Pekka Kuure

Security and trust in Open RAN

Master's thesis

Information Technology

Cybersecurity

2024

South-Eastern Finland
University of Applied Sciences

Degree title	Master of Engineering
Author (authors)	Pekka Kuure
Thesis title	Security and trust in Open RAN
Commissioned by	Nokia Bell Labs
Time	2024
Pages	109 pages, 7 pages of appendices
Supervisor	Vesa Kankare

ABSTRACT

Open Radio Access Networks (Open RAN) is a new industry approach to build mobile networks. Open RAN is driven by disaggregation, cloudification and automation of the radio network functions and open interfaces between disaggregated functions. Traditional approach in mobile networks has been that single supplier provides complete radio access solution in certain area. Open RAN approach promotes radio functions from multiple vendors to be integrated in the RAN network.

Open RAN Architecture specified by O-RAN ALLIANCE is evolving and new functionalities and features are added in each release. The current O-RAN Architecture however lacks any function to verify the integrity of distributed RAN functions, which are delivered by multiple parties.

This research studies how trust and security can be enabled in cloud-native multi-vendor Open RAN. The key concepts, semantics, and ontology related to trust as applied in Open RAN environment are described and explained. In addition to research, a solution based on remote attestation and trusted platform modules is created and verified in an experimental laboratory setup.

The results from this project suggest that remote attestation can be used to establish, maintain, monitor, and improve the integrity and good security posture of Open RAN network. The findings further conclude that further research is needed to define optimal solution when connecting functionalities from different trust domains together.

Keywords: trust, security, cloud-native, Open RAN, remote attestation, trusted platform module

Tutkintonimike	Insinööri YAMK
Tekijä/Tekijät	Pekka Kuure
Työn nimi	Tietoturva ja luottamus avoimen teknologian radioverkossa
Toimeksiantaja	Nokia Bell Labs
Vuosi	2024
Sivut	109 sivua, liitteitä 7 sivua
Työn ohjaaja	Vesa Kankare

TIIVISTELMÄ

Avoimen teknologian radioverkot (Engl. Open RAN) on uusi teollisuuden lähestymistapa mobiiliverkkojen rakentamiseen. Open RAN perustuu radioverkon toimintojen hajauttamiseen, pilvistämiseen ja automatisointiin sekä avoimiin rajapintoihin hajautettujen toimintojen välillä. Perinteinen lähestymistapa mobiiliverkoissa on ollut, että yksi toimittaja tarjoaa kokonaisen radioverkkoratkaisun tietyllä alueella. Open RAN edistää ratkaisua missä radioverkko integroidaan usean toimittajan ratkaisuista.

O-RAN ALLIANCE:n määrittelemä Open RAN -arkkitehtuuri kehittyy koko ajan, ja uusia toiminnallisuuksia ja ominaisuuksia lisätään uusiin versioihin. Nykyinen O-RAN arkkitehtuuri ei kuitenkaan sisällä ratkaisua millä hajautetun monitoimittaja-radioverkon komponenttien alkuperä ja eheys voitaisiin varmistaa.

Tämä tutkimus käsittelee, miten luottamus ja tietoturva voidaan ottaa käyttöön pilvipohjaisessa Open RAN -ympäristössä. Luottamukseen liittyvät keskeiset käsitteet, semantiikka ja ontologia Open RAN -ympäristössä kuvataan ja selitetään. Teoreettisen tutkimuksen lisäksi esitetään käytännön ratkaisu etävarmennuksen toteuttamiseksi, joka varmennetaan kokeellisessa laboratorioympäristössä.

Tämän projektin tulokset osoittavat, että etävarmennusta voidaan käyttää muodostamaan, ylläpitämään, monitoroimaan ja parantamaan Open RAN -verkon eheyttä ja hyvää tietoturvan tasoa. Tulokset osoittavat myös, että optimaalisen ratkaisun määrittämiseksi tarvitaan lisätutkimusta.

Avainsanat: luottamus, tietoturva, natiivi pilviratkaisu, avoimen teknologian radioverkot, etävarmennus, luotettu järjestelmämoduli

ACKNOWLEDGEMENT

First, I want to express my gratitude to Prof. Dr. Ian Oliver from Jyväskylä University for his excellent guiding to the world of trusted computing and remote attestation. The sessions with the raspberries and brainstorming by the whiteboard have been extremely brilliant and educative. Diolch yn fawr am addysgu rhagorol.

This research has been done for Nokia Bell Labs and I have had the pleasure to exchange views, get challenged and learn from my wonderful colleagues. I would like to express my sincere thanks to Kaisa, Roosa, Leo, Gerg0, Istvan, Jonne, Tero, Sari and many others. Special thanks to Petteri as without his helping hands with Kubernetes and Docker, the experiment lab would not have been realized.

I want to thank my supervisor Vesa Kankare for his excellent guidance and constructive feedback throughout the thesis process. Great thanks also to Kimmo Kääriäinen and Marko Oras for your excellent lecturing on cybersecurity. You are all teaching with big heart.

Finally special thanks go to my wonderful family Satu, Otto, Oula & Unna. I know that you have had to share my pain when the lab experiments have failed, yet you have supported me full heart on my studies.

Espoo, April 30th, 2024

Pekka Kuure

ABBREVIATIONS AND SYMBOLS

3GPP	3rd Generation Partnership Project
5G	5th Generation
6G	6th Generation
AAL	Acceleration Abstraction Layer
AI/ML	Artificial Intelligence/Machine Learning
AK	Attestation Key
API	Application Programming Interface
ARM	Advanced RISC Machine
BIOS	Basic Input/Output System
CLI	Command Line Interface
CNCF	Cloud-native Computing Foundation
CPU	Central Processing Unit
CRI-O	Container Runtime Interface for Open Container Initiative
CRTM	Core Root of Trust Measurement
CVE	Common Vulnerabilities and Exposures
CU	Central Unit
DU	Distributed Unit
EK	Endorsement Key
EO	Executive Order
etcd	/etc distributed
FCAPS	Fault, Configuration, Accounting, Performance and Security
gNB	Next Generation Node B
HroT	Hardware Root of Trust
HW	Hardware
IETF	Internet Engineering Task Force
ISA	Information Society of Automation
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
IT	Information Technology
JANE	Jyväskylä AttestatioN Engine
MAC	Medium Access Control
MANO	Management and Orchestration
MB	Megabyte
MSRC	Microsoft Security Response Center

NF	Network Function
NFV	Network Functions Virtualization
NGC	Next Generation Core Network
NIST	National Institute of Standards and Technology
O-CLOUD	O-RAN Cloud
O-CU	O-RAN Central Unit
O-CU-CP	O-RAN Central Unit Control Plane
O-CU-UP	O-RAN Central Unit User Plane
O-DU	O-RAN Distributed Unit
O-RAN	Open Radio Access Network based on O-RAN ALLIANCE
O-RU	O-RAN Radio Unit
OS	Operating System
OT	Operational Technology
OWASP	Open Worldwide Application Security Project
PCR	Platform Configuration Register
PDCP	Packet Data Convergence Protocol
PHY	Physical Layer
PTP	Precision Time Protocol
RA	Remote Attestation
RAN	Radio Access Network
RATS	Remote Attestation Procedures
RFC	Requests for Comments
RoT	Root of Trust
RRC	Radio Resource Control
RTS	Root of Trust for Storage
RTM	Root of Trust for Measurement
RTR	Root of Trust for Report
RU	Radio Unit
SDAP	Service Data Adaptation Protocol
SMO	Service Management and Orchestration
SP	Special Publication
SRTM	Static Root of Trust Measurement
TBB	Trusted Building Block
TCB	Trusted Compute Base
TCG	Trusted Computing Group
TEE	Trusted Execution Environment

TEEP	Trusted Execution Environment Provisioning
TPM	Trusted Platform Module
UEFI	Universal Extensible Firmware Interface
VM	Virtual Machine
VNFM	Virtualized Network Function Management
ZTA	Zero Trust Architecture

CONTENTS

1 INTRODUCTION

Since the introduction of the first commercial cellular network 1979 in Japan, mobile networks have evolved to become central nervous systems of modern societies. These systems connect humans, machines, and services together and their criticality to economy and daily life of society has been recognized by governments. All critical sectors of society such as finance, energy, water, transport, healthcare, are dependent on communications sector currently catered by 5G system (DIGITALEUROPE 2023; Barr 2020).

It is not only the criticality of the systems, but also the new architecture and the deployment models that set completely new challenges to end-to-end security. The communication through mobile networks takes place between the endpoints that are the user equipment and service residing in internet. In between there is 5G system, which can be further broken down to Core Network and Radio Access Network parts. Figure 1 below describes the 5G system based on the new Service Based Architecture (SBA) as defined by 3rd Generation Partnership Project (3GPP). Core Network provides end users reliable and secure connectivity to network with help of its key functions such as connectivity and mobility management, authentication and authorization, subscriber management and policy management. In 3GPP defined 5G architecture, Core Network consists of services offered by each Network Function (NF) over standard defined Application Programming Interfaces (API).

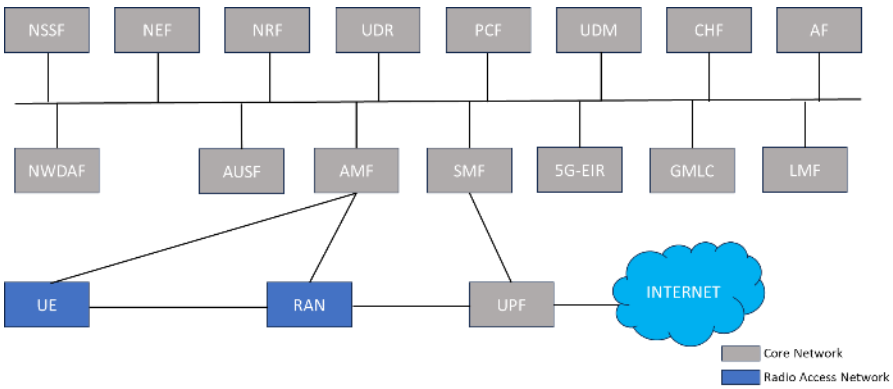

Figure 1. 5G System based on Service Based Architecture

Radio Access Network comprises of radio parts and connects the user devices through the network to services in the cloud. The radio part conceptually consists of the User Equipment (UE) and RAN functions. In 5G system the RAN function supporting 5th generation radio technology is called a gNB, standing for 3GPP 5G Next Generation base station.

The security approach that has been used for the previous generations purpose-built point-to-point architecture radio networks has been considered no more valid for the disaggregated, cloudified multi-vendor Radio Access Network (RAN), whose attack surface is significantly expanded by completely new types of threat models with their specific attack vectors (Liyanage et al. 2023).

Cloudification, virtualization, disaggregation, automation, and interoperability of the radio access networks are the major themes currently driven by the industry. O-RAN ALLIANCE was launched early 2018 to promote and work on these themes to enable Open RAN that is interoperable and can be cloudified. In the O-RAN Architecture, the RAN is split into Central unit (CU), Distributed Unit (DU) and Radio Unit (RU) and interfaces are defined between those to support interoperability. Service Management and Orchestration (SMO) layer is designed to manage the underlying network including the Cloud Infrastructure and Network Functions over O2 and O1 interfaces. The actual deployments are selected based on technical and business criteria and in real life there will be plethora of different combinations with both cloud based and classic deployments (O-RAN 2023a).

From security point of view there are multiple new aspects to be considered. The key problem to resolve is how to enable trust in this dynamic system with a substantial number of players and their products involved, and how to maintain it over the lifetime of the service. The present research project studies how trust and security can be enabled in an Open RAN architecture. The project is made for Nokia Bell Labs research unit and its goal is to create security solutions that can be used in practical network deployments based on cloud-native, disaggregated Open RAN system.

2 RESEARCH PROBLEM, METHODS, AND GOALS

The dynamic and complex environment of Open RAN with increased number of stakeholders sets completely new challenges to secure the RAN. Extreme speed and agility of the loosely coupled systems with frequent component and micro-service updates and configuration changes requires automation and with automation also the resiliency of the system can be enhanced by using observability, analytics, and controls. While this sounds ideal, it also increases the attack surface and opens broad opportunities to attack these systems.

The essential criteria for appropriate security posture in this kind of environment is trust. How trust can be enabled and assured in such a critical system? How to trust that the function is the one it claims to be? How to trust that each function is hosting and executing software that it is supposed to do? How to validate the entire system stack from bottom up? How to ensure that the system has not been intruded and tampered? How to create the Root of Trust and Chain of Trust in the environment with multiple stakeholders?

When security of a critical system is based on concept of trust, it is essential first to define trust and the related terms. Different standards and industry organizations are not aligned with the terminology and there are several different definitions used for the trust related terms. As an example, the US National Institute of Standards and Technology (US NIST) alone has several descriptions for the term “trustworthiness”. In their publication “Developing Cyber-Resilient Systems” trustworthiness is described as:

> Worthy of being trusted to fulfil whatever critical requirements may be needed for a particular component, subsystem, system, network, application, mission, business function, enterprise, or other entity. (NIST SP 800-160 2021)

Their other document,” Risk Management Framework for Information Systems and Organizations” defines it differently:

> The attribute of a person or enterprise that provides confidence to others of the qualifications, capabilities, and reliability of that entity to perform specific tasks and fulfil assigned responsibilities. (NIST SP 800-37 2018)

As trust is the foundation of the security in a system, this research aims to study how trust is established. Further, it will be explained how that trust is transferred and limited across the system (the chain of trust) and how trust is maintained over time. Once the concept of trust is clearly defined, the focus will shift to relationships and interactions of the functions. Figure 2 below depicts a process of how trust develops based on actions between the entities.

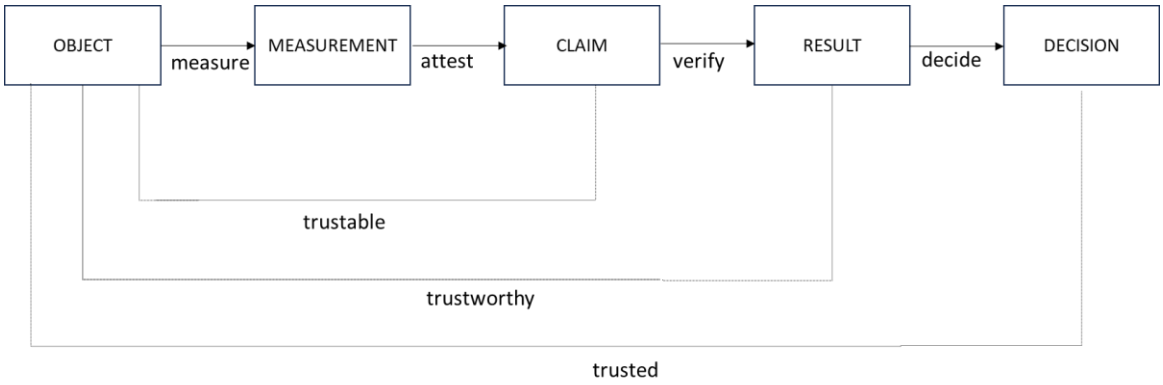

Figure 2. Development of trust between entities (Oliver et al. 2023)

Figure above lists three different trust related adjectives: trustable, trustworthy, and trusted. In this research work they are defined as follows:

- A trustable object is one on which attestation can be performed.
- A trustworthy object is one that can be verified. An object with a reference, expected values which can be compared.
- A trusted object is one for which a positive decision can be made.

