026 RAN market.

2 | Open RAN assets and threat landscape

Securing the RAN is paramount to protecting customer private information as well as upholding the availability of national critical communication infrastructure.

The RAN is composed of several components: The base blueprint of all 5G networks (including non-Open RAN) is defined by 3GPP. In 3GPP 5G RAN, the base station (gNB) is split into two logical functions: the centralized unit (CU) and the distributed unit (DU). Both may be virtualized (vDU, vCU) and these two entities are connected by F1-C and F1-U interfaces.

The 3GPP base blueprint leaves many choices to each individual vendor. For example, the interface between PHY and RF layers. Different vendor implementations might therefore be incompatible with one another.

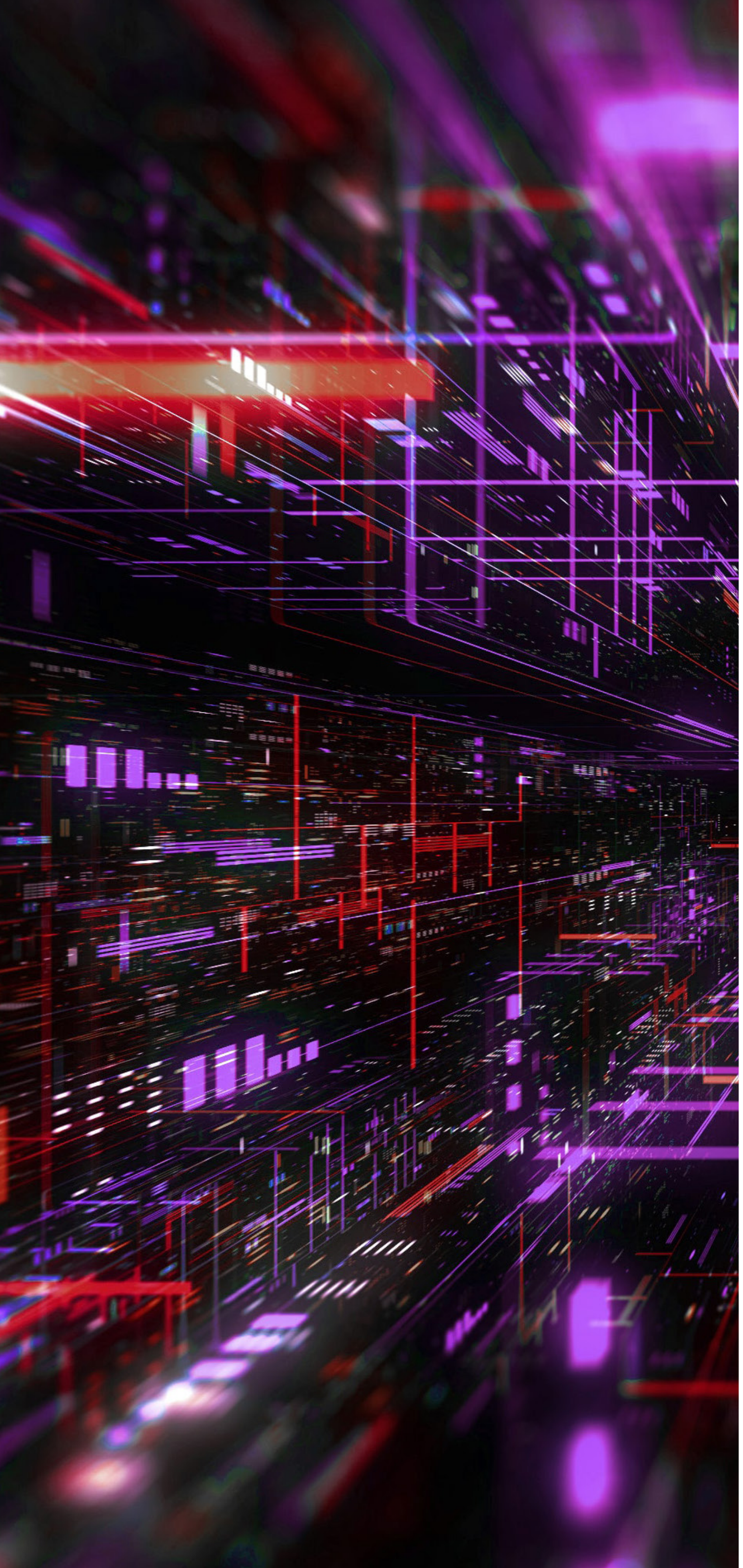
The O-RAN Alliance fills this specification gap. O-RAN Alliance is a group of leading vendors and operators creating RAN specifications that expand on the 3GPP base blueprint. The O-RAN Alliance specifications further disaggregate the CU and DU network functions [19] and introduce other network functions, such as near-real-time radio intelligent controllers (RIC) and non-real-time RIC, and service management and orchestration (SMO), which are interconnected over additional open interfaces (E1, E2, O1, O2, A1), and utilize 3rd party applications (3rd party X Apps).

The O-RAN Alliance is defining security best practices for these additional interfaces and components. In particular, the alliance helps test networks for security, which is simpler than testing individual vendor's bespoke interfaces, functions, and security controls.

2.1 | Open RAN assets

Open RAN encompasses additional assets that need to be protected [9], [19].

- **New infrastructure.** Cloud computing platforms comprising a collection of physical infrastructure nodes that meet O-RAN requirements for hosting O-RAN functions such as Near-RT RIC, O-CU-CP, O-CU-UP, and O-DU, the supporting software components (operating system, virtual machine monitor, container runtime, etc.) and their orchestration functions.
- **New network functions.** O-RAN network functions and applications such as SMO, non-RT RIC and rApps, near-RT RIC and xApps.
- **New interfaces.** O-RAN interfaces including A1 interface between non-RT RIC and near-RT RIC to enable policy-driven guidance of near-RT RIC applications/functions, and to support AI/ML workflow.
- **Critical Data.** Examples of critical assets that need to be protected within the O-RAN system include O-cloud provisioning information, AI/ML data, sensitive data on open fronthaul, security private keys, X.509 certificates, credentials, security event log files etc.



2.2 | Open RAN threats

The O-RAN architecture introduces new infrastructure, functions, and interfaces. These additions, and the decoupling of hardware and software, can expand the attack surface if matching security controls are missing. NIST [4] defines threat as follows:

"A threat is any circumstance or event with the potential to adversely impact organizational operations and assets, individuals, other organizations, or the nation through an information system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service. Threat events are caused by threat sources. A threat source is characterized as: (i) the intent and method targeted at the exploitation of a vulnerability; or (ii) a situation and method that may accidentally exploit a vulnerability."

NIST [4] defines vulnerability as "Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source."

In an Open RAN network, vulnerabilities usually arise due to:

a.) misconfiguration or poorly configured O-RAN components e.g., lack of isolation between network functions in the form of micro-segmentation or not implementing a policy of least privilege access etc.

b.) lack of appropriate access controls in management of O-RAN components

c.) lack of authentication on Open RAN interfaces, such as unauthenticated access to open fronthaul networks

d.) not following best practices for container images including vulnerabilities in images and libraries included

Attackers exploit these vulnerabilities to inflict damage on the infrastructure. Some of the common threats include:

a.) An attacker exploits insufficient/improper mechanisms for authentication and authorization in management systems to compromise O-RAN components;

b.) an attacker exploits un-authenticated/un-authorized access to open fronthaul Ethernet L1 physical layer interface(s) to masquerade as a PTP master node and impact timing on the Open Front Haul interface;

c.) an attacker uses vulnerabilities in the deployed workload to escape from the container and mount lateral attacks;

d.) an attacker exploits misconfiguration in Kubernetes and container security to escalate its privileges and cause damage to the host.

A more detailed list of common vulnerabilities that may be exploited by threat sources is covered in Appendix 14.1.

2.3 | Security principles for mitigating Open RAN threats

The impact of threats can be minimized when a network is built based on a well-established set of security principles. Security principles provide a high level and abstract statement of the intended solution to countering potential threats and serves as the foundation for a secure Open RAN network.

In this section, we define a consolidated set of security principles that form the basis for Rakuten Symphony's secure Open RAN architecture.

1. Mutual authentication SHOULD be established between communicating entities in an O-RAN system, with each entity identified by a unique identifier and credentials.
2. Access control SHOULD be implemented that only allows authenticated and authorized personnel and services to access O-RAN resources or services.
3. The principle of least privilege (PoLP) SHOULD be followed to ensure that accounts have the least amount of privilege required to perform their business processes.
4. Measures SHOULD be taken to provide a secure and trusted runtime environment for cloud applications by implementing security controls that reduce the risk of successful firmware exploitation and impact of many published early-boot vulnerabilities when bootstrapping a cloud native platform.
5. Domain separation or security zones SHOULD be implemented to group systems and resources that have similar needs for information protection, access controls and other security requirements.
6. Security controls SHOULD ensure that lateral movement is detected and prevented when attackers have successfully exploited a vulnerability to gain initial access into a 5G cloud system
7. The system SHOULD ensure protection of data-at-rest, data-in-transit, and data-in-use according to industry best practices.
8. The O-RAN system SHOULD be bootstrapped to be secure by default and it should be up to the user to reduce their security – if they are allowed.
9. Industry best security practices SHOULD be followed when using open source components, to minimize risks.

2.4 | Rakuten Symphony Security Architecture for Open RAN

When building a mobile network according to only 3GPP and O-RAN Alliance specifications, some security-relevant areas remain unclear. Therefore, mobile network vendors and operators need to apply additional security controls to reach an appropriate level of hacking resilience.

Rakuten Symphony has adopted several requirements that influence security architecture for Open RAN:

REQ-1: An Open RAN system shall implement a unified identity, credential, and access management (ICAM) based on zero-trust network access principles, to securely connect different types of subjects such as human user, services, and device access, to a variety of objects including cloud native platforms, services, and network devices.