This model is part of ongoing ontology research and aims to be a "unifying" model of said terminology.

2.1 Research Problem

The research problem is to enable trust in disaggregated, cloud-native Open RAN system and includes describing the semantics and ontologies of trust and in parallel demonstrating a technical solution how to secure a multi-vendor Open RAN based upon trust. The basis for the research work is the O-RAN ALLIANCE defined architecture as illustrated in Figure 3 below.

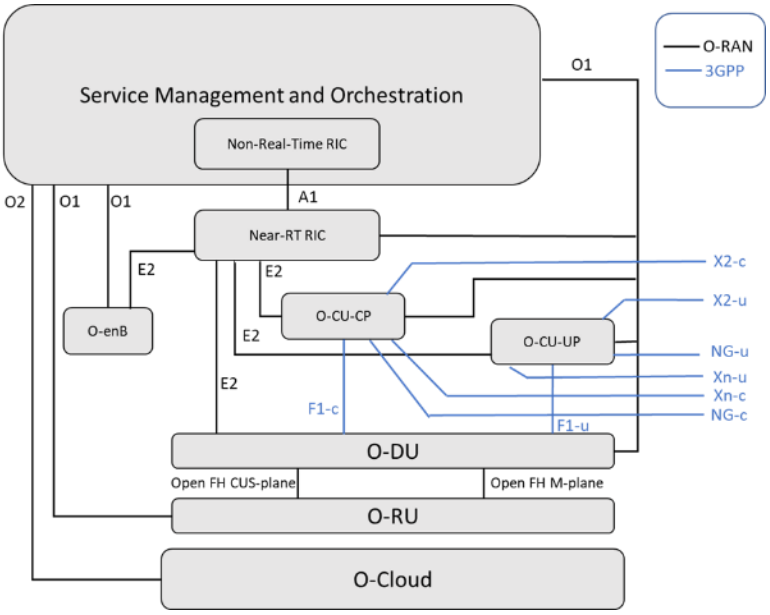

Figure 3. O-RAN ALLIANCE logical architecture (adapted from O-RAN 2023a)

In addition to O-RAN Architecture, this research assumes practical deployment scenario based on cloud-native technologies where the functions of the network are built as scalable components or micro-services. The deployed network is not only consisting of the network functions, but there are also applications exposed with the information from the network that are allowed to influence the operation of the network via declarative policies.

Research Questions

The primary goal of the work is to research how trust can be enabled in Open RAN system. While this may sound trivial, there are aspects that must be explained and clarified. First, it needs to be defined what trust is and explain the re-

lated semantics and ontologies. Secondly, it must be explained what Radio Access Network (RAN) is, and specifically what is meant by Open RAN. Once the foundation is properly laid down, concepts of trust and RAN are connected with each other and described how trust is applied and enabled in Open RAN to enforce security and integrity in the system to be maintained over the lifetime of a service.

Research aim 1: Enabling Trust in Open RAN

Aligned with the primary goal of the research, the first question focuses on what is trust in context of RAN and how trust can be enabled in Open RAN system.

Deploying cloudified, disaggregated, and Open 5G RAN network based on 3GPP defined Service Based Architecture brings several technological changes compared to previous generation point-to-point networks. From security point of view the old perimeter-based security model is no more feasible, but instead due to number of stakeholders and different combinations of functions, it is fair to assume the attacker already inside the system. This assumption has also been driver in US NIST defined Zero-Trust Architecture (NIST SP 800-207) where there is no implicit trust inside the system, but instead trust is built upon validation.

Research aim 2: Requirements of trust in multi-vendor Open RAN

Considering the key characteristics of an Open RAN network, where the functions of a system may come from multiple vendors, it is important to study **what the specific requirements of trust are in this scenario.** Answering this question requires to define an ontology of the trust with all the entities and their relationships and actions included. Figure 4 describes a high-level model of trust relationships in an O-RAN system.

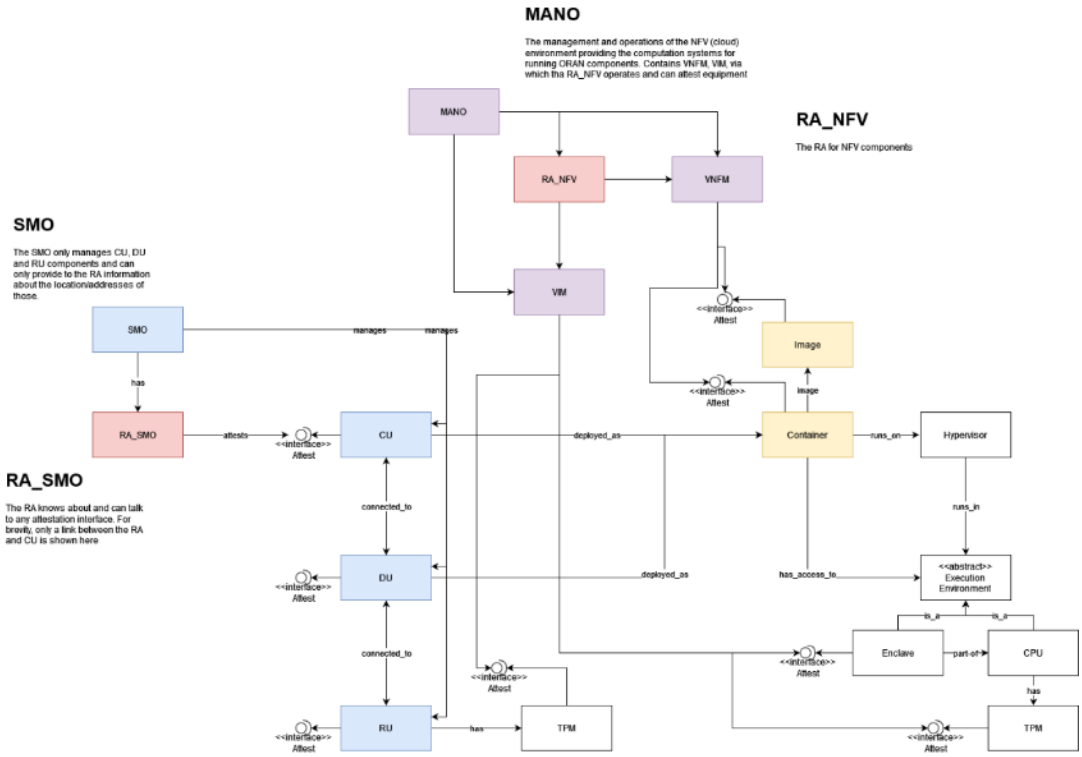

Figure 4. Ontology of Trust in O-RAN system

Research aim 3: Drivers for trusted systems

When trust is based on validation and using measured results that can be queried at any moment afterwards, the integrity of the measurement results and the validation process must be protected. IETF Internet Security Glossary (RFC 4949 2013) describes the terms used for Internet Security and describes an established point of trust as a source where the validation path begins. The document further defines Trusted Computing Base as the totality of protection mechanisms within a computer system, including hardware, firmware, and software, the combination of which is responsible for enforcing a security policy. While there exist several mechanisms to enforce security policies, the trust must be anchored in a robust way. For Open RAN systems, based on industry view it is assumed that trusted systems and trusted modules are critical elements in enabling the trust. To validate this assumption, this research studies **what are the drivers to use trusted systems in cloud-native O-RAN system.**

There are few alternatives for trusted systems and based on the requirements the best solution for the needs can be selected. Hardware Security Modules (HSM) have been already widely deployed in the industry and many of the current smartphones and laptops use Trusted Platform Module (TCG 2019a; TCG 2019b).

Research aim 4: Changes required in O-RAN architecture

As O-RAN ALLIANCE architecture is used as a basis in this research and as the current architecture does not have any functionality for this area, it is likely that there will be additions or changes in the current architecture. The last research question is therefore **what kind of changes need to be introduced in the current O-RAN architecture to enable trust in different deployments.**

Research Methods

Selecting a research methodology is important to plan properly as it will determine the success of the research. When considering Qualitative research and the overall problem setting, there are several distinctions: are there singular or multiple realities, is the problem unique or can it be generalized, is there are need to follow a particular methodology of qualitative research or not. While there are different approaches, the common nominator is the need to provide credible and trustworthy research results (Yin 2016).

The starting point for a qualitative research project may vary, but at least the following three topics must be covered: A topic, that defines what is going to be studied, a data collection method explaining the ways how data is collected, and sources of data defining where the data will be collected (Yin 2016).

When starting a research project, the potential ways forward may be influenced by the past experiences or interest on the topic, which could already set the directions for the work. In case there is low or no exposure to the topic, a fresh start would become an alternative, which would allow building completely new way of

approaching the field. For this research, the different research methodologies presented in Figure 5 have been considered.

Used guidelines for research methodologies	
Social Research Methods 4rd Edition, Bryman, A. 2012. Oxford University Press.	Katri Ojasalo, Minna Koskelo and Anu K. Nousiainen. 2015. Foresight and Service Design Boosting Dynamic Capabilities in Service Innovation. Laurea University of Applied Sciences
Yin, R. K. 2016. Qualitative research from start to finish 2nd edition. New York: The Guilford Press.	Kivinen, S. 2023. Integrating Innovations Into a Company's Strategy Through a Concept of Sustainable Futures Business Design, Laurea University of Applied Sciences.
Moilanen, T., Ojasalo, K., & Ritalahti, J. 2022. Methods for development work: New kinds of competencies in business operations. Helsinki. Books on demand.	Red Teaming Handbook 3rd Edition. 2021. UK Ministry of Defense.
Stickdorn, M. 2018. This Is service design doing. O'Reilly Media.	Tate, J., Happ, M. (2017). Qualitative Secondary Analysis: A Case Examplar. Journal of Pediatric Health Care. Research Methods Volume 32, Issue 3. 308-312, May 2018.
Kananen, J. 2017. Laadullinen tutkimus pro graduna ja opinnäytetyönä. Jyväskylän ammattikorkeakoulu.	

Figure 5. Different research methodologies analyzed for this research.

Research design can be considered a logical blueprint for the work, and they provide logical plans what will be executed, but not the logistics that relate more to project management aspects such as scheduling and coordination (Yin 2016). For a study like this, it is likely that selecting a single methodology would result in suboptimal results and that instead a set of mixed methodologies would provide better results. As described in section 2.1, the research problem indicates that the goal is to create a practical security solution for Open RAN environment, which will influence the selected approach. The research will be started with a qualitative secondary analysis (Tate et al. 2017), a phase which is a common approach to a systematic investigation where the research depends solely on existing data from appropriate and available sources.

Information gathering is a critical phase of a project and prone to many errors that could strongly influence the research directions and results. The key criteria for problem solving are to properly understand the problem that needs to be solved as it will set directions to materials that need to be collected (UK Ministry of Defence 2021). In addition, depending on experience of the researcher, one may

collect material from sources where they have better or previous access, such as their own network (Yin 2016). Also known bias is goal-directed behavior, where an individual has a strong or clear preconceived idea of what they are looking for or expecting to happen (UK Ministry of Defense 2021). From these points of view, the list of materials listed above is indicated to be an initial list and those may even completely change along the progress.

The considerations above suggest that there are several unknowns in the overall process, and both the methodologies and the used source information have uncertainties. Following strictly and merely qualitative secondary research could bring us answer to the research problem based on used information that was available. But what if the information used is wrong or too narrow or completely based on false assumptions? How to ensure that the problem solved is right problem to solve and worth solving? Moilanen et al. (2022) have studied methods for the development work and are discussing about research orientation highlighting the need for systematic, analytic, and critical approach, where solutions and knowledge is built on the foundation of existing knowledge. They define the research-based development with the aim to solve practical problem while often producing new knowledge and where data that is systematically collected, is critically evaluated against both practice and theory. In their defined model the research-based development consists of a combination of different methods that are used in multiple ways. The research orientation is described as organized working where data is acquired both from research studies and practical knowledge with analytical and critical thinking ensuring that different perspectives are considered. The model suggested by Moilanen et al. is a P-I-E process having the stages of Planning, Implementation and Evaluation.

Moilanen et al. (2022) also present different approaches where research can be connected to development work. A Case Study represents more of pure research, while Innovation Generation is considered pure Development. In the middle of research and development there is Constructive Research where the goal is to solve a problem by creating a new construct. This process has six steps: 1)

Searching for a meaningful problem, 2) acquiring in-depth theoretical and practical knowledge, 3) planning solutions, 4) testing the functionality and showing the viability, 5) showing the connections of the solutions to the theory and 6) examining the extent which the solution can be applied.

This Constructive Research Process has the flavors that make it relevant candidate process in enabling trust and security in Open RAN. In their research Moilanen et al also present Service Design as one of the approaches, positioning it as pure development and having limited research focus. Stickdorn (2018) has been strong advocate of Service Design, highlighting its great advantage to decrease the gap between theory and practice. In his approach, there is the double-diamond model as illustrated in Figure 6, which presents the diverge-converge characteristics of the model.

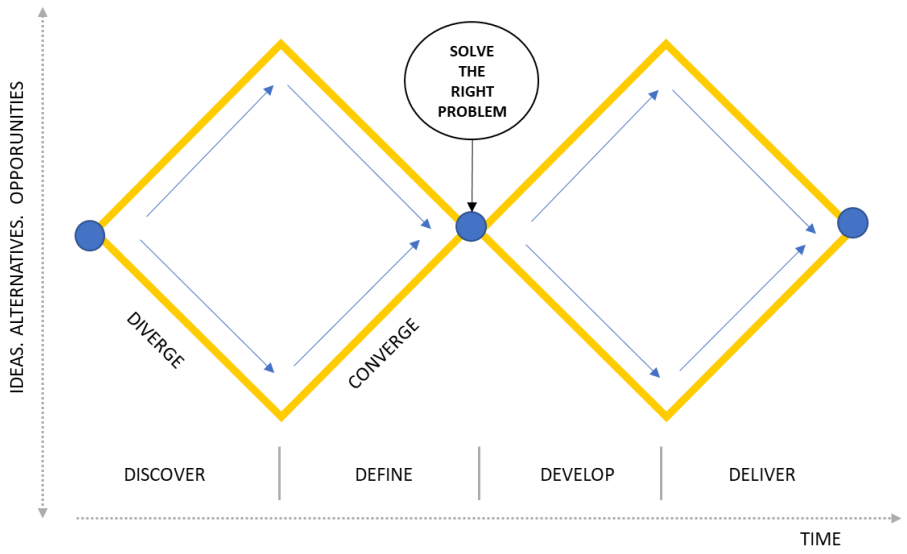

Figure 6. Double diamond with divergent and convergent thinking and doing (adapted from Stickdorn 2018)

Double diamond methodology is designed to ensure that the researchers will solve the right problem. It can be simplified to two main phases, first to ensure understanding what is going to be resolved and what is the real need, and then to create service or product that resolves the problem and provides true value. According to Stickdorn (2018), the four core activities of Service Design are research, ideation, prototyping and implementation. In his description, the desk research is preparatory research conducted as secondary research.