REQ-2: API and API endpoints shall be secured using strong authentication and authorization mechanisms such as OAuth 2.0 and OpenID Connect.

REQ-3: A policy-driven architecture that can dynamically react to events and securely configure systems and networks on-demand shall be implemented.

REQ-4: An Open RAN system shall be based on a secure, cloud native hardware platform that uses hardware capabilities to establish chain of trust in a cloud native platform.

REQ-5: An Open RAN system shall provide a secure cloud native software infrastructure that is provisioned with necessary security controls and hardened as per industry best practices including CIS configuration benchmarks.

REQ-6: An Open RAN system shall be configured to protect workloads from external (north-south) and lateral attacks (east-west traffic).

REQ-7: Storage of data shall be decoupled from utilization and sensitive data is encrypted at rest.

REQ-8: Open RAN system shall adopt DevSecOps and secure SDLC principles for building, distributing, and deploying container images, configurations, and policies.

REQ-9: Implementing container image security through build processes, scanning, and using runtime security measures to detect and prevent malicious activity during container execution.

The remainder of this guide is focused on security solutions that help to meet the architectural requirements and is structured as follows:

- Chapter 4 introduces key security features and capabilities that are required to secure a Kubernetes-based cloud native platform used in an Open RAN to host network functions.
- Chapter 5 explores areas that relate to providing secure communication between all network functions in Open RAN including establishing trust based on zero-trust provisioning of O-DU and O-RU.
- Chapter 6 provides insights on how to secure the OAM system used to manage Open RAN deployments.
- Chapter 7 looks at how container security is implemented across its lifecycle.

3 | Secure Cloud Native Platform (CNP) for Open RAN network functions

The O-RAN Alliance RAN architecture is built on a fully cloud native architecture (see Figure 1) – the same cloud architecture that is the bedrock of today's internet and public cloud. The cloud native network functions in the O-RAN network – O-CU-CP, O-CU-UP, O-DU, Near-RT RIC, and Non-RT RIC – are hosted on a cloud native platform, very similar to the cloud native platform used in the cloud computing industry.

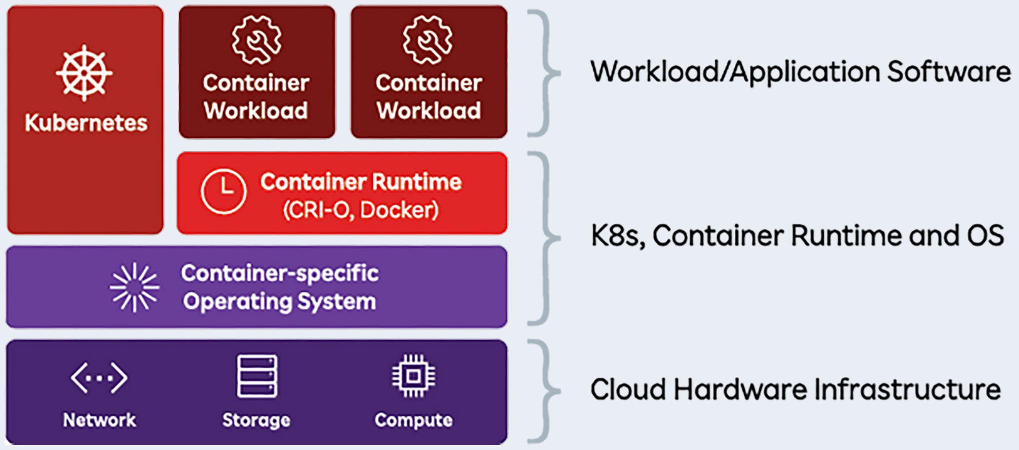

Kubernetes
Container Workload
Container Workload
Workload/Application Software
Container Runtime (CRI-O, Docker)
Container-specific Operating System
K8s, Container Runtime and OS
Network
Storage
Compute
Cloud Hardware Infrastructure

Figure 1: Cloud Native Platform

In the following sections we explore the following aspects of securing a cloud native platform:

a.) Establish end to end chain of trust with Secure Boot.

b.) Authenticated and authorized access to the platform via K8s API server.

c.) Securing internal communication between cloud native platform control plane components.

d.) Providing a secure storage for critical/sensitive data.

e.) Using infrastructure as code (IaC) to manage configuration drifts.

f.) Policy based management for platform security.

g.) Using SR-IOV techniques for isolation at network layer.

3.1 | Establishing end-to-end chain of trust with secure boot

This addresses requirement REQ-4. Firmware is stored and executes from memory that is separate from the operating system and storage media. Antivirus software, which runs after the operating system has loaded, is ineffective at detecting and remediating malware in the early-boot firmware environment that executes before the operating system. Secure Boot is an industry standard that provides a validation mechanism that reduces the risk of successful firmware exploitation and mitigates many published early-boot vulnerabilities.

Secure Boot is an important security feature designed to prevent malicious firmware such as UEFI firmware for e.g, from loading when the platform starts up (boots). Secure Boot necessitates that every boot up begins with a piece of software that cannot be updated in the field. This piece of software is known as core root of trust for measurement (CRTM).

Following that, every software program in the platform will be integrity-verified during the boot process before being executed by the software at the lower layer. This establishes an end-to-end software chain of trust.
The trust anchor of the software integrity verification is a software signing certificate. Rakuten Symphony recommends using the Secure Boot based on a hardware-based root of trust such as TPM.

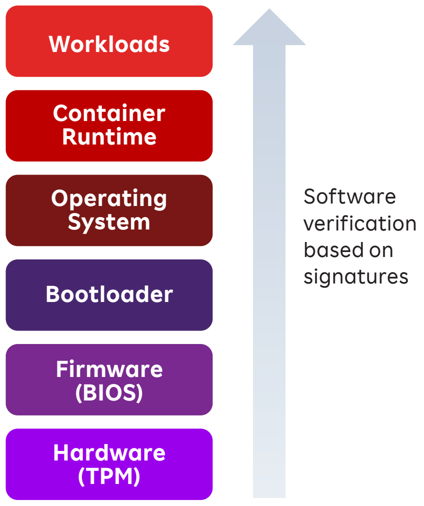


Figure 2: Secure boot

3.2 | Securing access via Kubernetes API server

This clause addresses REQ-1, REQ-2, and REQ-5 in clause 6. A cloud native platform uses container orchestration software, such as Kubernetes, to automate the deployment, management, scaling, and networking of containers. In Kubernetes, all operations and communications between its components, and external user commands are REST API calls that the API server handles. The API server is the most critical component in Kubernetes control plane, and it is accessed by different clients to perform Kubernetes cluster operations. A secure, authenticated, authorized and admission-controlled access to API server is key to securing the API server and the cluster from malicious actors.

The API server (see Figure 3) services incoming API requests (REST operations) and provides the frontend to the cluster's shared state through which all other components interact.

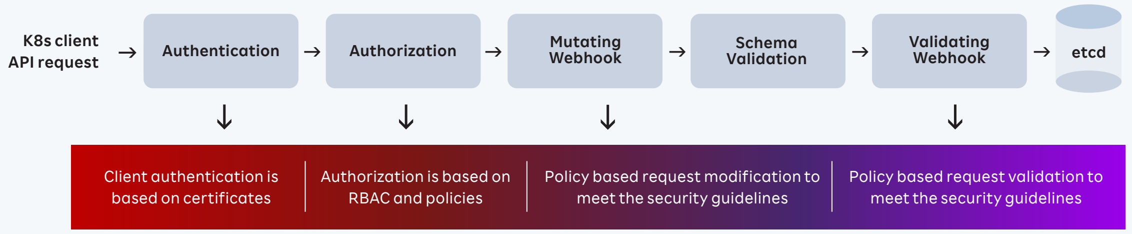

K8s client API request
Authentication
Authorization
Mutating Webhook
Schema Validation
Validating Webhook
etcd
Client authentication is based on certificates
Authorization is based on RBAC and policies
Policy based request modification to meet the security guidelines
Policy based request validation to meet the security guidelines

Figure 3: Securing access to a cloud native platform via Kubernetes API server.

The cloud resource requests (API requests) go through authentication, authorization and admission control gates [8] before getting written to the object store in an etcd database.

The authentication is performed using either the client certificates or Open ID Connect ID tokens provided by an IdP or service account tokens. The mounting of default service account tokens to workload instances is disabled to provide access to API server from workloads on need basis and not by default.

The next gate is authorization, which is implemented using role-based access control (RBAC). RBAC contains additional security constructs apart from the standard Kubernetes RBAC, where a “tenant” can be defined, and different named users / service accounts can be associated with the tenants as a discrete group. A tenant user will be allowed access to only those system resources (storage, CPUs, and memory) which are assigned to their respective tenant.

The final gate is admission control, which is performed using deployment-specific policy definitions. The policy definitions or framework provides dynamic control over the cloud resource management and gives the flexibility to allow exceptions by an authorized cluster administrator, e.g., during a cluster’s maintenance.

3.3 | Securing Kubernetes control plane communications

This clause addresses requirements REQ-1, REQ-2, and REQ-5 in clause 6. Unsecured communication mechanism for Kubernetes control plane components will increase the attack surface area and allow an attacker to disrupt/take control of the cluster and use these clusters for malicious activities. It is essential to secure the control plane communication using secure/trusted protocols.

As depicted in Figure 4, in Kubernetes, various control plane components talk to each other to monitor and maintain the desired state of the system and micro services running as containers in its nodes. By default, not all these control plane components use secure protocols to communicate with each other.

As depicted in Figure 4, communication between various control plane components can be secured using TLS certificates [9]. A cluster level certificate authority (CA) server or an external CA server can be used to generate required entity-level certificates. This allows communicating entities to establish trust based on mutual TLS and use TLS capabilities to secure the communication. This ensures integrity, confidentiality of the messages exchanged amongst different control plane components, messages exchanged between workloads and control plane component.

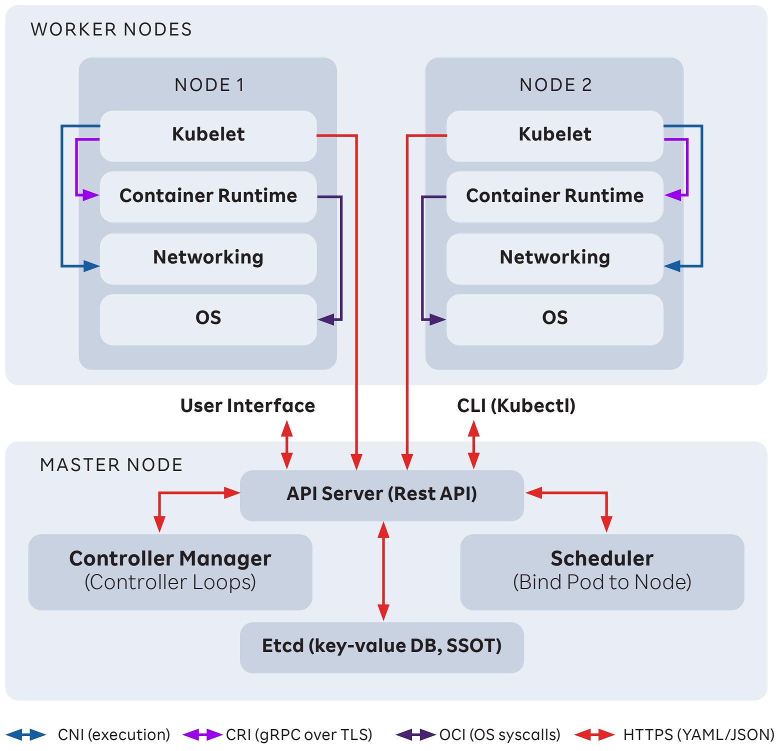


Figure 4: Securing Kubernetes control plane communications

3.4 | Securing critical data at rest

This clause addresses the REQ-7 requirement in clause 6. The critical data such as cryptographic keys, passwords, subscriber information and others must be protected and should be accessible only when explicit permissions are provided to a user or a service.

There are several mechanisms that can be used to secure data at rest

a.) Using an API server's underlying datastore, etcd

b.) Using Vault to secure critical data

The resource data stored in Kubernetes etcd is not encrypted by default. The data (including sensitive information such as secrets) can be read by any user who has access to perform kubectl exec commands on the Kubernetes etcd POD or any user who has 'get' access to Kubernetes secret resources. The secrets in Kubernetes are base64 encoded without any encryption by default.

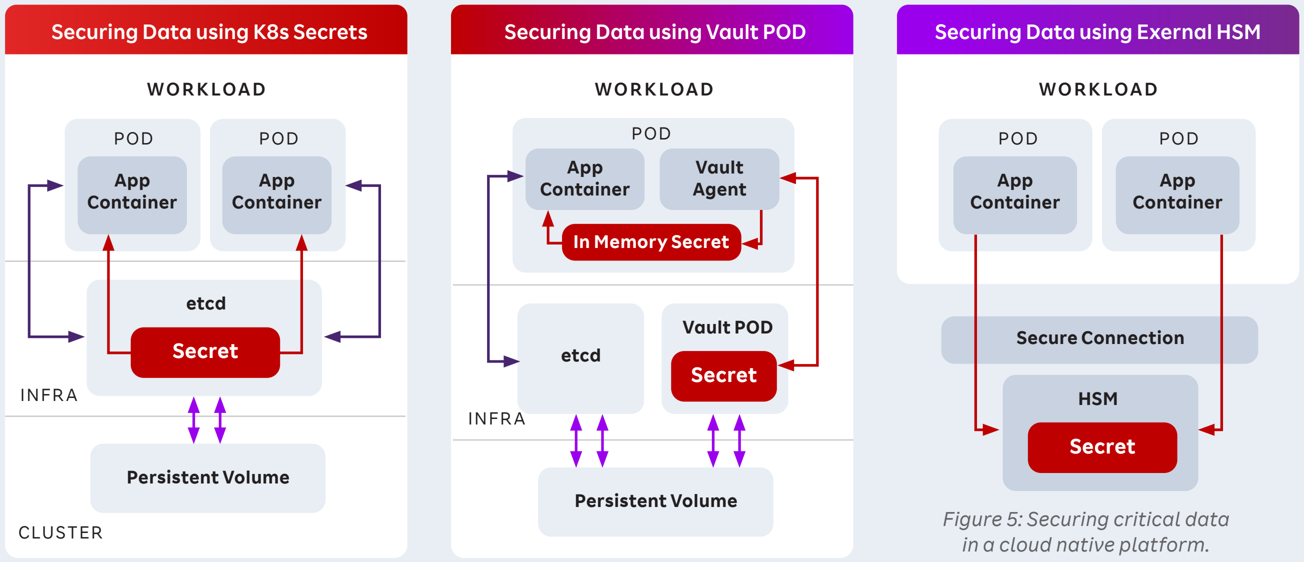


Figure 5: Securing critical data in a cloud native platform.

Figure 5 shows three options for data security. One of the mechanisms that can be used for securing the data is to deny users / service accounts access to secrets or other data, following the principle of least privilege. This can be achieved using Kubernetes RBAC or other authorizers to deny the ability to perform get and create verbs on resources. Encryption of etcd data could also be enabled to secure secret resources stored in etcd.

A Vault POD could be used for securing data where sensitive information, including tokens and keys, is stored in a Vault instance within the cluster or in a centralized vault stored in a secure location. Other POD data will be stored in etcd. Both Vault and etcd would then use the configured persistent volume (PV) as backend storage. Vault ensures proper access control mechanisms and confidentiality protection for the sensitive information using a Vault agent and making the information available only to trusted applications and user identities using shared memory.

3.5 | Managing configuration drifts using Infrastructure as Code (IaC)

This clause addresses REQ-8 in clause 6. A declarative way of infrastructure provisioning simplifies automation and reduces human errors. IaC is a vital concept that is implemented by orchestrators for easier, flexible, and faster deployments of different types of applications.

Infrastructure as Code (IaC) is an approach to managing and provisioning of cloud native infrastructure through code instead of through manual processes. Automating infrastructure provisioning with IaC means that developers don't need to manually provision and manage servers, operating systems, storage, and other infrastructure components each time they develop or deploy an application. Apart from the various benefits that the IaC approach provides, it helps in reducing errors in configuration and avoiding configuration drift.

3.6 | Policy-based governance for Kubernetes using OPA gatekeeper

This clause addresses REQ-3 in clause 6. Dynamic policy-based control of Kubernetes cluster management provides the flexibility to take status of the infrastructure / clusters / workflows into account and respond to the needs of cluster management / access using well defined rules (REQ-3).

Kubernetes supports three phases namely authentication, authorization, and admission control through which an API server request must pass before the changes requested are committed to the Kubernetes control plane database (etcd).

Open Policy Agent (OPA) for admission control in Kubernetes (see Figure 6) allows dynamic control using webhooks for validation / mutating phases of admission control. The Kubernetes gatekeeper module supports adding OPA-based Constraint Template resources, with embedded OPA Rego policies at runtime to the Kubernetes cluster. These constraint templates enable adding constraints that validate incoming requests to the API server and allow / reject the request based on the constraint.

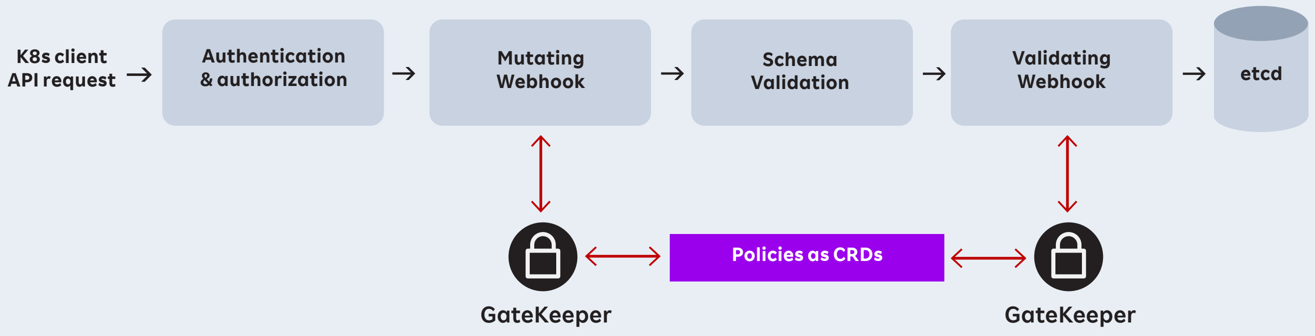


Figure 6: OPA gatekeeper for policy-based governance of Kubernetes

The policies embedded in the OPA gatekeeper CRDs can contain complex policy logic to cater to multiple use cases of admission control. Some of the example use cases of policies are described below:

- Do not allow containers that are privileged, or which can perform privilege escalation.
- Based on exception allow only certain capabilities (e.g., raw packet capture) to certain containers.
- Limit resources to a specific namespace to avoid service denial scenarios if a container within this namespace is compromised.
- Restrict giving access to Kubernetes resources by performing checks on Kubernetes role / role-binding creation requests.

The gatekeeper enables writing policies to take input in the standard Kubernetes format and these policies, once written, can be re-used across multiple clusters with cluster specific information provided to the policies as data input at runtime.

3.7 | Strong isolation of I/O devices using single root-I/O virtualization (SR-IOV)

This clause addresses REQ-6 in clause 6. Virtualization of a network I/O device will help to optimally utilize the device, isolate the different types of network traffic for controlling resource limits, and enable usage of dedicated network devices for fast path packet processing.