The researchers widely use the diamond model, and while Stickdorn presents double diamond, Ojasalo et al. (2015) have adapted the same diverge-converge model but extended it to include future-oriented design aspects. Their service innovation process grounded on foresight and service design consists of four distinct phases: map & understand, forecast & ideate, model & evaluate, and conceptualize and influence. The interesting future-looking characteristics in the approach is achieved by taking the findings from the mapping and understanding phase forward as inputs to ideation and forecasting alternative futures. From cybersecurity point of view as the development of AI and Quantum technologies are accelerating, it is essential to include future-orientation to any research and design on security of a communications system. Otherwise, one would end up designing and documenting history.

The research approach designed for this research consists of mixed methodologies and adds separate phases with the methodologies that are considered best suited for the phase. The work started with a fieldwork to interview subject matter experts and to gather understanding for the right study questions. The overall process consists of three distinct phases as depicted in Figure 7.

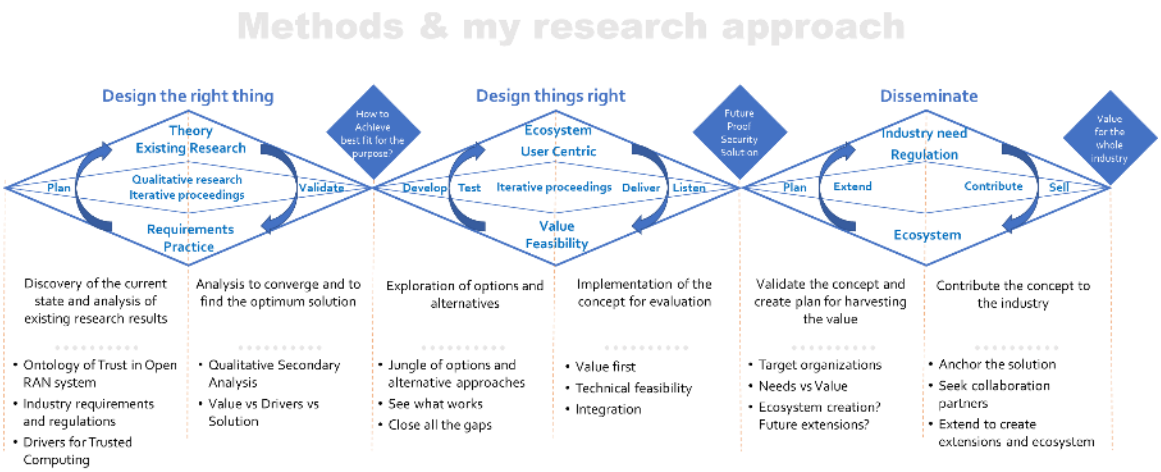

Figure 7. Triple diamond – research, design, disseminate. (adapted from Kivinen 2023)

Each different phase in the process has its own purpose and they all use process considered optimum for the phase. The first phase is to ensure that the right problem to resolve is found and in this phase the theory and connected to found

requirements and practice. The validated results from this phase are fed as input to next phase that aims to design the solution the right way. The focus in this phase is value and other aspects such as technical feasibility comes after the value. This part of the overall process will also consider future-aspects of the solution. The last phase on dissemination of the results is compact and focuses to ensure that the developed solution will have future and get traction in the industry.

The process designed for this research can be also illustrated with a visualized model inspired by research on integrating innovations through a concept of sustainable futures business design (Kivinen 2023). Figure 8 highlights the motivations of separate phases of the research and the questions they will seek answers.

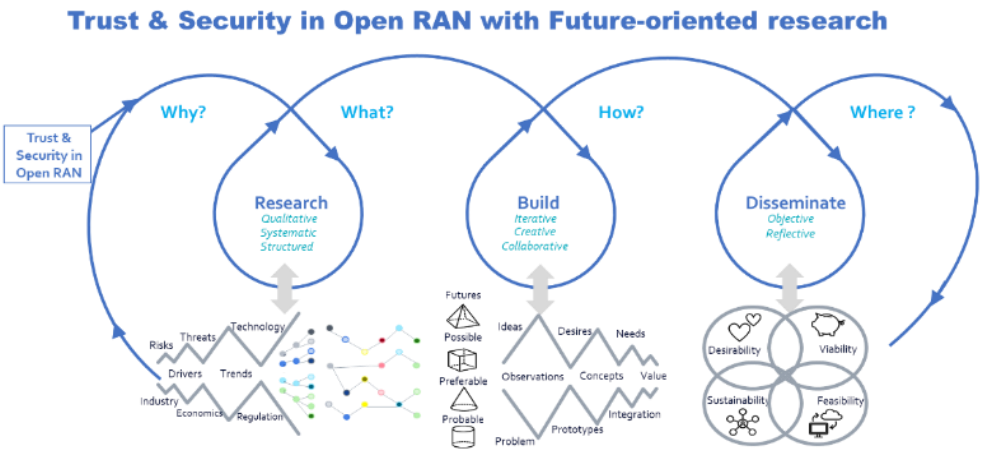

Figure 8. Visualization of the project (adapted from Kivinen 2023)

What should be highlighted in the process is that this is a logical model and not a logistical model. While phases are presented to be pipelined, in real life they will be running in overlap and there will be interactions between the phases. As an example, the build process will start early to prepare the experimental security lab for the extensions required by Open RAN security functions, and there may be observations or findings that will be fed as input to the research phase. Similarly in the closing phase of dissemination, should there be findings that would suggest further iterations in the build process, those could be taken as input to few more iterations.

The qualitative research phase of the research aims to lay the foundation for the work by examining and describing the trust related concepts, semantics, and ontologies that are relevant for the work. After that, the work focuses in studying and defining how the trust can be enabled in the Open RAN environment. The literature studies will be conducted from the documents of the leading global industry bodies in the subject area as those that are considered most relevant for this work. Examples of those are listed in the Figure 9 below.

Initial list of Data Sources	Available at
O-RAN ALLIANCE technical specifications and reports.	https://www.o-ran.org
Internet Engineering Task Force (IETF) RFCs and internet-drafts in security	https://www.ietf.org/
3GPP technical specifications and reports	https://www.3gpp.org/
Trusted Computing Group Technical specifications and reports	https://trustedcomputinggroup.org/
European Union Network and Information Security Agency (ENISA) requirements	https://www.enisa.org
National Institute of Standards and Technologies (NIST) requirements documents	https://nvlpubs.nist.gov/nistpubs/
Customer requirements from various operators	Several sources

Figure 9. Initial list of Data sources

As trust is indicated to be the foundation for the Open RAN security, it is crucial that the related terminology is properly explained and aligned with the industry that any selected subject area expert would understand what is being suggested. In this work, an analysis from the documents of the industry leading organizations such as IETF, NIST, TCG is performed, collecting the trust related terms and concepts, and defining how those (trust related terms) are used in this work. The documents of those industry leading expert organizations are further used in defining the technical solution for Open RAN environment. This will ensure that the top-notch work from the industry is efficiently re-used where applicable.

The data used in this research is not limited to the above listed sources, but as indicated in the research process description, the separate phases have interactions between each other. The design phase that is intended to validate the research results will hopefully provide findings and data that can be fed back to the

qualitative research process. The progress, test results and findings from the design phase are recorded in a lab diary that is maintained from the beginning of the design phase. The concrete results of the laboratory work will hopefully testify that the theories and concepts presented in this work will work and that those can be beneficially used to create secure Open RAN systems. To summarize, the research phases 1 and 2 and their contents are presented in figure 10 below showing the theme and its motivation as well as sources and types of material used in the phase.

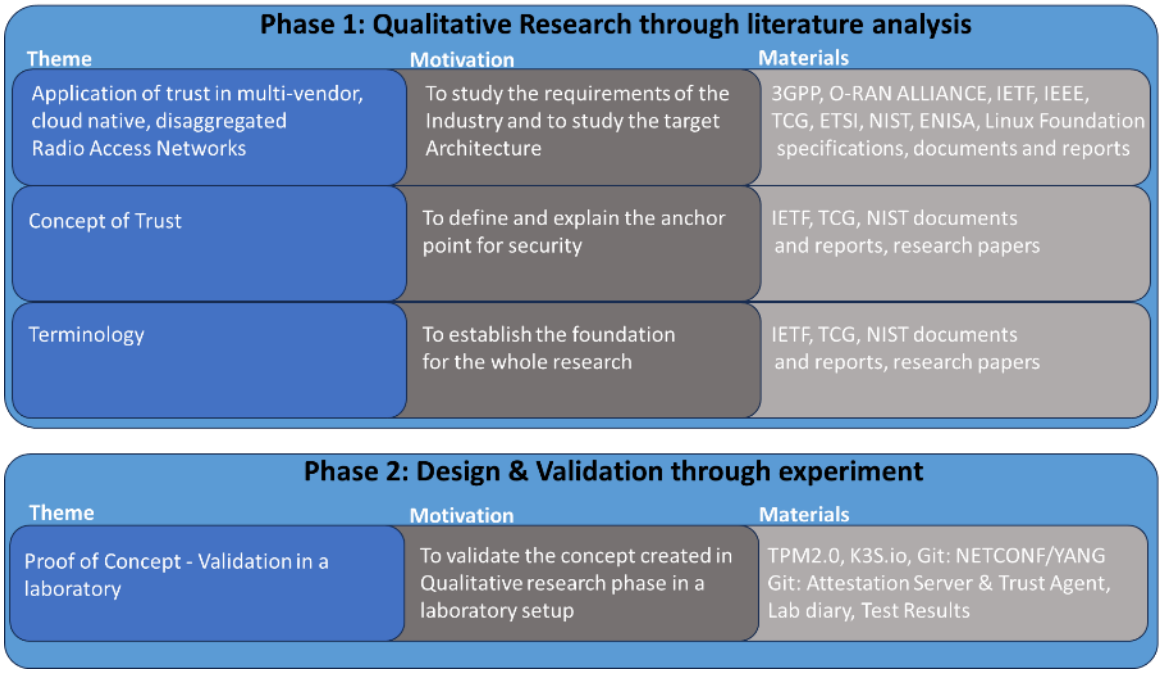

Figure 10. Research themes, their motivation and use of information in different phases of the project

2.2 Research Objectives

The research is primarily a functional research development project with the goal to define a security solution for a practical Open RAN deployment case. Before entering deep in the solution space, the security related concepts and the terminology are studied and defined. To understand the different relationships between the entities in an Open RAN network, the ontology of trust and the related requirements on trust and security are studied and discussed in detail. This provides the necessary motivation why protecting the system with the developed solution is such a necessity. Finally based on the derived requirements, a practical

solution that enables trust and security in the complex multi-party Open RAN system is developed and presented.

2.3 Theoretical framework

The research work will focus on studying how trust can be enabled in cloud-native Open RAN system. As trust is used as a foundation for security and while the telecom and internet industries use trust related terms broadly and inconsistently, it is essential to properly define and explain the related terminology and how it is applied in Open RAN system. As an example, the term trustworthy has become endemic in different industry and standardization forums and their proceedings. As a result, it has received multiple different meanings. The research paper Oliver et al. (2023) focuses on ontologies related to trust in Open and Cloudified RAN environment and further explain the levels of trust, how trust is transferred in the system and how the concept of trust can be used overall.

In Open RAN deployment the network functions are split and can be hosted in private, public or hybrid cloud environments. Further different vendors may provide the different functions. This assumption is valid not only for the RAN and Core network functions, but also for the management systems and the infrastructure hosting these functions. In this kind of scenario, the trust cannot be assumed based on the location or on the vendor of the function, but moreover trust can be only based upon validation of identity, integrity, and the trust relationships that individual elements of the system build between other elements.

US NIST has defined a concept called Zero Trust, where the defenses are moved from a static network-based perimeters to focus on users, assets, and resources (NIST SP 800-207 2020). In this approach there is no implicit trust assumed for an asset based on its location or ownership, but rather there is always authentication and authorization before a session to a resource is accepted to be established.

In an open RAN system, there are numerous functions or entities in the end-to-end chain that must be trusted and following the principle of Zero Trust, the trust

can be only build on validation. IETF has made efforts to standardize Remote Attestation where series of RFCs describe the architecture and procedures that can be used to establish trust between the endpoints. This project follows the RATS architecture (Remote Attestation Procedures) and the roles that are used in a system to establish trustworthiness of a remote peer. In this approach, the trustworthiness of a system component is based on evidence that may include system component identity, composition of components, root(s) of trust, system component configuration and system component integrity.

In addition to the RATS architecture, the IETF TEEP (RFC 9397 2023)
A Trusted Execution Environment Provisioning Architecture is relevant to consider. This work specifies Trusted Execution Environment (TEE) as an environment that enforces that any code within the environment cannot be tampered with, and any data used by such code cannot be read or tampered with by any code outside the environment.

The key criteria in assessing the trustworthiness and validating the integrity of a system (and the components of the system) is that trust is anchored somewhere in secure way. Trusted Computing Group has been working on concepts to enable secure computing and one of their key work areas is defining standards for hardware-based security. Trusted Platform Module (TCG 2019b) is a secure crypto processor that has been globally adapted as an ISO/IEC 11889 standard. It is a dedicated microcontroller that can secure hardware through integrated cryptographic keys, and it has been already successfully used in billions of smart phones and computers. TPM or Hardware Secure Module in general is key component in enabling trust as it can be used to anchor the trust and to establish Root of Trust in a system.

In this project, Remote Attestation together with Hardware System Module in anchoring the trust will be studied. Based on the existing work in other fields of technology, it is believed that creating a Root of Trust allows enabling trust and integrity of the Open RAN system through all the layers, from the HW in the bottom to the applications on the top of the layered architecture.

3 TRUST – THE FOUNDATION OF SECURITY

Open RAN as a new paradigm in the telco industry is interesting from the security point of view due to multiple aspects. Combined with the parallel developments in the virtualization and AI, both the complexity and the number of combinations in an Open RAN system increase significantly, which results in remarkably increased attack surface (Liyanage et al. 2023). Further, as the network functions of the overall end to end solution comes from several different suppliers, the visibility to the distinct functions and their details is lowered or completely absent. In such environment, there are more unknowns and less control, which will reflect the approach that should be used in enabling the security. Open RAN is a complex system and can be approached using the principles of system theory. This will help to understand its behavior as defining the components and their relations helps to structure and analyze the problem objectively. The system can be defined as group of interacting or interrelated elements that act according to a set of rules to form a unified whole (Backlund 2000). When there is limited visibility to components and their characteristics, assumptions prevail. When defining the interactions between the components for purposes of security, the trust relationships between the components needs also to be defined. This is where the foundation of security, the trust comes into play.

The concept of trust has been broadly discussed in numerous areas of research including moral and ethics, game theory, decision theory, systems theory, social contract theory and global political science. It has been considered a foundation that everything else is relying and built upon. Ancient Greek philosopher Aristotle has already studied trust and trustworthiness. In researching how humans can reach (ethical) happiness, Aristotle stated the role of trust as “trust is a virtue and along with being trustworthy, it assists a person to achieve eudaimonia” (Irwin 1999). In human-to-human relationships, this can be explained, as social beings, we value character traits such as trust, honesty, and kindness that allow us to live well together collaboratively.

Niklas Luhmann, a German philosopher and one of the prominent scientists in systems theory has extensively studied trust in sociological research and his principal work on trust “Vertrauen” presents the Luhmannian concept of trust, confidence, and familiarity. Luhmann’s research connects trust with familiarity and confidence and explains how those are related to each other (Luhmann 1968). Familiarity, confidence, and trust are all about asserting expectations, however they can be considered different modes, as if they were distinct types of self-assurance (Luhmann 1988). Familiarity is based on clear distinction between familiar and unfamiliar with no need for self-reflection. Confidence relates to situations with contingency and danger, where reflection on pre-adaptive or protective measures is seen useful. With trust there is a clear distinction that it is about making a conscious decision on something, and that decision always comes with the connected risk. Risks emerge only as a component of decision and action (Luhmann 1988). In other words, for Luhmann trust is a decision “in making a decision concerning trust, we must make a distinction: to trust or not to trust” (Luhmann 1979).