SR-IOV allows the mapping of multiple virtual functions (VF) to a physical function (PF), where each mapping gets its own specific configuration. Along with providing a definite performance improvement over using the infrastructure host, SR-IOV allows the isolation of traffic between the different VFs and can provide a level of security for the different traffic streams.

4 | Establishing trust between Open RAN network functions

Open RAN network functions provide communication services to subscribers and contain highly sensitive subscriber data that should be protected from leaks to ensure privacy and confidentiality. Open RAN specifications recommend, and in some cases mandate, controls to secure the interfaces between different logical network functions. Apart from securing data in transit on the interfaces, it is important to secure important information including configuration, AI/ML models, logs with sensitive information, etc.)

The following security aspects are considered when evaluating trust between network functions in Open RAN.

a.) Secure communication on Open RAN interfaces

b.) Securing Open Fronthaul Interface using IEEE 802.1x

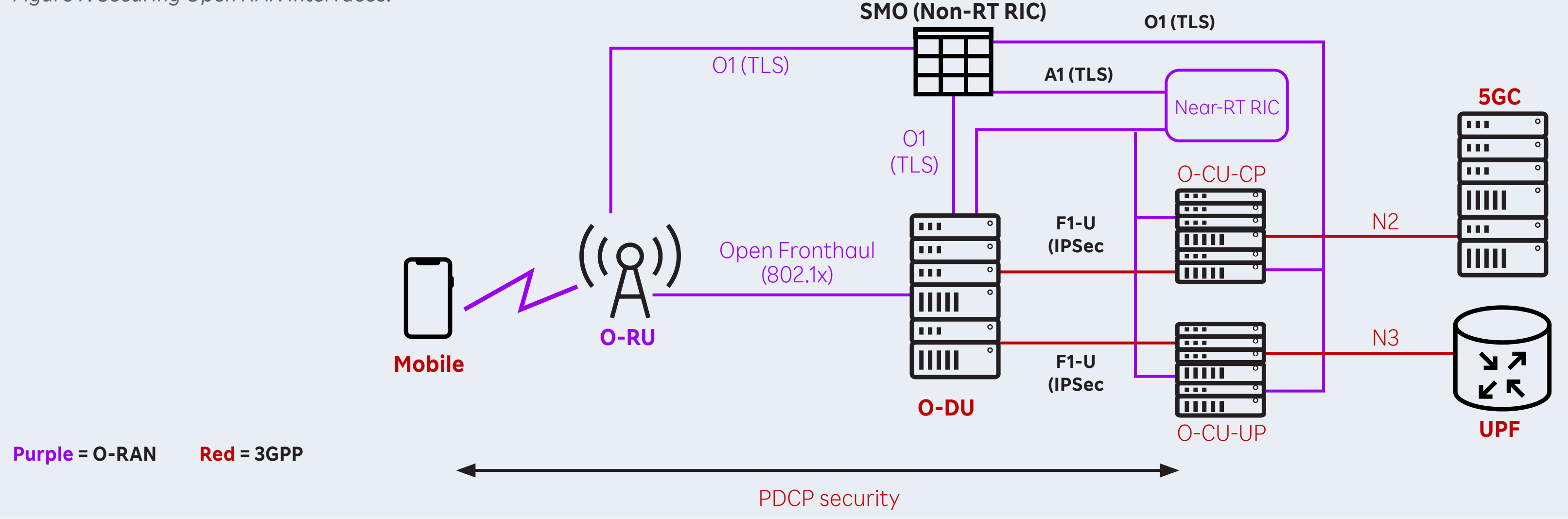
c.) Securing non-RT RIC and near-RT RIC functions

d.) Zero-trust provisioning for O-DU and O-RU

4.1 | Secure communication on Open RAN interfaces

The O-RAN Alliance specifies an open and secure architecture (see Fig. 7) that includes secure interfaces between all its components. Communications exchanged on these interfaces are cryptographically protected for encryption, integrity protection and replay protection.

Figure 7: Securing Open RAN interfaces.



Interface	Between Nodes	Security mechanism	Specified by
E1	O-CU-CP and O-CU-UP	NDS/IP (IPSec) or DTLS	3GPP
Xn	Source gNB and Target gNB	NDS/IP (IPSec) or DTLS	3GPP
Backhaul	O-CU-CP and 5GC (N2) O-CU-UP and 5GC (N3)	NDS/IP (IPSec) or DTLS	3GPP
Midhaul (F1)	O-CU-CP and O-DU (F1-C) O-CU-UP and O-DU (F1-U)	NDS/IP (IPSec) or DTLS	3GPP
Open Fronthaul (M-Plane)	O-RU and O-DU/SMO	mTLS, SSHv2	O-RAN WG4
Open Fronthaul (CUS-Plane)	O-DU and O-RU	IEEE 802.1x with EAP-TLS	O-RAN WG11
O1	SMO and O-RAN Managed elements	mTLS	O-RAN WG11
E2	Near-RT RIC (xAPPs) and O-CU-CP	NDS/IP (IPSec) or DTLS	O-RAN WG11
A1	Near-RT RIC and Non-RT RIC	mTLS	O-RAN WG11
O2	SMO and O-Cloud	Work in progress (2022)	O-RAN WG11
r/xAPPS	Non/Near-RT RIC	Work in progress (2022)	O-RAN WG11

Figure 8 summarizes the protection mechanism used for each interface in an O-RAN-based network.

Figure 8: Securing Open RAN interfaces using industry standard protocols.

4.2 | Securing Open Fronthaul using IEEE 802.1x authentication

Device authentication is of paramount importance in Open vRAN where only authenticated devices should be allowed access to the operator's network. IEEE 802.1x-2020[5] provides the means to ensure that only authenticated devices can access the network, thereby ensuring that malicious actors are unable to enter and threaten the operator network.

802.1x protocol works based on Extensible Authentication Protocol (EAP) which in turn makes use of different schemes to achieve authentication. One of the most secure schemes is the use of Transport Level Security TLS within the EAP (EAP-TLS) protocol whereby authentication is achieved using a X.509 certificate. The O-RAN Alliance specifies use of EAP-TLS for 802.1x based authentication on the Open Fronthaul interface.

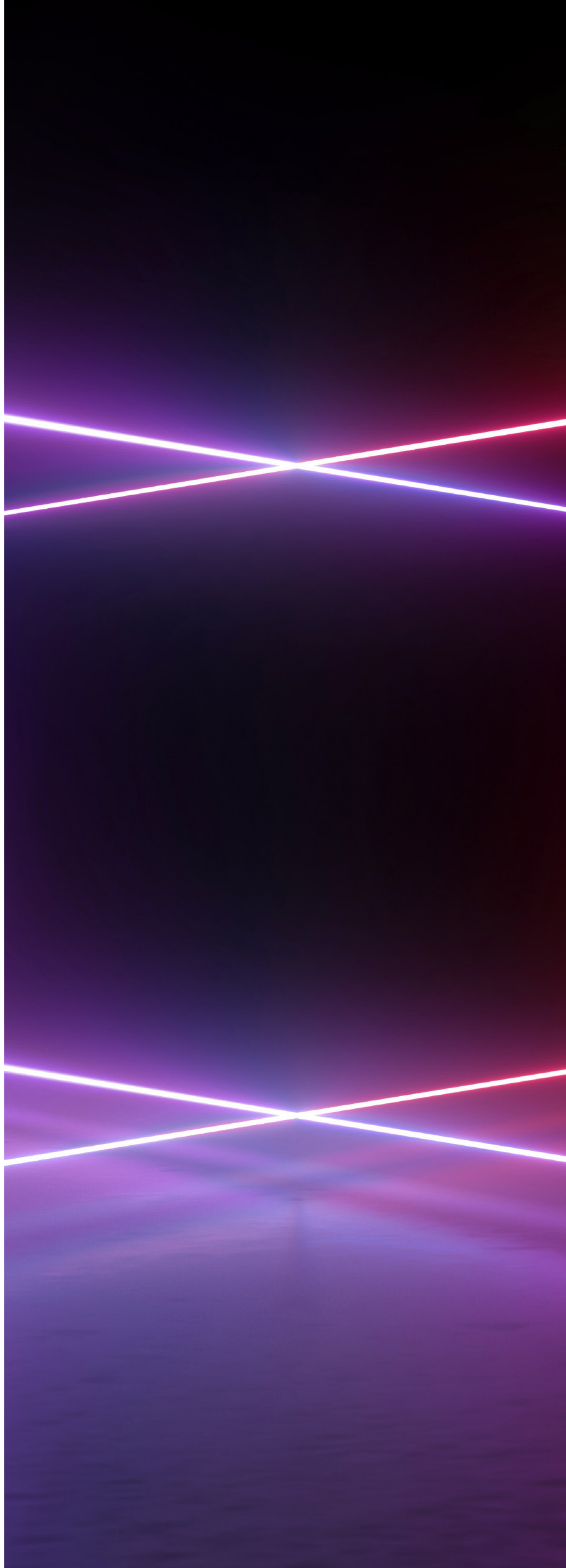
4.3 | Securing Non-RT RIC and Near-RT RIC Functions in SMO

RAN intelligent controllers play an important role in the optimized performance and responsiveness of radio access networks. The control loops in RIC network functions operate under strict latency requirements and provide interfaces to rApps / xApps for getting RAN metrics and enrichment information. The RICs implement AI/ML workflow of model learning and inference for optimization of RAN resources. Securing the different RIC interfaces, data processed, and policies stored and transferred are the keys to avoiding outages and data leaks / thefts.

The RIC framework enables onboarding third-party xApps/rApps, the apps are automation tools that provide business logic to use real-time RAN metrics to derive policies for RAN operational efficiency and generate enrichment information. The third-party xApps / rApps are authenticated and only authorized access is provided to RIC interfaces.

AI/ML workflow involves processing of a lot of data to train an ML model. The integrity, and in some cases, confidentiality of the data should be protected in the training phase to allow access to data for only the services that are authenticated and authorized. The generated ML models should also be integrity protected to avoid model poisoning attacks. During the inference phase, the stream of input data provided to the model should also be protected for integrity and confidentiality to avoid wrong inferences by manipulation of data by malicious actors. For online ML algorithms for which the model hyperparameters are tuned during runtime, more stringent checks and balances should be applied on the model input data as model drift can be achieved by manipulating input data.

As recommended by the O-RAN specifications, the interfaces between RIC and other network functions should be protected using the appropriate security controls (IPsec for E2 interface, TLS for O1 and A1 interfaces).



4.4 | Establishing trust based on zero-touch provisioning for O-DU and O-RU

O-RUs and O-DUs are introduced to the network based on automated mutual authentication, confidentiality and integrity protection using X.509 certificates (see Figure 9).

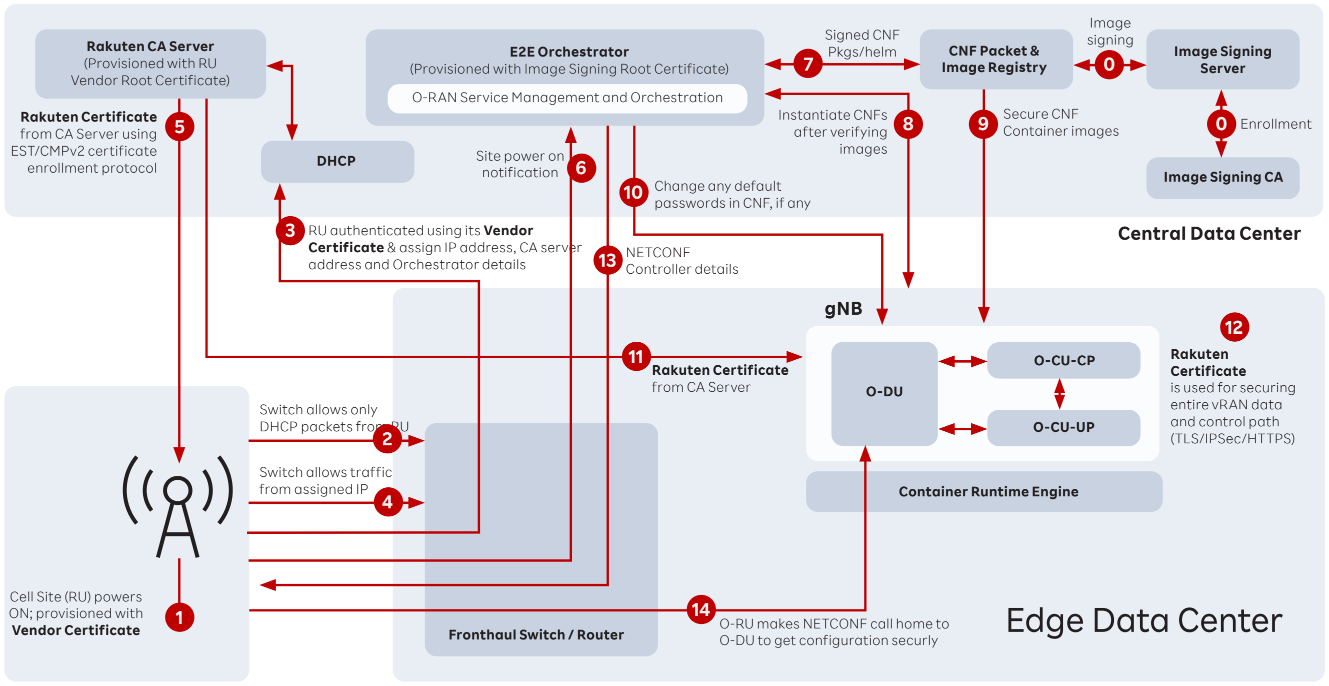

Rakuten CA Server (Provisioned with RU Vendor Root Certificate)
E2E Orchestrator (Provisioned with Image Signing Root Certificate)
O-RAN Service Management and Orchestration
Signed CNF Pkgs/helm
7
CNF Packet & Image Registry
Image signing
0
Image Signing Server
0
Enrollment
Image Signing CA
Rakuten Certificate from CA Server using EST/CMPv2 certificate enrollment protocol
5
DHCP
Site power on notification
6
Instantiate CNFs after verifying images
8
9
Secure CNF Container images
10
Change any default passwords in CNF, if any
Central Data Center
3
RU authenticated using its Vendor Certificate & assign IP address, CA server address and Orchestrator details
13
NETCONF Controller details
gNB
12
11
Rakuten Certificate from CA Server
O-DU
O-CU-CP
O-CU-UP
Rakuten Certificate is used for securing entire vRAN data and control path (TLS/IPSec/HTTPS)
Container Runtime Engine
Switch allows only DHCP packets from RU
2
Switch allows traffic from assigned IP
4
Cell Site (RU) powers ON; provisioned with Vendor Certificate
1
14
O-RU makes NETCONF call home to O-DU to get configuration securly
Fronthaul Switch / Router
Edge Data Center

Figure 9: Zero touch provisioning of O-RU and O-DU.

The different steps involved in zero touch provisioning (ZTP) when an O-RU is newly commissioned are:

- Step-1: The O-RU containing O-RU vendor certificates, is powered on at the site. The O-RU is connected to the fronthaul router or switch before power-on.
- Step-2: The O-RU sends out DHCP request packets and the switch **allows only DHCP packets** initially from O-RU and **drops all other types of packets.**
- Step-3: The DHCP server **authenticates** the O-RU using the vendor certificates and then responds back with an IP address, orchestrator and **Rakuten CA enrollment** details.
- Step-4,5: O-RU enrolls with Rakuten CA using EST/CMPv2 procedures and obtains a **Rakuten CA signed certificate.**
- Step-6: O-RU **using Rakuten certificate,** securely notifies the E2E orchestrator using a power-on message containing the O-RU inventory details.
- Step-7, 8, 9: Orchestrator **authenticates** the O-RU and then determines if additional resources (O-DU or O-CUUP) need to be instantiated to serve the O-RU. If instantiation is needed, then, orchestrator **securely** downloads the CNF packages and **verifies their signatures** and instantiates the CNFs.

- Steps 10: Any **default passwords** are changed by orchestrator for the newly instantiated CNFs.
- Step 11, 12: The newly instantiated CNF is provisioned with Rakuten CA server details during instantiation and as part of CNF bootstrap procedure, the **CNF enrolls and obtains a Rakuten CA signed certificate** which is used for mutual authentication and secure transport.
- Step 13, 14: The orchestrator provides the details of NETCONF controller (O-DU) to O-RU. O-RU performs **secure NETCONF call home procedures** to establish a NETCONF session, **obtains O-RU configuration** and then loads the configuration to become operationally active.

5 | Secure management of Open RAN networks

Service management and orchestration (SMO) is the component that oversees all the orchestration aspects, management, and automation of the RAN elements. A typical network management platform comes with several applications that allow network engineers to operate, administer and manage a network's independent components.

When implemented inside a cloud native infrastructure, network management functions such as SMO are implemented as containerized applications that expose APIs to offer various services to clients.

These functions therefore have a different type of attack surface requiring additional security controls for protection. In the following sections we explore several security techniques that provide protection for these management functions.

5.1 | Identity and access management based on zero-trust network access principles

This addresses requirement REQ-1, REQ-3 in clause 6. Rakuten Symphony offers a ZTN-based Identity and access management solution (see Figure 10) that supports identity management, role/permission administration, single-sign-on, strong authentication mechanisms and dynamic policy-driven access control for accessing network infrastructure including cloud native platforms.

The identity and access management (IAM) solution is responsible for provisioning identity and credentials to users, machines and applications, which makes it possible to implement secure communications between user and device, user, and application, and between applications. All kinds of resources, including devices, hosts, containers, services, and applications, are objects to be accessed by subjects.

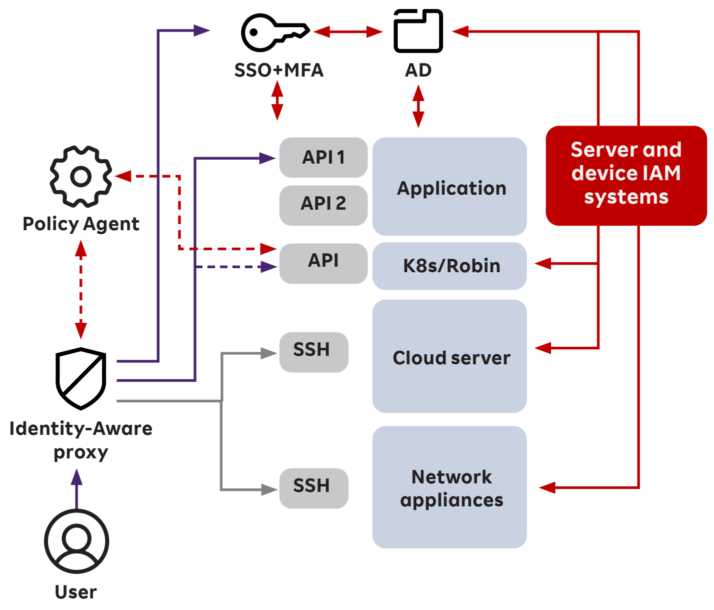


Figure 10: Zero-trust network access for production environment.

Access control decisions are made by a generic authorization gateway, or a proxy based on the identity of the subject, the device identity, and the context of the access, including the location of the requestor, the time of the access, how the connection is made to the resources, and the security posture of the devices. A role-based access control model is used to simplify the management of access permissions.

The policies used to authorize access to different devices can be controlled from a central location. Since, the policies are dynamic in nature and should be generated using input from other systems, a component of the policy infrastructure (e.g., based on OPA) could be used to generate the just-in-time policies that are then pushed to the authorization gateway or the identity aware proxy.