Trust and the related terms are conventionally used in broad manner and there is usually significant risk of misunderstanding and ambiguity involved. Therefore, when using trust as a foundation for security, we must ensure that all the ambiguity is dissolved, and that there is complete clarity what is being discussed. This chapter aims to define the central trust related terms that are also used in this research.

Generalized Trust

Generalized trust has been studied widely in various fields and it can be concluded that there are plenty of variations in multiple dimensions. Already a question whether the trust should be generalized or not is not trivial to answer. As a result, what generalized trust means, and whether there are several types of generalized trust, remains largely unknown. On a broad level, generalized trust is considered a general attitude towards the trustworthiness of others, thus differing

from trust (Hardin 2002). In some definitions generalized trust is considered as the optimism and unconditional faith people have in unknown others (Uslaner 2002). Generalized trust is too broad concept for Open RAN security and instead something more precise must be used.

Trust

To understand the role of trust and how trust is used in different use cases, whether it is between humans or machines, we may use a model with the interactions between the entities. System theory is useful for this purpose as it is about how parts of the system (humans or machines) interact to affect the entire system. Infante et al (1997) define a system as hierarchical — a set of interdependent units working together to adapt to a changing environment. Complex system has been described as one made of substantial number of parts that have many interactions (Simon 1996). In a study related to organizational research, complex organization has been defined as a set of interdependent parts, which together make up a whole that is interdependent with some larger environment (Thompson 1967)

Trust is considered a critical and foundational component in enabling secure interactions in machine-to-machine, human-to-human and human-to-machine communications. One could imagine that the meaning of trust is established and so well-known and understood, that there is no need to define it again (Barber 1983). However, trust is a topic that has been extensively researched and has provoked considerable interest, often with heated debate within different areas such as psychology, political science, economics, anthropology, history, and sociobiology (Gambetta 1988). In research on systems theory, it has been studied how the concept of trust would function in the context of advanced systems theory and concluded “trust as a decision, the function and meaning of which is to reduce the complexity of society” (Jalava 2005). Trust as concept is future-oriented but it is built on history, as familiarity is the precondition for trust. There cannot be future without a past (Luhmann 1979) and having a trust without prior knowledge (or evidence), is actually not trust but pure hope.

As concluded above, trust is to reduce complexity in systems. The levels of complexity and use of trust may be different in cases between humans compared to case between machines. It could be expected that the computers or computer systems are less complex that humans, and therefore applying trust would be easier. However, that is unwarranted assumption, and due to vastly different nature and characteristics of humans and computers, we cannot simply apply the concept of trust the same way to the interactions with computers as we do to the interactions with humans (Bursell 2022). Open RAN can be considered as a computer system and therefore the valid trust scenario for this case would be machine-to-machine. This assumption can be made with the caveat that there is a human element in the trust relations of Open RAN, but that the human part is represented by the policy function(s) that can be considered as “machine,” but which is/are acting as trust agent for the human user.

Trust has been extensively discussed in context of trust between the machine entities. Yet there are aspects that are not necessarily trivial or acknowledged even by the experts in the industry. This can be exemplified by looking how trust has been defined by the leading global organizations that work on security. US National Institute of Standards and Technology (NIST) has several definitions for trust in its different technical proceedings. Their special publication on a profile for US Federal Key Management Systems defines trust as follows (NIST SP 800-152 2015):

> A characteristic of an entity that indicates its ability to perform certain functions or services correctly, fairly, and impartially, along with assurance that the entity and its identifier are genuine.

Another special publication from NIST that addresses the Supply Chain related risks and how those should be managed by the organizations has a different way of describing trust (NIST SP 800-161r1 2022):

> The confidence one element has in another, that the second element will behave as expected.

To demonstrate further that even a single organization such as NIST is not aligned with their terminology, let us take one more definition for trust. NIST publication guiding to secure web services defines the trust as (NIST SP 800-95 2007):

> The willingness to take actions expecting beneficial outcomes, based on assertions by other parties.

The main point to observe these definitions from NIST is to highlight that even a single organization is not internally aligned, but moreover has conceptually quite different descriptions. The two other descriptions are both based on one party (trustor) to expect something from the other party (trustee) while the first one approaches the concept completely differently by defining it an absolute characteristic for the trustee to have an ability to behave correctly, fairly, or impartially. This is very different from the others and deviates from the principles discussed before for human-to-human context, which are defined as relative and not absolute.

Bursell (2022) has studied trust from different fields of research, including computer systems, (trusted) computing, Socio-philosophical sciences, mathematics, and cryptography. He has considered the vastly different definitions from the industry and academia and approached the trust from systems theory point of view. The basis for his definition of trust is the one defined by Gambetta (1988) in his essay on cooperative relations between humans:

> Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.

While Gambetta's definition is more valid for human-to-human relations, Bursell focuses more on machine-to-machine relationships and has simplified the definition to better suit computer systems. As clear differences in their definitions, Bursell has defined term agent to be someone acting on behalf of other entity while Gambetta uses it as an actor taking an active role in an interaction. The commonality however is the one (trustor) about setting expectations on the other (trustee) before any actions are performed, based on the information collected beforehand. Bursell (2022) has ended with the definition that suits computer systems (and Open RAN) extremely well:

> Trust is the assurance that one entity holds that another will perform particular actions according to specific expectation.

The definition above is largely aligned with the two definitions presented before from NIST and in its compact form is very well suited for computer systems. There are few observations that can be made. The scope of the definition is narrowed to "particular actions". This indicates that for trust, there is always a clear context or scope. Trust has very precise limits, and the assurance in this case is only valid for that specific well-defined action and nothing more. Trust is also defined as relative terms - one is expecting another to do something. Trust is asymmetric – while A may expect B to perform according to specific expectations, it does not mean anything how B expects A to behave. The last related matter is not an observation per se, but moreover a consequence of the definition. As one is expecting another to perform something according to specific way, the assurance is built upon information collected beforehand. Outside of the actual definition for trust, it is well known that the validity of information is time-dependent and its value decreases as time passes. This influences the assurance part of the trust, where the assurance is based on a priori information. The context of time needs to be considered when forming the decision on assurance. This is also considered by Bursell in his definition and corollaries of trust.

For this project, trust and security in Open RAN, the trust definition below is followed:

> Trust is the assurance that one entity holds that another will perform particular actions according to specific expectation (Bursell 2022)

The other remark that should be noted in context of Open RAN is that trust is always based upon decision. To trust is a decision and the related residue risk is accepted with the decision.

Zero Trust

As technologies related to communications infrastructure are evolving with a speed and magnitude that are referred as revolutionary, the old security approaches have reached their boundaries. The mobile networks based on hierarchical point-to-point architecture, deployed as single vendor solutions have been conventionally protected with perimeter-based security model. Considering the advancements in technology development and the added cost driven operability models, the current networks based on virtualization, disaggregation and multi-vendor model are more complex in nature and have more players involved. This results in significantly increased attack surface (Liyanage et al. 2023), that is calling for new approach in security.

Zero-Trust is a security approach whose origins are debated. As a term, it was already introduced by in Stephen Marsh in his doctoral thesis on Formalizing Trust as a Computational Concept (Marsh 1994), however with a meaning different from what is considered for Zero-Trust as a security approach now. In his description, trust could be presented with an equation capturing the values of trust as $-1 \leq Tx(y) < +1$, which represents the amount of trust x has in y. Zero trust is the case where one does not have trust in another e.g. if the trustor does not know the trustee or if the trustor is impartial with respect to the trustee. There is plethora of different descriptions for zero-trust and as a trending term it is used loosely and broadly within the industry.

There are also slogans used by the industry players to advertise their zero-trust concept – “trust but verify”, “never trust, always verify”, “trust is good, control is

better” or “Assume nothing, believe no one, Check everything”. These all demonstrate the trendiness of the zero-trust, and if we extend our research beyond technology, we can find out that in fact zero-trust had been used also in global politics, specifically in context of nuclear disarmament during the cold war by the US president Ronald Reagan (Watson 2011)

US NIST has worked extensively on zero-trust and created a set of different documents on zero-trust principles, framework, and architecture. They have defined zero-trust as an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources (NIST SP 800-207 2020). In the same zero trust architecture document, they also give the following definition:

> Zero trust assumes there is no implicit trust granted to assets or user accounts based solely on their physical or network location (i.e., local area networks versus the internet) or based on asset ownership (enterprise or personally owned).

While the above NIST definition is clear it is also adapted and endorsed by several other organizations (ATIS 2023) and industry players (Ericsson 2023). As relying merely on single definition may not be sufficient, NIST has issued a list of seven basic tenets to be used to adhere with zero trust architecture (NIST SP 800-207 2020):

1. All data sources and computing services are considered resources.
2. All communication is secured regardless of network location.
3. Access to individual resources is granted on a per-session basis.
4. Access to resources is determined by dynamic policy.
5. The enterprise monitors and measures the integrity and security posture of all owned and associated assets.
6. All resource authentication and authorization are dynamic and strictly enforced before access is allowed.
7. The enterprise collects information about the current state of assets, network infrastructure and communications and uses it to improve its security posture.

Zero trust approach is a concept that is widely misunderstood, and the very typical misconception has been that there is no trust in zero trust. This is not correct as trust plays very significant role in the concept, as validation results are used to decide if the other party is trusted or not. From terminology point of view, it would be more appropriate to use the term “explicit trust” instead. Another issue typically overlooked or completely ignored with the zero trust is the need for continuous monitoring. Setting micro-perimeters and enforcing a strong identity and access management is not sufficient, but the resources or assets needs to be continuously monitored to ensure the integrity and security posture of the system.

Distrust

It is common belief and understanding, that distrust is equal with no trust. While that could sound right interpretation by quick thinking, that is however not correct. When discussing zero trust we used the mathematical notation for trust i.e. $-1 \leq Tx(y) < +1$ (Marsh 1994), where zero trust would have value 0. Distrust would gain value -1 and it would mean that there is an explicit, conscious decision, usually based on evidence not to trust some entity. In case of zero trust, the trust is formed upon verification, and it may turn to be trusted or not trusted.

Trustworthiness

In human-to-human context, when using the common layman terms, a person is considered trustworthy when there is belief that the other party is honest (with the intentions) or able to be relied on as honest or truthful. A classic example could be a neighbor in whose hands you would leave a spare key to your house. When laying a foundation to security, we must be more precise with the definition of trustworthy and trustworthiness. Gambetta (1988) has defined for a trustworthy person, that we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him. When considering trustworthiness in industrial systems with computers, Connett et al. (2018) have studied the definitions of trustworthiness from four industry perspectives: how the system is

designed, how it behaves, how it is architected, and built and, how it may be disrupted (Connett et al. 2018). In their review, from a design point of view, the trustworthiness has been defined as “demonstrable likelihood that the system performs according to designed behavior under any set of conditions as evidenced by characteristics.” When taking a different angle and looking the question from the behavior point of view, one of the definitions for trustworthiness comes as: “trust is one’s willingness to be vulnerable to another based on the confidence that the other is benevolent, honest, open, reliable and competent”. One of the aspects related to trustworthiness is what we expect the entity to perform, and what we do not expect. From the zero-trust approach point of view, related to anomalies and their detection, the trustworthiness can be stated as: “system does what it is required despite environmental disruption, human user and operator error, and attacks by hostile parties and no other things” (Schneider et al. 1999).

As discussed, trust is the foundation of security and trustworthy systems are essential to keep the digital societies running. In USA, this has been raised as a priority topic and therefore the President signed an Executive Order to Improve the Nation’s Cybersecurity. The Executive Order frames the need for better protection with the change in the security environment and calls for actions to improve the efforts to identify, deter, protect against, detect, and respond to these actions and actors (EO14028 2021). The EO sets requirements on trust and trustworthiness in comparison to the risks by stating: “The trust we place in our digital infrastructure should be proportional to how trustworthy and transparent that infrastructure is and to the consequences we will incur if that trust is misplaced.”

To respond to the call made by the EO and the need to provide systems that can be trusted, the US NIST have worked and produced a report on Engineering Trustworthy Secure Systems. This document is truly relevant for communications infrastructure and in addition to providing design guidelines, they also motivate the trustworthiness and explain what it means in this context. It is delightful to observe that trustworthiness defined by NIST is built upon assurance and that they are extremely precise in definition the conditions: “Assurance is the grounds for

justified confidence that a claim or set of claims has been or will be achieved" (NIST SP 800-160 2022). Further they quality the evidence and bind it to the associated risk level: "Justified confidence is derived from objective evidence that reduces uncertainty to an acceptable level and, in doing so, reduces the associated risk" (NIST SP 800-160 2022). Their definition for the trustworthiness is carefully considered and fits perfectly for this project.

> A trustworthy entity is one for which sufficient evidence exists to support its claimed trustworthiness. Thus, trustworthiness is the demonstrated ability and, therefore, the worthiness of an entity to be trusted to satisfy expectations, including satisfying expectations in the face of adversity. Since trustworthiness is something demonstrated, it is based on evidence that supports a claim or judgment of an entity being worthy of trust (NIST SP 800-160 2022).

4 TRUST AND SECURITY IN OPEN RAN

Mobile Networks have evolved from their original introduction and the current 5th generation networks are considered as nervous systems of the digital societies. Their evolution has been remarkable, and it has changed the industries, businesses, and civil life of the citizens. And yet the development is accelerating as we are in the middle of fast-paced technological innovation bringing changes that may be revolutionary.

One of the changes that started with the 5th Generation networks has been Open RAN - a paradigm shift in the design and operation of the networks. Open RAN refers to the disaggregation of Radio Access Network (RAN) functions, using open interfaces between elements, and enabled by software virtualization. This allows implementation and deployment based on vendor-agnostic hardware and software. Furthermore, Open RAN includes intelligence through artificial intelligence / machine learning (AI/ML) to optimize the network performance and the end user experience. The Open RAN approach has been led by an industry

group called O-RAN ALLIANCE that was launched 2018 to promote the openness, automation, intelligence, and interoperability of the networks. The driver for the new approach has been the diversity enabled by the openness to boost for the faster innovation and lower the total cost of ownership. Security has been one of the initial drivers for Open RAN as western governments led by US have been concerned about the global dominance of Chinese infrastructure vendors and strong dependence on them. Open RAN was initiated as a vehicle to promote the supplier diversity and to create the opportunities for the domestic vendors.

4.1 Enabling Trust step-by-step

There are several benefits associated to Open RAN and it has been also suggested to be one of the building blocks for future 6G systems. As overall, there is greater flexibility in selecting and optimizing the network components and services. The architecture has been defined from the cloudification perspective, which facilitates smooth transfer of the network functions into the cloud, catering greater efficiency and scalability. The interoperability with the new interfaces and O-RAN ALLIANCE specified profiles increases flexibility in the market and potentially leads to more competition in the telecoms supply chains. The open and disaggregated network architecture provides smooth transition path to 6G as it allows better re-use of the multi-vendor network components, which could further impact positively in the total cost of ownership.

While there are benefits in the Open RAN architecture, there are also several challenges. The use of open interfaces and managing multiple vendors and their software development lifecycle processes will pose a security challenge. Further the overall complexity of the disaggregated cloud-native network and the integration of different vendor components in a mix and match approach may appear significant challenge from the performance and security validation point of view.

When considering the security of Open RAN, the architecture and how it is deployed must be understood. The logical architecture that is already presented in section 2.1 describes the network functions and the interfaces between them.

While that is also useful, from the trust and security point of view it is essential to see the whole picture, including the host environment where the O-RAN system is deployed. Figure 11 below illustrates an exemplary deployment of O-RAN in cloud-native environment.

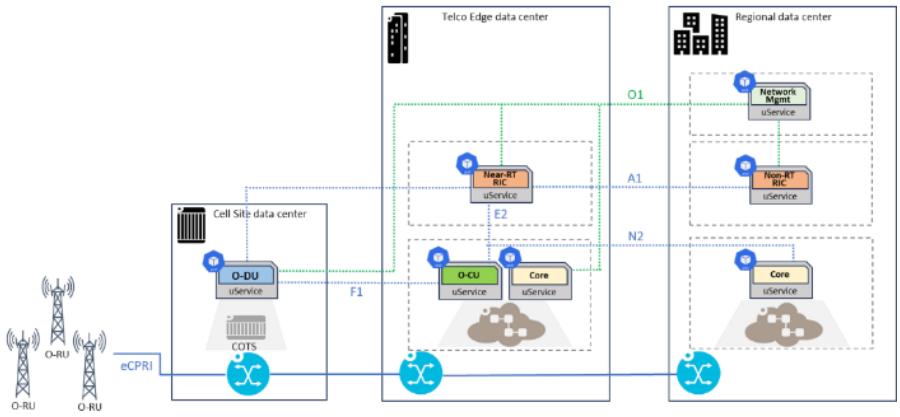

Figure 11. O-RAN in cloud-native deployment

Deployment according to figure 11 shows that the principles of Open RAN are included: disaggregation, cloudification, automation and intelligence. O-RU radio units deployed close to end users at the cell sites conventionally consist of purpose-built hardware and they are connected to virtualized and cloudified network functions over the open fronthaul interface. The virtualized part of the Open RAN start from the distributed unit O-DU that is running the higher part of Layer 1 and the Layer 2 processes. As these are performance critical processes, the O-DU functions are deployed closer to the radio units e.g. at cell site datacenters. The centralized unit O-CU is deployed at edge cloud datacenter and could be running in public cloud. The O-CU is co-located with the distributed core network function and the Near-real time Radio Intelligent Controller. In more centralized location at the regional cloud there are Core Network, Network Management, and the non-real time RIC functionalities.

The different O-RAN network functions constructed as microservices are deployed in different data centers and deployed as containers in Kubernetes pods. Kubernetes as the container management and orchestration system is the one that makes decisions where to deploy each container. The deployment location

decision is based on several factors such as resource availability, workload constraints and affinity/anti-affinity rules. In our example open RAN deployment, the functions with higher real time processing demands are deployed closer to the radio units (O-RU) and their deployment may be guided by setting constraints for their desired location.

O-RAN network can be trusted only if there is trust for all the components of the system. Considering all the different functions, components, tenants, vendors, and suppliers, achieving complete trust becomes a challenge. To tackle the problem of complexity, a simplified step-by-step approach can be taken to examine building the trust, starting from basic combination. In Open RAN, the network functions are run as virtualized workloads, instantiated as containers in the target hosts known as servers. To trust the network function, there must be trust both on the container and on the target host (server). Using a mathematical expression, the research from Marsh (1994) who studied computational methods to formalize trust, can be used. He has defined several notations that can be used to describe trust. The basic equation for a trust with a specific condition can be written as:

$$T_x\,(y, \alpha) \in [-1, +1] \quad (1)$$

which expresses the trust x has on y under the condition of α, receiving values between -1 and +1.

The same equation can be also written as:

$$-1 \leq T_x\,(y, \alpha) \leq 1 \quad (2)$$

The above equations (1) and (2) use the value range between -1 and +1, where -1 maps to complete distrust while +1 stands for complete trust, which is also referred as blind trust. For our case to define trust on container and server, we use binary notation 0 for no trust and 1 for trust. For further simplification we also refrain of using the condition. The equation becomes:

$$T_x\,(y) \in [0,1] \quad (3)$$

which expresses the trust x has on y, receiving values between 0 and 1.

When considering the first problem, trust on container and trust on server, a traditional Four Field Matrix analysis with two dimensions can be used - trust on server and trust on container as illustrated in Figure 12.

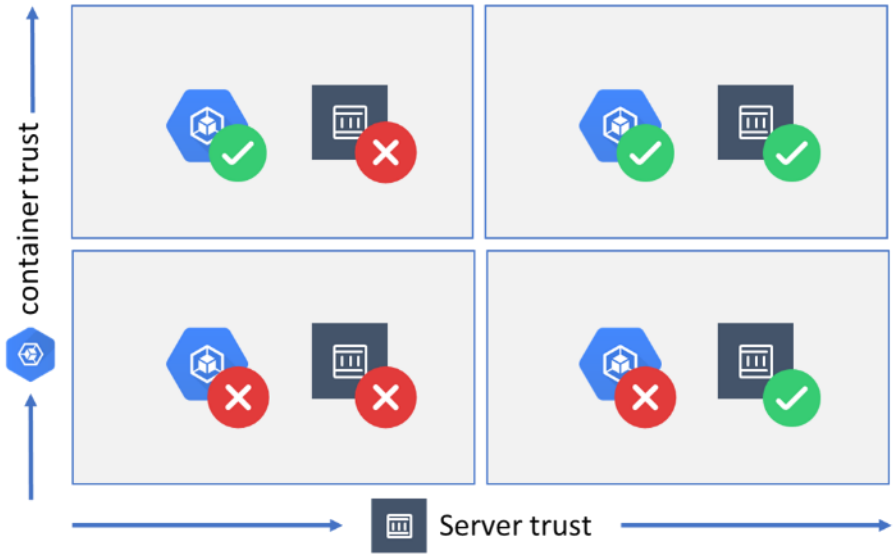

Figure 12. Trust combinations for the container and for the server

In the Four Field Matrix, the dimensions are server trust on x-axis and the container trust on y-axis. This gives four different value pairs of trust:

$$T_S = 0 \ \wedge \ T_C = 0 \quad (4)$$

expressing that there is no trust on server and no trust on container.

$$T_S = 0 \ \wedge \ T_C = 1 \quad (5)$$

expressing that there is no trust on server, but that there is trust on container.

$$T_S = 1 \ \wedge \ T_C = 0 \quad (6)$$

expressing that there is trust on server, but that there is no trust on container.

$$T_S = 1 \ \wedge \ T_C = 1 \quad (7)$$

expressing that there is trust on server, and that there is also trust on container.

Considering a system with considerable number of functions and trust relationships, there could be different strategies in deciding whether to trust the system,

implemented as part of the system security policies. In this example case, there are four value pairs for trust expressed in (4), (5), (6) and (7), and there is only one combination that has trust on both the server and the container. Conversely this indicates that there are three pairs, where there is at least one entity with no trust. The worst combination is the one with no trust on server nor trust on the container. There are potential implications related to different combinations.

For a case with trust on the container, but no trust on the server one could if the combination still could be trusted, assuming that the server is operated by a trusted company with good reputation. Considering the potential threats exposed by the non-trusted server, different attack vectors that are considered valid for our target environment can be looked. MITRE Corporation is an US not-for-profit organization that is dedicated to advance the cybersecurity practices and their MITRE ATT&CK framework is a knowledge base of different adversary tactics and techniques based on observations from the real-world cases.

For the cloud-native deployment MITRE ATT&CK Matrix for Enterprise can be used as a basis for analysis. The ATT&CK Matrix knowledge base collects 14 different tactics with altogether 234 different techniques assumed valid for a typical enterprise case. The tactics included are Reconnaissance, Resource Development, Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Command and Control, Exfiltration, and Impact (MITRE 2024c). While the ATT&CK Matrix is generic for several cases, not all the listed techniques are valid attack vectors for a server. However, the adversary only needs to find one way to get access.

In addition to ATT&CK framework that collects different techniques, concrete proof points from the known cyberattacks against compromised hosts can be considered. With a very quick search already a few cyber incidents against cloud hosting providers can be found: Azero, CloudNordic, Tietoevry and Cognizant as examples (Norton 2023; Cybernews 2024). There are several ways to get initial access and to compromise the Cloud host, but typically the conventional cyber kill chain (Hutchins et al 2011) is followed. Once there is hold on the host, there

are several ways to attack and exploit the services hosted on the same server. One example attack type is the Server-side request forgery (SSRF) where the local processing may allow by-passing the protection and allow access to resources normally not authorized. The reason for this could be that the protection may be performed by another function in front of the server (PortSwigger 2024). This would imply that there is effectively an implicit trust for the requests that come from the local machine.

When considering the security of the containers deployed on the cloud, the MITRE ATT&CK Cloud and the Containers Matrixes can be used. The container matrix has been designed from containers technology point of view and it collects nine different tactics and 37 techniques that can be used by the adversaries (MITRE 2024b). Containers themselves can be attacked and compromised several ways and then exploited later. One significant and heavily increasing cyberattack type is the supply chain attack. The adversary may compromise the CI/CD environment and insert the malware in the source code repositories that are used later when building the container images. Once the compromised container is deployed and instantiated on a host that also has other containers running, the compromised container may start its operation. After getting initial access and foothold in the target, the other typical actions followed are lateral movement in the system and privilege escalation. With the elevated permissions in the system, the compromised container may then use several attack vectors, including stealing the secrets, binding, and escalating the existing roles and impersonating the existing entities in the cluster. A compromised container with the root rights may cause severe damage to other existing services and for example cause a starvation attack by consuming all the resources in the user space of the host.

The above examples of compromised host and compromised container highlight the importance of accepting only hosts and containers for which a (knowing) trust decision has been made.

4.2 Trust relations in Open RAN system

Using trust of a container and trust of a server is simplified example to demonstrate the trust relationship and how trust is built. In a full-scale Open RAN system establishing the trust is much more complex process with a plethora of distinct functions each having different tasks and purposes. In section 3 it was discussed how trust is defined in systems theory point of view and how it can be applied in computer systems. The concept of generalized trust referring as general attitude and unconditional faith on something was also discussed, while in case of a complex Open RAN system one must be precise what is expected from the other function (trustee). For example, it can be expected that the function behaves correctly, fairly, and impartially, along with assurance that the entity and its identifier are genuine. Or following the definition for trust in this project:

> Trust is the assurance that one entity holds that another will perform particular actions according to specific expectation.

This means that to make a trust decision, it needs to be known and defined in detail what is expected from the function (trustee). In this context, it needs to be understood what functions there are, what their roles are, how they are interacting with other functions, and finally what are the expected behaviors. This means that trust relationships of a system must be defined.

This can start by examining the Open RAN system from the O-RAN logical architecture presented in Figure 13.

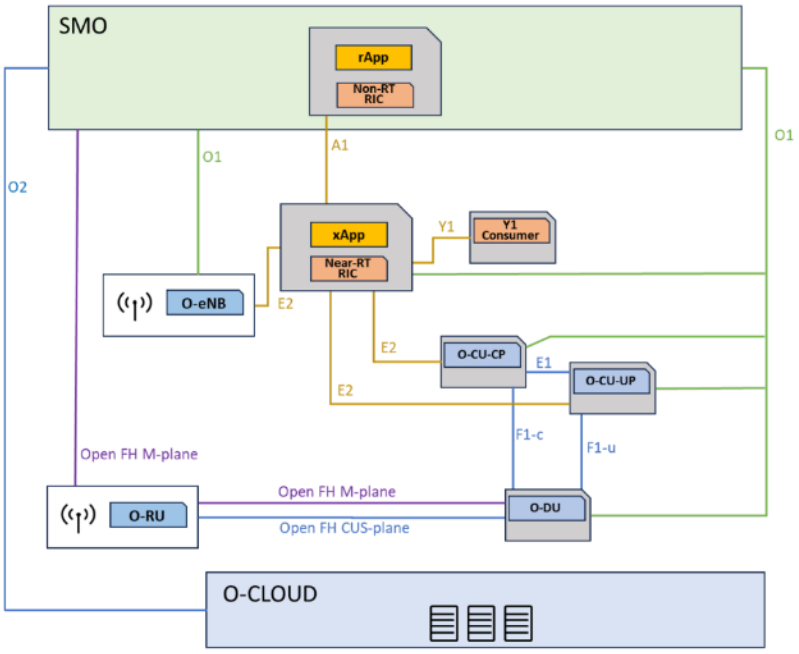

Figure 13. O-RAN logical architecture

O-RAN aim is to define the next generation Radio Access Networks that are open, intelligent, virtualized, and fully interoperable. To support this, a cloud infrastructure that can be managed and where to deploy the virtualized network functions is needed.

O-CLOUD

The O-RAN virtualized functions are deployed on the virtualization infrastructure and in the O-RAN architecture this function is O-CLOUD. O-CLOUD is a cloud computing platform consisting of the physical infrastructure nodes and the supporting software components such as Operating system and virtualization related components. O-CLOUD is connected to Service Management and Orchestration (SMO) via O2 interface, which supports the management of the infrastructure itself (O2ims) and the management of the workloads deployed in O-CLOUD (O2dms). In building the trust relationship model the following expected functionalities related to O-CLOUD can be considered:

- O-CLOUD exposes the O2 interface for cloud resources and workload management to the Service Management and Orchestration (SMO)
- O-CLOUD exposes Acceleration Abstraction Layer (AAL) interface for the hosted workloads for hardware accelerator management.

- O-CLOUD offers hosting environment for the virtualized network functions.
- O-CLOUD exposes O-CLOUD notification interface towards the hosted workloads, for example to provide PTP synchronization information.
- O-CLOUD offers infrastructure inventory services for the SMO.
- O-CLOUD offers infrastructure monitoring services for the SMO.
- O-CLOUD offers deployment management services based on Kubernetes for the SMO.

The bulleted list describes how O-CLOUD is expected to interact with the other surrounding functions. The expected behavior and related transactions can be described using the declarative programming language Prolog, which is well suited to describe the logic in or of a system. Prolog allows to define logical rules which can be presented as:

$$is\_trusted(X, Y) :- Function1(X), Function2(Y) \quad (8)$$

expressing that Function X is trusted by Function Y

Defining the rules for the O-CLOUD related expected functionalities, the following is obtained:

Offers_workload_mgmt (O-CLOUD, SMO) (9)

Offers_acceleration_hw_mgmt (O-CLOUD, NF) (10)

Offers_Infra_Inventory_service (O-CLOUD, SMO) (11)

Offers_Infra_mgmt_Service (O-CLOUD, SMO) (12)

Offers_hosting (O-CLOUD, O-DU) (13)

Offers_hosting (O-CLOUD, O-CU-UP) (14)

Offers_hosting (O-CLOUD, O-CU-UP) (15)

Offers_hosting (O-CLOUD, Near-RT RIC) (16)

Service Management and Orchestration (SMO)

The Service Management and Orchestration (SMO) is a functionality in the O-RAN Architecture for managing the Radio Access Network domain. The SMO is defined based on the Service Based Architecture (SBA) and the key services of the SMO are the FCAPS management of the O-RAN Network Functions, the RAN optimization with the Non-Real-Time Radio Intelligent Controller (Non-RT RIC), the management of the C-CLOUD infrastructure and the orchestration of the workloads. These services are provided through A1, O1 and O2 interfaces between the SMO and the O-RAN Network Functions. The following expected functionalities are related to SMO:

- SMO manages O-CLOUD infrastructure resources via O2 interface.
- SMO orchestrates the workloads in the O-CLOUD via O2 interface.
- SMO manages the workloads in the O-CLOUD via O2 interface.
- SMO manages the faults of the O-RAN Network Functions.
- SMO manages the performance of the O-RAN Network Functions.
- SMO manages the configuration of the O-RAN Network Functions.
- SMO may manage the O-RU via FH M-plane interface in case of hybrid management mode.