5.2 | Securing API requests with an API gateway

This addresses requirement REQ-2 in clause 6. An API gateway is a service that accepts incoming API requests from clients, directs the request to the appropriate application service, processes that service's response and relays that response to the requesting client (see Fig. 11).

By managing API requests from external systems through a central gateway outside of the application, developers can create a more secure application while also simplifying communication management.

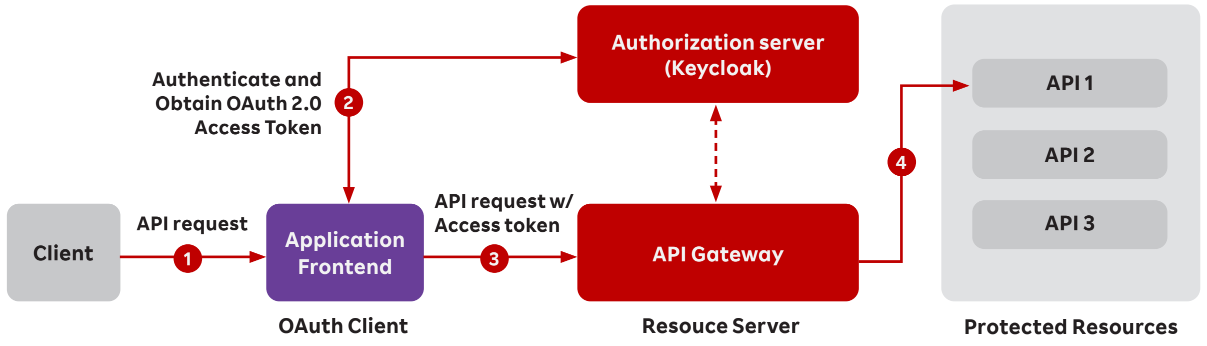


Figure 11: Securing APIs using an API gateway.

OAuth 2.0 [6] and token-based authorization is used to authorize API access by external clients. The application frontend plays the role of the OAuth client. It registers with the authorization server (e.g., Keycloak) with a client ID and a client secret. The API gateway plays the role of the resource server in OAuth, which is therefore responsible for validating the OAuth client before granting it access to the protected API resource.

- Step 1) External requests are received by the application frontend. The application frontend follows OAuth 2.0 Authorization Grant flow to authenticate the client (external) and obtain an authorization code from the authorization server.
- Step 2) The application frontend exchanges the authorization code with an access token and a refresh token authenticating with the authorization server using its client ID and client secret.
- Step 3) The application frontend appends the access token to the API request and forwards it to the API gateway.
- Step 4) The API gateway verifies the access token and forwards the request to the concerned service. It is important to note that verifying the access token ensures that the external client is authorized to perform the API request.

5.3 | Securing service to service communication with service mesh

This addresses requirement REQ-6 in clause 6. The service mesh (see Figure 12) brings the following capabilities to a platform and its constituent services by transparently implementing service-to-service (i.e., east-west traffic) resilience and security, including end-user authorization verification, mutual TLS, service-to-service RBAC/ABAC, service-to-service rate limiting, quota enforcement, etc.

In a typical network management platform, services interact with each other, sometimes over an untrusted interface, for example when they cross a cluster boundary or traverse across untrusted nodes, etc.

In such scenarios, a service mesh can help in implementing and enforcing cross-cutting security requirements, such as providing service identity (via x509 certificates), enabling application-level service / network segmentation (e.g., "service A" can communicate with "service B," but not "service C") ensuring all communication is encrypted (via TLS), and ensuring the presence of valid user-level identity tokens.

The sidecar provided by the service mesh can initiate mutual TLS (mTLS), encrypt service-to-service traffic, and achieve non-repudiation for requests without requiring any changes or support from the applications. This layer of security reduces the likelihood of a successful man-in-the-middle (MitM) attack by requiring all parties in a request to have valid certificates that trust each other.

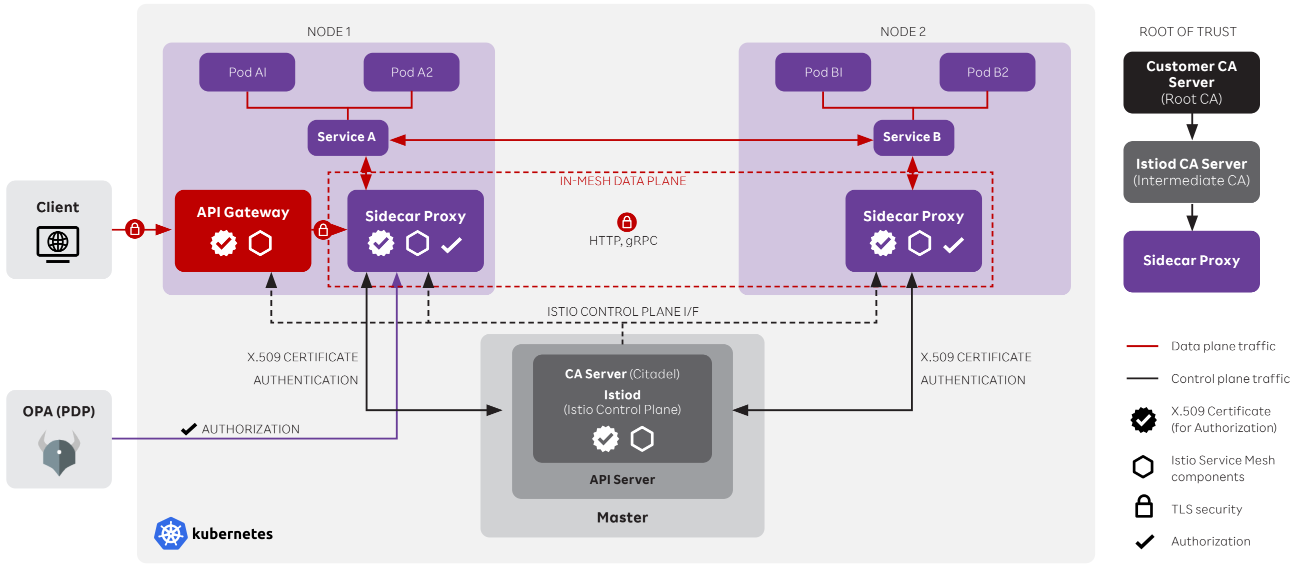


Figure 12: Securing service-to-service communication with service mesh

Allowed user behavior is addressed with role-based access control (RBAC). With these controls, the zero-trust philosophy of "trust no one, authenticate everyone" stays in force by providing enforceable least privilege access to services in the mesh. Role-based access control provides the ability to enforce the principle of least privilege in a cloud native platform.

5.4 | Container isolation using K8s namespaces and network policies

This addresses requirement REQ-6 in clause 6. In some scenarios, a set of services may need to be fully isolated from other services, essentially creating a separate security group or a zone which contains critical services.

In Kubernetes, Namespaces enable isolation and grouping of containers into specific virtual clusters and allowing independent, exclusive security controls to be applied on each namespace / container within the namespace. To allow communication only between well-defined endpoints, by default the ingress and egress traffic to a service could be disabled. And only based on a communication matrix, the ingress and egress rules in the form of network policies could be written to allow the traffic and secure the east / west communication between services.

Network policies limit the extent of damage a compromised service can cause by not allowing unchecked access, especially on the egress interfaces, and preventing the compromised service to be used as a bot to attack other services in the cluster.

Namespaces allow resource limits to be defined, which prevents a service from utilizing all the resources available in the system. Also, critical services can be placed in a different namespace with specific resource limits and non-critical services could be grouped together in another namespace with a combined resource limit for robustness and "blast radius" containment in case of a breach.

5.5 | Centralized policy infrastructure for security management

This addresses requirement REQ-3 in clause 6. Policies to take decisions about security access, especially for authorization requests can be dynamic in nature. These policies are updated dynamically based on different management tasks and network states. A central policy infrastructure can be built where the policies are stored and policy decision points (PDPs) can respond to a policy decision request [10].

The policies can take the network state information and other inputs (such as time of day, maintenance window status, user privileges, exception list) and determine the decision to be taken for each of the incoming policy decision requests. Fig. 13 shows a central policy engine (PDP) catering to multiple requestors also called policy enforcement points (PEPS).

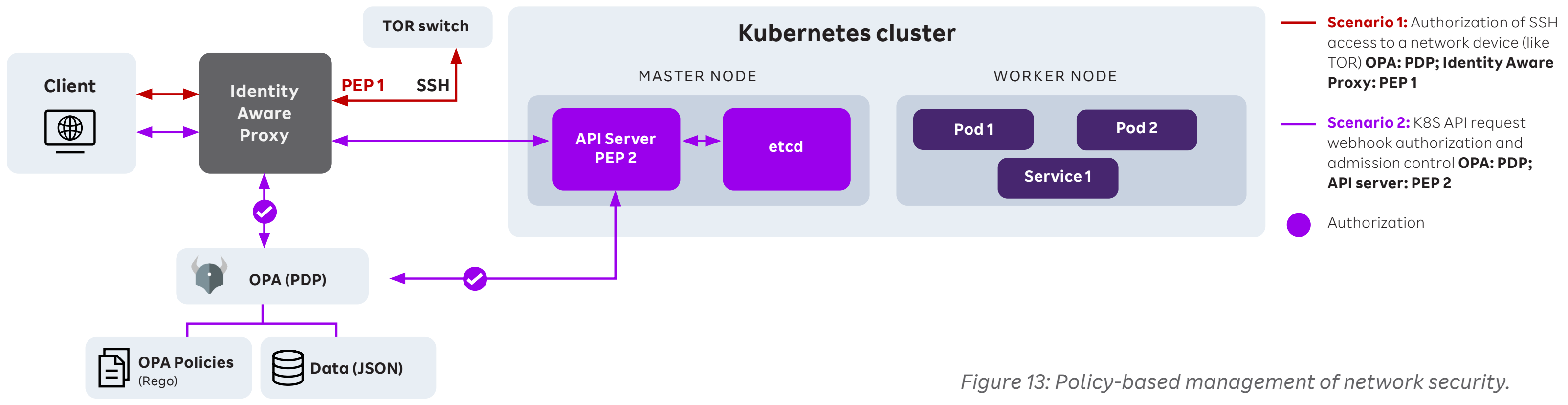

TOR switch
Kubernetes cluster
MASTER NODE
WORKER NODE
Client
Identity Aware Proxy
PEP 1
SSH
API Server PEP 2
etcd
Pod 1
Pod 2
Service 1
OPA (PDP)
OPA Policies (Rego)
Data (JSON)
Scenario 1: Authorization of SSH access to a network device (like TOR) OPA: PDP; Identity Aware Proxy: PEP 1
Scenario 2: K8S API request webhook authorization and admission control OPA: PDP; API server: PEP 2
Authorization

Figure 13: Policy-based management of network security.