Using the programmable logic notation, the following is obtained:

Manages_cloud_infra (SMO, O-CLOUD) (17)

Orchestrates_workloads (SMO, NF) (18)

Manages_workloads (SMO, NF) (19)

Manages_faults (SMO, NF) (20)

Manages_perf (SMO, NF) (21)

Manages_config (SMO, NF) (22)

Manages_O-RU_in_hybrid (SMO, O-RU) (23)

Non-Real-Time RIC

The Non-Real-Time Radio Intelligent Controller is a functionality internal to the SMO and it supports optimization of the Radio Access Networks by policy-based guidance. It can also use AI/ML training and inference to provide the ML models and enrichment information for the Near-RT RIC function. The non-RT RIC can use other services offered by the SMO such as data collection or provisioning. The non-RT RIC consists of the non-RT RIC framework that connects to Near-RT RIC via A1 interface and exposes services to rApps via R1 interface, and of the Non-RT RIC Applications called rApps that are consuming the Non-RT RIC services and supporting the optimization of the RAN. Non-RT RIC is managed by SMO via unspecified interface. The following functionalities are expected for the non-RT RIC:

- Non-RT RIC exposes RAN optimization services to rApps via R1 interface.
- Non-RT RIC uses A1 interface to send RAN optimization related policies to Near-RT RIC.
- Non-RT RIC may use data analytics and AI/ML training and inference.
- Non-RT RIC may collect data from the O-RAN Nodes via O1 and O2 interfaces.
- Non-RT RIC may send enrichment information to the non-RT RIC.
- Non-RT RIC may consume services offered by the SMO.

With prolog, the following dependencies are obtained:

Offers_RAN_opt_services (non-RT, rApp) (24)

Sends_RAN_policies (Non-RT, Near-RT) (25)

Consumes_data (Non-RT, NF) (26)

Consumes_data (Non-RT, O-CLOUD) (27)

Sends_data (Non-RT, AI/ML) (28)

Receives_AI/ML_model (Non-RT, AI/ML) (29)

Sends_AI/ML_model (Non-RT, Near-RT) (30)

Is_managed (Non-RT, SMO) (31)

Near-Real-Time RIC

Near-Real-Time Radio Intelligent Controller is an O-RAN Network Function that allows near real-time control and the optimization of the O-RAN Radio Functions. The Near-RT RIC hosts xApps that use the E2 interface to collect information from the O-RAN Radio Functions (E2 Nodes) and to provide controls to the O-RAN Radio Functions. The controls provided by the Near-RT RIC are steered by the policies received from the non-RT RIC over the A1 interface. Near-RT RIC may provide radio analytics information to the consuming function via the Y1 interface. Near-RT RIC relates to the following functionalities:

- Near-RT RIC hosts xApps.
- Near-RT RIC exposes E2 services to xApps over Near-RT RIC API
- Near-RT RIC exposes Y1 services to Y1 consumers.
- Near-RT RIC receives policies from non-RT RIC via A1 interface.
- Near-RT RIC receives AI/ML models from the non-RT RIC
- Near-RT RIC collects data from the O-RAN radio functions.
- Near-RT RIC is managed by SMO over O1 interface.

Using the Prolog notations, the following is obtained:

Offers_hosting (Near-RT, xApp) (32)

Offers_RAN_opt_services (non-RT, xApp) (33)

Offers_RAN_information (Near-RT, Y1 consumer) (34)

Reveives_RAN_policies (Near-RT, Non-RT) (35)

Receives_AI/ML model (Near-RT, Non-RT) (36)

Consumes_data (Near-RT, NF) (37)

Consumes_data (Near-RT, NF) (38)

Is_managed (Non-RT, SMO) (39)

O-CU-CP

O-CU-CP is the control plane part of the O-RAN Centralized Unit (O-CU) function, whose main function is to control the PDU session and the connection between the user device and the network (3GPP TS 38.401, 13). O-CU-CP is a logical node that is hosting the Radio Resource Control (RRC) functionality and the control plane part of the Packet Data Convergence Protocol (PDCP) (3GPP TS 38.401, 10). O-CU-CP is connected to the O-DU over the F1-c interface, to the O-CU-CP via the E1 interface, to the Near-RT RIC over the E2 interface, and it is managed by the SMO over the O1 interface. O-CU-CP is connected to other O-CU-CP or gNB-CU over the X2-c or Xn-c interfaces, and it is connected to the Next Generation core (NGC) over the NG-c interface.

Using the Prolog notations, following relationships are obtained:

Is_managed (O-CU-CP, SMO) (40)

Runs_on (O-CU-CP, O-CLOUD) (41)

Sends_data (O-CU-CP, Near-RT) (42)

Receives_policies (O-CU-CP, Near-RT) (43)

Coordinates_control (O-CU-CP, O-CU-UP) (44)

Coordinates_control (O-CU-CP, O-CU-CP) (45)

Controls (O-CU-CP, O-DU) (46)

Is_Controlled (O-CU-CP, NGC) (47)

O-CU-UP

O-CU-UP is the user plane part of the O-RAN Centralized Unit (O-CU) function, whose main function is to carry the user data through the access network (3GPP TS 38.401, 13). It is a logical node that hosts the user plane part of the PDCP and SDAP protocols and it is connected to a O-CU-CP over the E1 interface, to a O-DU over the F1-u interface, to a Near-RT RIC via the E2 interface, and it is managed by an SMO via the O1 interface. O-CU-UP is connected to a Next Generation core network over the NG-u interface and to another O-CU-UP via the Xn-u or X2-u interfaces.

The following relationships can be derived for O-CU-UP:

Is_managed (O-CU-UP, SMO) (48)

Runs_on (O-CU-UP, O-CLOUD) (49)

Sends_data (O-CU-UP, Near-RT) (50)

Receives_policies (O-CU-UP, Near-RT) (51)

Coordinates_control (O-CU-UP, O-CU-CP) (52)

Coordinates_control (O-CU-UP, O-CU-UP) (53)

Sends_data (O-CU-CP, O-DU) (54)

Receives_data (O-CU-CP, O-DU) (55)

Receives_data (O-CU-CP, NGC) (56)

O-DU

O-DU is an O-RAN Distributed Unit function, a logical node hosting the RLC and MAC layers as well as higher parts of the PHY layer (3GPP TS 38.401, 10). The O-CU-CP controls the operation of the O-DU, and it receives the data from the O-CU-UP. O-DU may receive policies from and send data to the Near-RT RIC over the E2 interface. O-DU manages the O-RU via the Open FH M-plane and controls the O-RU via the Open FH CUS-plane. The O-DU is synchronized with the O-RU via the Open FH CUS-plane. The O-DU sends data to and receives data from the O-RU. The O-DU is managed by the SMO via the O1 interface.

The following relationships are valid for the O-DU:

Is_managed (O-DU, SMO) (57)

Runs_on (O-DU, O-CLOUD) (58)

Receives_policies (O-DU, Near-RT) (59)

Sends_data (O-DU, Near-RT) (60)

Is_controlled (O-DU, O-CU-CP) (61)

Sends_data (O-DU, O-CU-UP) (62)

Receives_data (O-DU, O-CU-UP) (63)

Receives_data (O-DU, O-RU) (64)

Sends_data (O-DU, O-RU) (65)

Manages (O-DU, O-RU) (66)

Configures_Synch (O-DU, O-RU) (67)

Synchronizes (O-DU, O-RU) (68)

Controls (O-DU, O-RU) (69)

O-RU

O-RU is an O-RAN Radio Unit, which is a logical node hosting the lower PHY layers. The O-RU is managed by the O-DU over the FH M-plane interface and in hybrid management mode, it may be also controlled by the SMO. An O-RU receives synchronization configuration from the O-DU or alternatively it may receive the synchronization from the O-DU. O-RU is controlled by the O-DU and it receives data from and transmits data to O-DU. O-RU is not typically hosted in O-CLOUD but rather it is based on purpose-built hardware.

The following Prolog notations can be defined for the O-RU:

Is_managed (O-RU, O-DU) (70)

Is_managed (O-RU, SMO) (71)

Runs_on (O-RU, gNB-HW) (72)

Sends_data (O-RU, O-DU) (73)

Receives_data (O-RU, O-DU) (74)

Is_controlled (O-RU, O-DU) (75)

Receives_Synch_conf (O-RU, O-DU) (76)

Receives_Synch (O-RU, O-DU) (77)

Summary on the trust relationships

By analyzing the whole O-RAN Architecture and the expected functionalities between the O-RAN Network Functions, 77 different notations are obtained. These notations describe in detail what is expected behavior in system and between the functions and those can be used to characterize the trust relationship between the functions. As example, if it is stated that O-RU receives synchronization from the O-DU, it does matter very much from system performance point of view and even a minor deviation from expected value could cause severe degradation of the system performance. It is essential that there is trust between the O-RU and O-DU that O-RU can rely on signal received from the O-DU.

4.3 Trust ontologies of cloud-native Open RAN

Previous section provided detailed description of all the Open RAN functionalities and how they are connected to different other functions. When deploying such a system in actual target environment with the hardware and software components, the trust relations will expand and get more complex. Therefore, when designing a security solution for such a system, it is crucial to understand in detail how the

different functions are connected to each other. This is where trust ontologies come in the picture.

The purpose of a trust ontology is to illustrate the relationships of the different entities in an entire system. Figure 14 presents a cloud-native Open RAN system deployed on bare metal server.

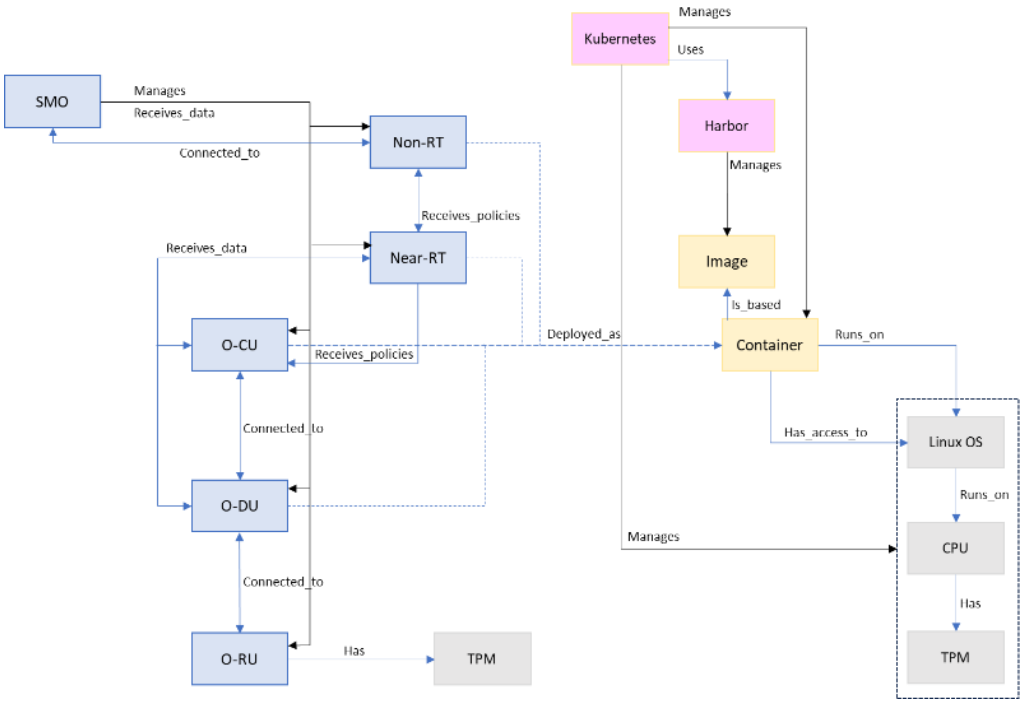

Figure 14. Cloud-native Open RAN running on bare metal.

The trust ontology figure illustrates distinct functions and their relationships and for clarity and visualization purposes, the relations between the functions are abstracted to a higher level and do not describe the interactions in the same detailed level as in previous section. As example, the O-CU is *connected_to* an O-DU that is further *connected_to* an O-RU. The O-CU is *managed* by and is *sending_data* to an SMO. An O-CU is *sending_data* to and *receiving_policies* from a Near-RT RIC and is *deployed_as* a container. The container *is_based* on an image that is *managed* by a harbor *used* by Kubernetes. The container is *running_on* a host on top of Linux OS, that further *runs_on* a CPU that *has* a TPM. The host and container are *managed* by Kubernetes. The host *runs* also other containers representing other functions such as O-DU, Near-RT RIC et cetera, *based* on images *managed* by Harbor *used* by Kubernetes. Making a similar

analysis for all the functions and walking through all the paths and dependencies in the trust ontology, it can be observed that nearly all the functions are either directly or indirectly interconnected, and that there may be multiple dependencies and interconnections between the distinct functions. From the attack surface point of view as discussed in section 4.1, there are multiple strategies and tactics as described in the MITRE ATT&CK matrixes that can be used in attacking the system, each resulting in with a different threat to a security of the system. As it is extremely difficult to analyze and compare the severity of potential future attacks in distinct parts of the system beforehand, for simplicity any attack can be considered equally severe for the system and therefore unacceptable. This assumption will have a major impact on the trust model as it suggests that there must be trust for each function where there is a dependency.

Taking the O-CU again as an example, to trust the O-CU there must be trust on the container(s) that realize the O-CU functionality as well as on the host server where the O-CU container is being run. Using the Prolog notation, the following is obtained:

trusted (O-CU):- trusted (container), trusted (host) (78)

However the above does not take into account all the dependencies of the O-CU as it omits that it is managed by an SMO, receives policies from a Near-RT RIC which receives the related policies from a Non-RT RIC, is connected to an O-DU, is based on an image that is managed by Harbor that is controlled by Kubernetes, and finally it is running on Linux OS that is running on a CPU. Based on this, the equation becomes much more complex:

trusted (O-CU):- trusted (container), trusted (Linux OS),
trusted (CPU), trusted (image), trusted (Harbor),
trusted (Kubernetes), trusted (SMO), trusted (non-RT),
trusted Near-RT), trusted (O-DU), trusted (O-RU) *(79)*

For completeness, the same exercise should be run for the rest of the functions as well. However, the analysis would be in all aspects like the one of O-CU and therefore it is not repeated for the rest of the functions. Instead, the analysis continues by looking what could happen with untrusted functionality in Open RAN system and what could be the consequences.

4.4 Vulnerabilities and threats

Adversaries are utilizing any route that gains them the access to and control of the target system and the attacks via the system management functions are more commonly used. Telecommunication networks are typically only accessible from the management systems and as the human users are usually the weakest link in cybersecurity, one of the early tasks in a kill chain is to get the user credentials for a system management account for example with a phishing campaign. The SMO is the top-level system for managing the Open RAN and getting a compromised SMO service account would give the adversaries an access to administer the entire system. A compromised SMO account would equip the attacker with a rich set of vectors to implement severe attacks and would further help in hiding those efficiently. An O-RAN security research from Liyanage et al. (2023) indicates that the security of SMO is essential for the O-RAN security and privacy and lists several SMO related attacks both on SMO external and internal interfaces. This highlights that it is critical to have a trust on SMO as untrusted SMO would mean a system whose confidentiality, integrity and availability were all uncertain and potentially endangered.

The non-RT RIC and near-RT RIC are functionalities that provide open platform for 3rd party applications, and they are designed to enable programmability and intelligent control of the radio access network. These 3rd party applications consume services offered by the Non- and Near-RT RIC over standardized APIs. The applications use radio network related data that is received from the O-RAN radio functions and exposed to the applications via the APIs. AI/ML may be used to process the data and to train different models to be further used in radio optimization. The applications and the open platforms are vulnerable to diverse types

of attacks. Based on O-RAN ALLIANCE threat modelling, there are various threats identified for Non-RT RIC, Near-RT RIC and the related applications including (O-RAN 2023c):

- penetration to non-RT RIC to cause denial of service
- tracking of user devices
- data corruption or modification
- theft of services, data leakage
- identification or user devices
- change of user device priorities (in a system)
- abusing radio network information
- man-in-the-middle attacks
- degradation of system performance

The list above is not exhaustive and other sources collect other threats that have been identified valid for these functionalities. Nevertheless, these examples alone indicate that an attack to these functionalities could seriously risk the confidentiality, integrity, or availability of the network.

The radio functions O-CU, O-DU and O-RU are the nodes that connect the end user devices to internet and in general they consist of the management-, control-, user- and synchronization-planes. All these planes have their key role and any attack on them cause diverse types of threats. The management-plane supports several functionalities and getting unauthorized access and control over it could allow passive wiretapping or denial of service type threats (O-RAN 2023c, 7.4.1.2). Also performing an unauthorized software update on O-RU over m-plane could result in with the attacker getting full control over the O-RU (BSI 2022, p.58). An attack on synchronization plane would allow serious degradation of the performance or even complete loss of availability as the degradation of the timing signal accuracy would bring down the cells of the O-RUs that are dependent on accurate time (O-RAN 2023c, p.34). Attack on control-plane would also have potentially severe consequences. Gaining control over O-CU would allow user level tracking incl. mobility and use of radio resources while control over O-DU would allow visibility to radio configuration information and for example spoofing of the c-plane messages enabling to block the user-plane traffic (O-RAN 2023c, p. 34). Finally, the control over user-plane would allow the adversary to

wiretap user plane flow or cause denial of service type attacks. To summarize, having the control over different interfaces of the O-RAN radio functions would compromise the confidentiality, integrity, or availability of the network.

In a cloud-native network, the Open RAN functions and the 3rd party applications are run in a cloud host server either on bare metal or on top of hypervisor. Host servers are typically operated in different type of data centers: cell site data center, telco edge data center or regional data center. These data centers have different security classifications, but in general the access to those premises is strictly restricted. There are however several cases of breached data centers and compromise of the hosts. As a fresh example there was a breach in the data center of TietoEvry in Sweden, that caused several businesses to close their online activities across the country. When data center and the host server is compromised, it may allow the attacker to perform nearly anything and impacting all the confidentiality, integrity, and availability. Compromised host may also be hard to detect and it is possible to hide the breach effectively. The 2015 attack to Ukrainian Power Grid is excellent example of compromised host. In that case, the BlackEnergy malware was fed via an email containing an innocent looking excel attachment. Opening the Excel caused the office laptop to be compromised and enabled an access inside the energy company. Since that the malware activated, connected to external command & control server, and based on received controls, scanning first the IT network, then spreading to OT network and scanning the network for the targets and finally installing the malware components to both IT and OT system devices. The malware performed additional cleaning to allow efficient hiding with the low footprint before the final activation to destroy the network after few months. (ISA 2017). This example is merely a case to demonstrate the severity of a compromised host, where the consequences may be disastrous.

Considering the cloud-native Open RAN the traditional management and orchestration (MANO) functionalities are taken care by Kubernetes, a system that is developed and maintained by an open-source project hosted by the Cloud-native Computing Foundation (CNCF). Kubernetes has been also described as Linux

kernel of distributed systems as its abstracts the underlying hardware of the nodes to allow the workloads (containers) to be deployed and to use the shared resources of the host. Kubernetes is a system for automating cloud deployments and manage the containerized applications (CNCF 2024a). As Kubernetes has been designed as a flexible system that can fit large variety of different scenarios and deployments, the flexibility has been considered a weakness from security point of view (OWASP 2024). When searching for the reported vulnerabilities from the NIST national vulnerability database there are several examples that highlight the criticality of the Kubernetes management system (NIST NVD 2024). CVE-2023-5528 reported a case, which allowed for a user that can create pods to be able to escalate to admin privileges on those nodes. CVE-2024-29990 reports a privilege escalation case for Microsoft Azure Kubernetes Service Confidential Containers with a severity score 9.0. With this exploit the attacker could steal credentials and affect the resources (MSRC 2024).

Registry in a cloud management system is a repository that stores and manages the container images. In addition to storing images, the registry usually also supports security related functionalities such as signing the images and scanning them to identify any vulnerabilities. Different policies can be also used to prevent vulnerable images from being deployed in the system. (Harbor 2024). Registry is not free from vulnerabilities and having the images of the deployed containers, it is interesting target for adversaries. As an example, a critical vulnerability CVE-2019-16097 was reported, where a malicious request from the attacker allowed to take over the registry (Unit 42 2019). CVE-2022-46463 reports a high severity issue in access control that allowed the attackers to access Harbor registries without authentication. These examples highlight that getting access to a registry and having admin or system level rights can have severe consequences and critically compromise the security of a system. For this reason, the security of a registry and integrity of the images is necessary for the system.

4.5 Enabling trust with Remote Attestation

As illustrated, cloudified Open RAN systems are complex and have many stakeholders in different areas of technology, deployment, and business ecosystems.

Cloudified and disaggregated radio access networks are the point where telecom and IT worlds meet and the technologies merge more towards from the IT. The addition of artificial intelligence as a new emerging technology increases the complexity further. The previous section presents the functionalities of cloud-native Open RAN system and demonstrates the fragility of the system with all the vulnerabilities that are yet to be discovered in future. The examples highlight that a threat to any part of the system may compromise any of the classic CIA-triad confidentiality, integrity, and availability, and pose a critical risk to business or society. This raises the importance of continuously monitoring the system and having full understanding on the trust of the system and its components.

Cloud-native Open RAN can be modeled as a large and complex network of computers connected to offer a service that the customers can trust being secure and reliable. In such a system the cornerstone of security upon which everything is built is trust. The industry has been working on concepts of confidential computing and trusted computing and defined standards, components, and technologies to enable computing environment that can be trusted. The concept of trust was discussed in section 3, and it was defined as “an assurance that one entity holds that another will perform particular actions according to specific expectation.” It was also concluded that trust is not built on empty ground, but that it is built upon information collected beforehand. This calls for two distinct issues: to trust there needs to be a foundation that can be always trusted, a trust anchor that is also called as Root of Trust (RoT), and to collect information or evidence for trust decision, a reliable process that ensures the integrity of the collected evidence.

Confidential and trusted computing technologies can be used to address security threats related to the compute infrastructure running cloudified Open RAN network functions. The goal of using trusted computing is to ensure that compute infrastructure is in a defined and expected state and that it has not been modified by an attacker. The goal of using confidential computing is to protect workloads from attacks originating from a potentially malicious compute infrastructure. Confidential computing technologies are especially useful in the case where the network operator and the provider of the compute infrastructure (e.g., a public cloud

service provider) are different entities and the network operator does not fully trust the cloud infrastructure that executes the workload.

The integrity of the compute infrastructure can be ensured by taking measurements of system components on each host during system boot. The actual measurements can be taken by a trust agent running on the host utilizing a hardware security function such as Trusted Platform Module (TPM). The measurement results are securely stored on the TPM where their authenticity and correctness can be ensured by utilizing the TPM and protocols defined by the trusted computing group. The results stored can be later queried from a verification service, which compares the measurement values against expected and known good values. The verification server can be optimally co-located with the other central functionalities of the system, for example with the service management and orchestration. In case of cloud infrastructure, the verification service can be combined with the cloud orchestrator such as Kubernetes. The overall knowledge and visibility that Kubernetes has, is useful in scheduling the workloads (e.g., Open RAN network functions) such that based on the policies, those are scheduled only on hosts with validated measurements.

The verification service that is usually centrally located is assigned to remotely assess the state of the computing infrastructure. For this purpose, the Remote Attestation architecture (RATS 2023) as defined by IETF can be used (RFC 9334 2023). RATS architecture specification defines procedures for attesting and verifying the systems, that they are in good state. In their definition, the “Attester” is the peer that provides believable information about itself to the “Relying Party” to consider whether the Attester is trustworthy. The overall process is facilitated by “Verifier”.

When considering the targeted cloud-native Open RAN environment, there are two domains that are logically different and may be independent – Open RAN telco domain and cloud provider domain. This will influence where the verification server functions, in this case Remote Attestation services, are located. Figure 15 presents two independent and new functionalities Remote Attestation SMO (RA

SMO) and Remote Attestation Cloud (RA Cloud) co-located with a system -level functionalities of SMO and Kubernetes, respectively.

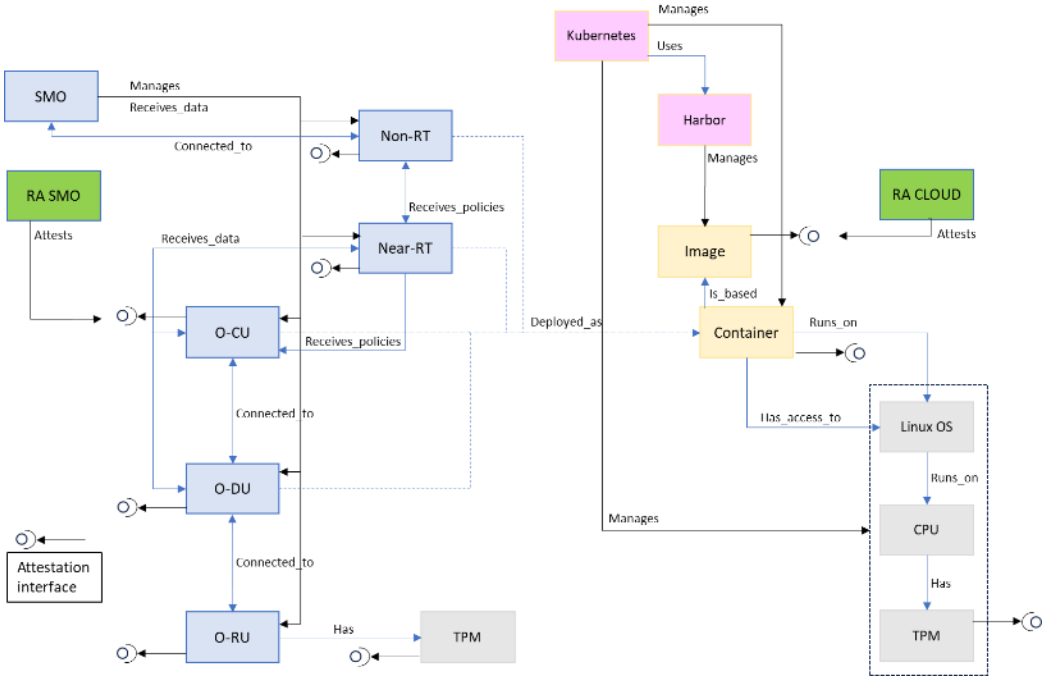

Figure 15. Remote Attestation services in cloud-native Open RAN

The responsibility of RA SMO is to validate the authenticity and integrity of the Open RAN telco domain functions. Remote Attestation service performs attestation of the O-CU, O-DU, and O-RU as well as near-RT RIC and non-RT RIC that all belong to Open RAN system and are managed by an SMO. SMO is a collection of system level functionalities tasked to manage the Open RAN network and it is having excellent system level visibility to the functions of the network and their configurations. The functionalities include management for the configurations, faults, and performance as well as communications surveillance. As SMO is also responsible of the SW management for the network functions, it has the best knowledge on the desired state of the system. Co-locating RA SMO with other SMO functionalities has benefits as it allows the RA SMO to use the services and information from the SMO functionalities in determining that the function is in good state. Also, in case where the received results deviate from those that were expected, it would be easier to take mitigating actions as all the policies of the Open RAN system are managed by the SMO.

The responsibility of RA Cloud is to validate that the underlying computing resources, workloads, and images of the cloud domain are in a known good state. Co-locating the functionality with Kubernetes is useful as it is the engine that orchestrates the cloud resources and workloads and coordinates the registry for the container images. As Kubernetes is the entity that knows where each workload is hosted, it can provide the information e.g., address of the host, that is needed to validate the correct target host.

Open RAN networks can be deployed and operated in diverse ways based on technical and business parameters, and there are also variations in the cloud deployments. This will also influence how RA SMO and RA Cloud are deployed. In case the Open RAN service provider uses its own cloud infrastructure and cloud infrastructure management, the RA SMO and RA Cloud while being separate independent entities, could be easier co-located, coordinated, and integrated as they could be in the same trust domain. In case the Open RAN service provider uses commercial cloud service provider to host its Open RAN functionalities, the RA Cloud function would be owned and controlled by the cloud service provider. In this case, to determine the trust on the network functions, the verification results for the cloud part would have to be queried from the cloud service provider. As Open RAN service provider and cloud service provider belong to different trust domain, a separate trust relationship would have to be established. In this case the overall attestation would become more complex as it would require defining detailed processes, procedures, key performance indicators and service level agreements for the information exchange between the trust domains.

4.6 Root of Trust

Cloud computing environment is a complex system consisting of network of computers put together to offer computing, networking, and storage services for the customers by the cloud service provider. Further each computer is a combination of multiple layers of software and underneath hardware. An application that is running on top of the stack on a computer is using services of the lower layers where each layer is conceptually responsible for different task and communicating with the layers above and below. In layer-to-layer exchange, each layer

should be able to trust that the other layer is processing the data according to expectation or raising an error. In case of cloud computing, the layered structure of can be illustrated with a Linux virtualization stack presented in Figure 16.