Policies are used in different contexts to provide the rules needed to take a decision. There are different kinds of security policies and these policies must be flexible to consider exceptions and the dynamic nature of micro services-based systems to allow or deny requests.

Some of the examples of requests where policies could be applied are:

- An authorization request to access a specific application based on user roles / permissions.
- An admission request to update the cloud resources in Kubernetes.
- An API access request from one service to another.
- A time bound device / system access request from a user.

To provide strong security, default-deny or least privilege access policies should be in place. And on top of the deny policy, specific allow policies should be applied. The policies should be dynamic in nature to administer the needed control over the network functions.

Dynamic policies take input from different sources such as:

- ticketing systems,
- inventory databases,
- identity management / proxy systems,
- privileged/exception user/service lists,

and so on, to build the logic needed for the policy.

A central policy infrastructure is a key to gathering inputs such as O-RAN network function connection parameters, metrics from network function, current state of network function, or micro services, maintenance window time intervals, etc. These are then used to provide input to the policy or change the policy dynamically.

The central policy infrastructure could act as a central storage and management endpoint for policies. But the policy enforcement must happen closer to the system that is handling the requests. The policies can be distributed to the enforcement points for faster policy-based decision making.

6 | Container security

Container security is the practice of protecting containerized applications from threats using a combination of security tools and policies. Container security spans the full lifecycle of the container:

- starting with the build phase that covers aspects such as software supply chain security and DevSecOps tools used during development of containerized images,
- continuing into the distribute phase that ensures secure delivery of containerized images to its destination,
- and runtime deployment in a cloud native platform which includes security measures implemented to protect containerized applications from threats emanating at runtime, and runtime security measures that implement observability and monitoring capabilities.

The following sections describe a few security controls that can be employed in different phases of container security.

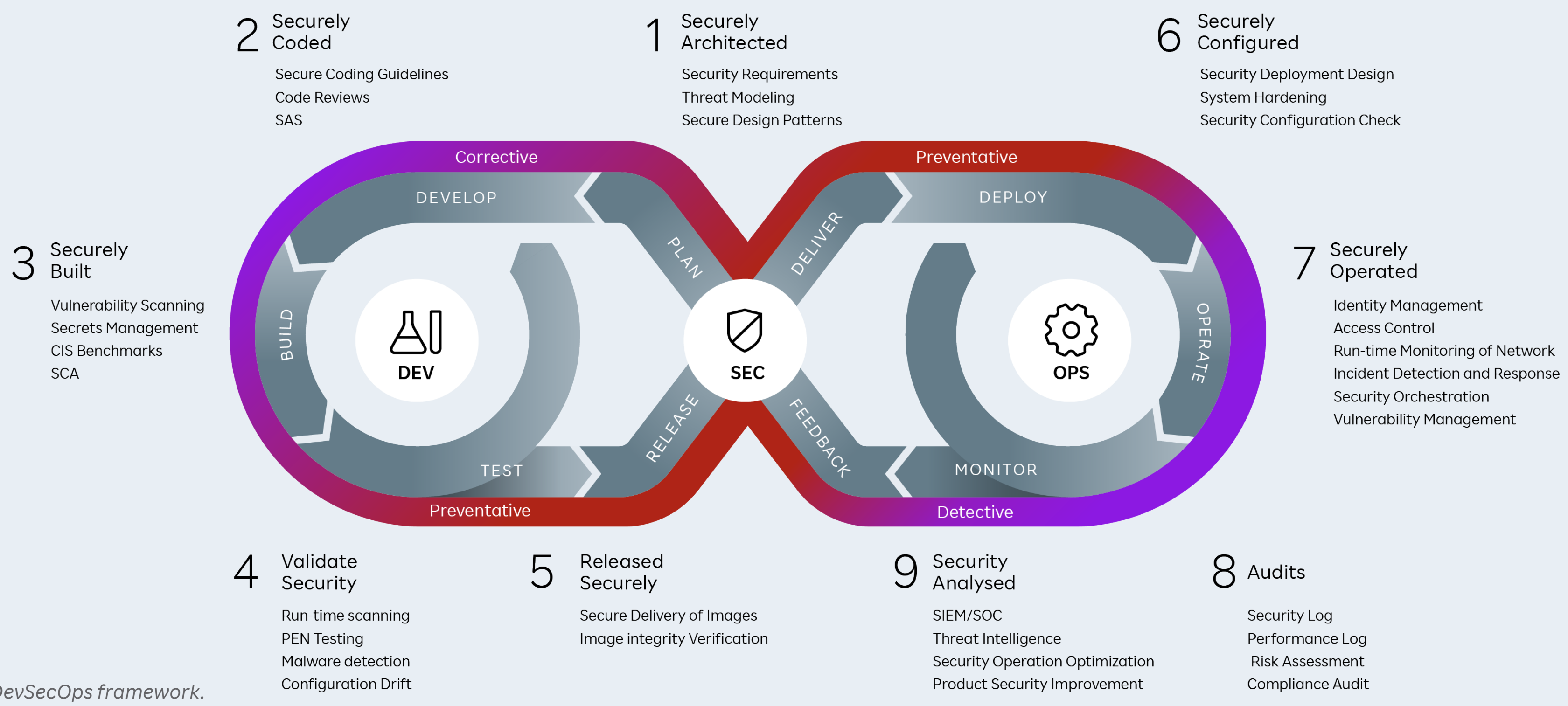


Figure 14: DevSecOps framework.

6.1 | Automated container lifecycle management using DevSecOps tools

This addresses requirement REQ-8 and REQ-9 in clause 6. Containers are inherently insecure as they contain a multitude of open-source components. To bring in the element of security into a container's lifecycle, we should begin at the inception phase (see Fig. 14). Securing a container begins with the container being built or packaged and continues throughout the life of the container until it reaches end of life.

Rakuten Symphony, as an organization, believes in security from the ground up and follows the below principles to ensure that every container workload provided to a customer has been secured:

- Secure design patterns, container base image selection.
- Secure coding guidelines for secure development and reviews.
- Vulnerability management – analysis and remediation.
- Static application security testing (SAST) during development and build phases.
- Secure image transfer and secure orchestration.
- Malware detection and CIS benchmark scanning during staging.
- Identity management / access control, run time monitoring and incident response.
- Log analysis and threat intelligence for solution optimization and improvement.

6.2 | Security assurance framework

Rakuten Symphony assures that every product has the highest standard of security. This is achieved by the Rakuten Symphony Security Assurance Platform, which brings together security specifications from 3GPP, ORAN and other standardization bodies, security best practices from CIS and NIST, along with security features that add a whole new dimension of security to telecom.

The Security Assurance Platform has baseline security requirements that bring in the essence of NESAS, NIST, 3GPP requirements (among a few), automated test cases to assess products against these requirements and providing remedial measures to ensure that Rakuten Symphony products have a high level of security.

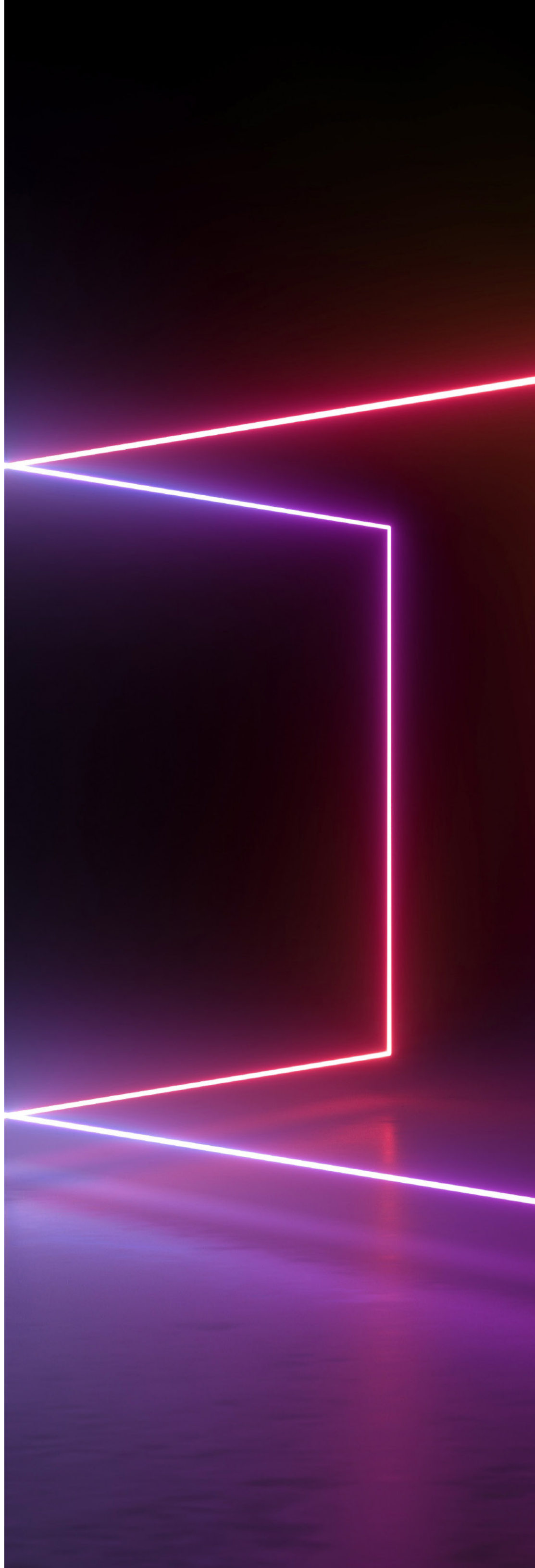
7 | Conclusion

5G and Open RAN introduce new infrastructure, functions, and interfaces. These technologies have the potential to make 5G the most secure network generation to date, but only when carefully configured and securely operated.

3GPP and the O-RAN Alliance provide a base blueprint and the design principles for securing telco-specific functions and interfaces. This security baseline must be completed by vendor- and operator-implemented security controls to reach a high level of hacking resistance, in particular for the IT and cloud systems underpinning modern networks.

When implementing these additional controls, telco vendors and operators can borrow knowledge and experience from related industries: Cloud operators have faced (and solved for the most part!) similar security challenges already.

Rakuten has been operating secure modern software for over a decade. We hope that the best practices shared in this guide contribute to the overall global security of 5G and Open RAN. Our societies rely on secure and reliable communication and information systems, and we are honored to contribute to the open knowledge pool on how to best secure the networks of the future.



8 | References

[1] Dell'Oro, Open RAN and vRAN Forecast, July 2022.

[2] ETSI, ETSI O-RAN specification, September 2022.

[3] TIP, Commercial realities, November 2021.

[4] NIST, Guide for Conducting Risk Assessments, September 2012.

[5] IEEE, Port based network access control, February 2020.

[6] IETF, OAuth 2.0 Authorization Framework, October 2012.

[7] O-RAN, O-RAN Security Threat Modeling and Remediation Analysis, July 2022.

[8] K8S, Kubernetes API access control, April 2022.

[9] K8S, Certificate management, April 2022.

[10] CNCF, Kubernetes policy management, (PDF) January 2022.

[11] 3GPP, TS 33.501 – Security architecture and procedures for 5G systems, (PDF) September 2022.

[12] German BSI, "Open-RAN Risk Analysis," December 2021.

[13] ENISA, "Threat Landscape for 5G networks – Updated threat assessment for the fifth generation of mobile telecommunications networks (5G)," December 2020.

[14] ENISA, "Cybersecurity of 5G networks - EU Toolbox," January 2020.

[15] NSA, "Potential Threat Vectors to 5G Infrastructure," June 2021.

[16] ENISA, "Threat Landscape 2021," December 2021.

[17] ENISA, "Understanding the increase in Supply Chain Security Attacks," June 2021.

[18] ENISA, "Cyber Espionage," January 2019.

[19] Altiostar, "Open RAN Security Whitepaper," 2021.

[20] ENISA, "Cybersecurity for 5G: ENISA Releases Report on Security Controls in 3GPP," February 2021.

[21] GSMA, "Network Equipment Security Assurance (NESAS)," December 2021.

[22] O-RAN SFG, "O-RAN Security Requirements Specifications v2.0," July 2021.

[23] O-RAN SFG: "O-RAN Security Protocols Specifications v3.0, July 2021.

[24] O-RAN SFG: "O-RAN Security Tests Specifications v2.0," July 2021.

[25] NIST SP 800-190, "Application Container Security Guide," September 2017.

[26] CSA, "Best Practices for Secure Container Architecture," July 2019.

[27] CSA, "Best practices for mitigating risks in virtualized environments," April 2015.

[28] NSA, "Security Guidance for 5G Cloud Infrastructures (Part I): Prevent and Detect Lateral Movement," October 2021.

[29] NSA, "Security Guidance for 5G Cloud Infrastructures (Part II): Securely Isolate Network Resources," November 2021.

[30] NSA, "Security Guidance for 5G Cloud Infrastructures (Part III): Data Protection," November 2021.

[31] NSA, "Security Guidance for 5G Cloud Infrastructures (Part IIV): Ensure Integrity of Cloud Infrastructure," December 2021.

9 | Terms and Abbreviations

Open RAN:	Generic term reflecting the broader movement to disaggregate the RAN
OpenRAN:	Disaggregated RAN using open interface specifications
O-RAN:	Disaggregated RAN using O-RAN Alliance specification
vRAN:	Disaggregates the software from the hardware
C-RAN:	Moves processing from the site to central location
Open vRAN:	vRAN plus Open RAN
Cloud RAN:	vRAN plus C-RAN

10 | Appendix - O-RAN threats

The O-RAN Alliance Technical Specification on Threat Modeling [7] has listed several threats that can be present in an Open RAN system.

Figure 15 lists the key threats:

Threat ID	Threat	Threat drivers
T-O-RAN-01	An attacker exploits lack of security posture in O-RAN components	· Outdated component from the lack of update or patch management
		· Poorly design architecture
		· Missing appropriate security hardening
		· Unnecessary or insecure function/protocol/component
T-O-RAN-02	An attacker exploits misconfigured or poorly configured O-RAN components	· Errors from the lack of configuration change management
		· Misconfigured or poorly configured O-RAN components
		· Improperly configured permissions
		· Unnecessary features are enabled (e.g., unnecessary ports, services, accounts, or privileges)
		· Default accounts and their passwords still enabled and unchanged
		· Security features are disabled or not configured securely
T-O-RAN-06	An attacker exploits insufficient/improper mechanisms for authentication and authorization to compromise O-RAN components	· Unauthenticated access to O-RAN functions
		· Improper authentication mechanisms
		· Use of Predefined/ default accounts
		· Weak or missing password policy
		· Lack of mutual authentication to O-RAN components and interfaces
		· Failure to block consecutive failed login attempts
		· Improper authorization and access control policy
T-O-RAN-08	An attacker compromises O-RAN data integrity, confidentiality and traceability	· Improper or missing ciphering of sensitive data in storage or in transfer
		· Improper or missing integrity mechanisms to protect sensitive data in storage or in transfer
		· Presence of active function(s) that reveal confidential internal data
		· No traceability (logging) of access to personal data
T-FRHAUL-02	An attacker exploits Unauthorized access to Open Front Haul Ethernet L1 physical layer interface(s)	Lack of authentication and access control to the Open Front Haul Ethernet L1 physical layer interface
T-MPLANE-01	An attacker attempts to intercept the Fronthaul (MITM) over M Plane	Lack of sufficient security measures in the Fronthaul due to the negative impact on the performance requirements
T-NEAR-RT-04	An attacker exploits non authorized Near-RT RIC APIs to access to resources and services which they are not entitled to use.	Non-authorized RT RIC APIs
T-xAPP-03	An attacker compromises xApp isolation	Vulnerabilities in the underlying system hosting xApps

T-rAPP-02	An attacker exploits rApp vulnerability for data breach or denial of service	rApp management is exposed to the tenant in a web front-end or REST API. These interfaces may contain software vulnerabilities or implement authentication and authorization insufficiently.
T-PNF-01	An attacker compromises a PNF to launch reverse attacks and other attacks against VNFs/CNFs	Mixed PNF-VNF/CNF deployments
T-SMO-01	An attacker exploits the improper/missing authentication weakness on SMO functions	Improper/missing authentication on SMO functions
T-SMO-02	An attacker exploits the improper/missing authorization weakness on SMO functions	Improper/missing authorization on SMO functions
T-A1-03	An attacker uses malicious function or application to modify messaging across A1 interface	weak mutual authentication
T-GEN-01	An attacker exploits flaws in deployed software	Vulnerable code exploits, Design Weakness
T-GEN-02	An attacker obtains malicious access to exposed services using valid accounts	Lack of authentication
T-GEN-04	An attacker exploits lack of Authentication & Authorization in interfaces between O-Cloud components	Lack of authentication, Insecure interfaces
T-VM-C-01	An attacker makes uses of privileged VM/containers that have all the root capabilities of a host machine, allowing the ability to access resources which are not accessible in ordinary containers.	Misconfiguration or Insecure VM/Container configurations
T-VM-C-02	An attacker exploits containerized application's vulnerabilities to breach its isolation boundary, gaining access to the host system's resources (VM/Container escape attack)	Shared tenancy vulnerabilities (multitenant environment), Lack of strong VM/Container isolation, lack of authentication, Insecure networking, Unrestricted communication between VMs/Containers
T-VM-C-03	An attacker takes advantage of insecure data storage and lack of authenticated access to it to steal VM/Container data	Lack of authentication, insecure data storage
T-VM-C-04	An attacker exploits the network vulnerability when VM/Containers are between two secured perimeters during migration, to gain unauthorized access to VMs, plant malicious code in the VM/Container images (VM/Container migration attacks)	Host misconfiguration, lack of authentication, memory pages copied in clear, vulnerable code exploits

T-VM-C-05	An attacker makes use of lack of access control (authorization) to change virtualization resources	Insecure O1/O2 interfaces, Lack of authentication/access control on IMS/DMS
T-IMG-01	An attacker tampers with VM/Container imag-es	Compromised VM/Container images (Build machine attacks, Supply chain attacks) at rest, lack of authen-tication, misconfiguration or Insecure VM/Container images configurations
T-IMG-02	An attacker exploits insecure channels be-tween images repository	Compromised VM/Container images in transit
T-IMG-03	An attacker gets access to secrets that are embedded in VM/Container images	Secret exposure in VNF/CNF images
T-OPENSRC-01	An attacker exploits known vulnerabilities in software component and untrusted libraries through a backdoor attack	· Inaccurate inventories of open-source software
		· Lack of consistent Supply Chain traceability and security
		· Lack of coding best practices
		· Modules with known vulnerabilities and untrusted libraries