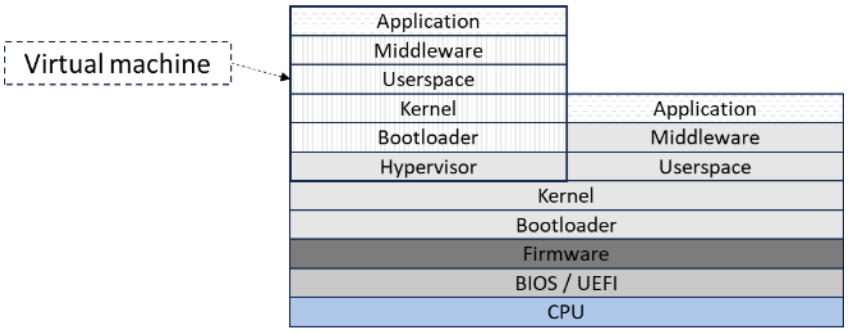

Figure 16. Linux Virtualization stack

When deploying application on the cloud host, the application owner should be able to trust that the underlying system is secure and that none of the layers below is compromised. In the Linux virtualization stack, each layer should be able to trust the next layer underneath, which would mean that there is chain of trust throughout the stack. The chain of trust makes only sense if there is an entity where the trust can be safely anchored or rooted. Considering the case of computing, the hardware at the bottom of the stack is the one where software runs on and the one that ultimately executes all the instructions of the layers above. If there is no trust on hardware, then there can be no trust in the system at all. (Bursell 2022, p. 107).

In a cryptographic system, Root of Trust (RoT) is a source that can always be trusted. Figure 17 illustrates the layers from the application to the Root of Trust.

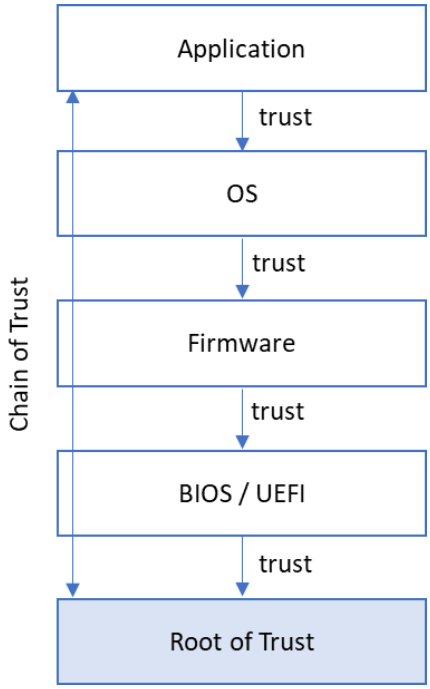

Figure 17. Root of Trust

As depicted in figure, the application trusts OS, which trusts the firmware, which further trusts the UEFI. This effectively forms a Chain of Trust throughout the stack from the application to the Root of Trust residing on hardware. Trusted Computing Group has defined Trusted Computing Base (TCB) that is a collection of system resources consisting of hardware and software that is responsible for maintaining the security policy of a system. The TCB should be able to protect itself from being compromised by external systems. TPM can be used to determine if TCB has been compromised and help preventing the system from starting if the TCB cannot be instantiated. (TCG 2016, p. 21).

In PC architecture, there is a Trusted Building Block (TBB) consisting of hardware and software that establishes a Root of Trust for Measurement and provides connectivity between TPM, Static Root of Trust for Measurement (SRTM), PC motherboard, the platform reset and the TPM physical presence signal. The TBB provides functionality that permits an entity to believe in the measurements that they describe the current state of the platform (TCG 2021, p. 14). Figure 18 below describes the general architectural components of the PC client platform and the

role of TBB. The components within the TBB are SRTM and TPM. There is however a logical 1-to-1 relationship between the SRTM, TPM and PC motherboard (TCG 2021, p. 13-14).

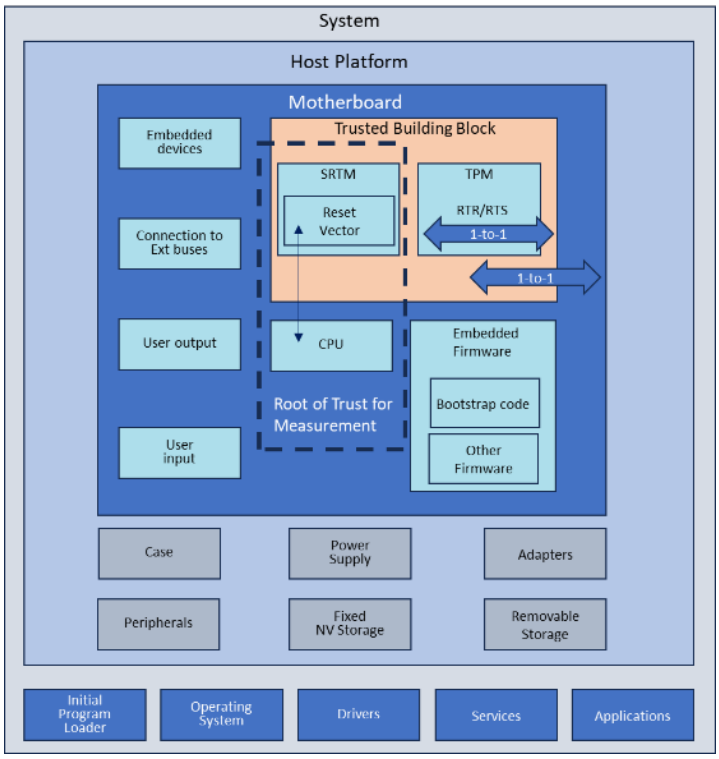

Figure 18. PC client platform architecture and the role of TBB (adapted from TCG 2021)

4.6.1 Trusted Platform Module as Root of Trust

Trusted Platform Module (TPM) is a cryptographic solution defined by Trusted Computing Group (TCG), which allows securely create and store cryptographic keys, secrets, and information. It supports processes that allow to validate, that the underlying operating system and the firmware on a host are as expected and that they have not been tampered. TPM is typically implemented and deployed as a separate and independent component (chip), but TPM 2.0 specification supports the processor manufacturers to embed the TPM capabilities inside their chipsets without the need to have an external component. The TPM standard defines a hardware root of trust (HRoT) which has been widely accepted more secure than software-based solutions.

TPM supports broad set of security related functionalities that can be used in various solutions based on the deployment and scenario. The functionalities offered by TPM are (TCG 2019b):

- symmetric encryption
- random number generation
- cryptographic services
- protected persistent storage for small amounts of data
- counters and extendible registers
- protected pseudo-persistent store for keys and data
- authorization methods to access protected keys and data
- platform identities
- signing and verifying digital signatures
- certifying the properties of keys and data
- attestation, i.e. supporting reporting platform state
- Sealing i.e. authorize access to keys and data based on platform state.

4.6.2 Use of TPM for Open RAN

In cloud-native deployment of Open RAN, the containerized workloads are deployed on cloud hosts that may be owned and managed either by the operator of the Open RAN system or external cloud service provider. The deployment may consist of thousands of workloads distributed in cell site, telco edge and regional data centers as described in section 4.1. To have assurance that the system can be trusted in such a complex scenario requires that each component of a system can be verified and based on that trusted. Due to complexity of the overall deployment, it is evident the process needs to be automated and that there is a functionality located centrally in a system that is allocated responsibility of the process.

Trust decision is based on evidence received from the trustee, in this case from the functions of the Open RAN network or the components they consist of. It is required that there is trust on whole system including all software and hardware components. Remote attestation supported by Trusted Platform Module as presented in Figure 19. can be used to reliably verify and decide on trust of a system and its components.

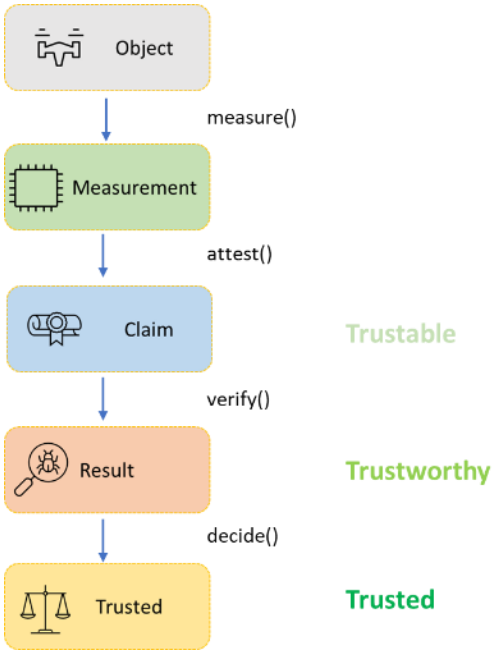

Figure 19. Remote attestation and levels of trust (Adapted from Turcany 2021)

The remote attestation process contains five entities: Object, measurement, claim, result and decision. The subject of the attestation is the object which needs to be measurable (e.g. that we can take a digest of the object or its component). The object is measured where its state is recorded and stored in a secure location. The measurement can be attested by TPM signing the measurement, resulting a claim from TPM. Having the capability to attest the object and receive a response in form of signed claim makes the object trustable. The claim is then passed to a verifier function, which verifies the claim comparing it to expected good values based on attestation policy. The capability to have signed results that can be verified makes the object trustworthy. The result of verification, also called as attestation result, is passed to relying party, which makes the final decision to trust or not trust the object based on its higher-level policies. (Oliver et al. 2023; Turcany 2021, p.32).

For Open RAN system the ability to trust the system requires, that there is capability to measure the components of a system and that evidence (measurement results) can be queried from the system. The integrity of the results must be maintained throughout the process. Trusted Platform Module (TPM) is originally

designed for this purpose, and it offers the capabilities that can be used in validation of the trust in Open RAN. TPM provides three Root of Trusts that fulfill the above requirements:

Root of Trust for Measurement (RTM),
Root of Trust for Storage (RTS), and
Root of Trust for Reporting (RTR).

Root of Trust for Measurement (RTM) and Root of Trust for Reporting (RTR) together comprise a TPM functionality of Platform Integrity Reporting. Per Trusted Computing Group definition “A RoT that makes the initial integrity measurement and adds it to a tamper-resistant log” (TCG 2017, p.5). RTM is critical process as it is the first component in line of the boot process to measure the system. RTM consists of Static Root of Trust for Measurement (SRTM) and Dynamic Root of Trust for Measurement (DRTM). In the SRTM process, each component is measured before it is executed, effectively forming a chain of trust bottom up. The first measurement that is forming the root of trust (RoT) is the Core Root of Trust Measurement (CRTM), which measures the next component in the boot sequence. RTM including CRTM are provided by the BIOS or UEFI provider and it needs to be an immutable portion of the host platform initialization code.

Root of Trust for Reporting (RTR) by a definition is “A RoT that reliably provides authenticity and non-repudiation services for the purposes of attesting to the origin and integrity of platform characteristics (TCG 2017). Its responsibility is to reliably report the measurements stored in the RTS and it uses a cryptographic signing key to sign the measurements, and a digital certificate as evidence to proof that the signing key belongs to a genuine RTR (Proudler et al. 2015, p.62). An RTR report is usually a digitally signed digest of the contents of selected values within a TPM. Typical values of RTR reports are evidence of a platform configuration in PCR such as TPM Quotes or audit logs. The RTR can be identified based on Endorsement Key (EK) that is derived from Endorsement seed of the TPM (TCG 2016).

Root of Trust for Storage (RTS) is responsible of safely storing and protecting the measurements of software performed by RTM or other entities. As the number of measurements is large, the RTS stores only a cumulative digest of all measurements into a Platform Configuration Registers (PCR). (Proudler et al. 2015. p.62)

4.6.3 Keys

TPM key generation produces two types of keys, ordinary keys and primary keys. For the ordinary keys the random number generator is producing the seed for the key computation. The resulted secret key value is stored in a shielded location. The primary keys are derived from seed values with the approved key derivation function (KDF). TPM uses has-based function to generate the keys with two different schemes defined by NIST: SP800-56A for Elliptic-curve Diffie-Hellman keys and SP800-108 for other keys (TCG 2016, p.43).

Every TPM has a unique and non-extractable Endorsement Key (EK) that is generated at a factory with an EK certificate generated by the TPM manufacturer at the time when TPM was produced (Risto 2023, p.18). EK is a key-pair that consists of private and public parts. The private key is stored securely in TPM, and the public part can be shared externally with other parties. TPM vendor may provision the TPM with both TPM vendor and Platform vendor Endorsement Keys together with corresponding certificates. The TPM vendor certificate asserts that the endorsement key is resident on an authentic TPM manufactured by the TPM vendor. The platform manufacturer certificate asserts that the key resident on TPM is part of the platform manufacturer’s platform (Arthur et al 2015).

Attestation Keys derived from Endorsement Key are created by TPM for the purpose of signing the TPM messages such as TPM Quotes, to produce evidence that the messages are genuine and sent by the particular TPM (Proudler et al 2015, p. 101). Attestation key is restricted signing key that may only sign a digest that has been produced by the TPM (TCG 2016).

4.6.4 Platform Configuration Registers (PCR)

Platform Configuration Registers of TPM are registers that are used to store the measurement results taken to monitor the host platform's state during the system boot process. PCR values typically represent the state of the platform, where lower-numbered PCRs are reserved for the process of booting of the system and higher-numbered ones used for events after the kernel has booted. These results together identity keys they can be used to evaluate the health of the platform and its boot sequence. The use of PCRs is architecture dependent, and their detailed use is documented in the architecture specific guidelines. Table 1 presents the PCR usage for x86 based UEFI platform.

Table 1. PCR usage for UEFI platform

PCR index	PCR usage
0	SRTM, BIOS, Host Platform Extensions, Embedded Option ROMs and PI drivers
1	Host Platform Configuration
2	UEFI driver and application code
3	UEFI driver and application Configuration and Data
4	UEFI Boot Manager code (usually MBR) and Boot Attempts
5	Boot Manager Code Configuration and Data, GPT/Partition Table
6	Host Platfrom Manufacturer Specific
7	Secure Boot Policy
8-15	Defined for use by the static OS
16	Debug
23	Application Support

PCR 0 is a key register as it stores information from the early phase of the boot process including the SRTM itself, PEI code, host platform firmware, and the manufacturer embedded drivers. As it contains the measurement results of the code of those components that are provided by the host platform manufacturer, it provides a consistent view of the host platform between the boot cycles. If for any reason a measurement to PCR 0 cannot be made, none of the other SRTM PCR measurements can be trusted and those would be outside the chain of trust (TCG 2021, p. 38).

The use of PCRs allows securing the integrity of the platform as the log of measured events and configuration related data of the underlying host can be stored in these registers in hashed format. The PCR update is based on extension procedure where the new value of the PCR is based on the existing PCR value and the new data. The following exemplary procedure highlights how PCR values are extended.

The current PCR value is *PCR_current_value* which is to be extended with the *new_data*. TPM offers different cryptographic hashing algorithms such as SHA-1, SHA-256, SHA-384. In the extension, the concatenation of the current PCR value and the new data are hashed, as illustrated below:

$$\text{PCR\_new\_value} = \text{Hash}\,(\text{PCR\_current\_value} \,||\, \text{new\_data}) \tag{80}$$

An exemplary PCR extension with SHA-256 could be presented as:

$$\text{PCR\_new\_value} = \text{SHA256}(0\text{x}123456789\text{ABCDEF} \,||\, 0\text{x}0011223344556677) \tag{81}$$

$$\text{PCR\_new\_value} = 0\text{x}3\text{e}5\text{d}18180\text{b}1\text{d}31730\text{eb}7\text{a}2\text{caecdf}\,26\text{c}8\text{ecd}60\text{b}37\text{de}5455\text{f}588651\text{bc}692881\text{be}7 \tag{82}$$

As the new PCR value is based on the existing PCR value and the new value, this integrity of the host can be preserved, as the results are verifiable and can be traced back to previous known configurations.

4.6.5 Platform Boot

Boot is a process to start up the computer from the power on, until the operating system and the applications defined for the startup are running. There are differences in the boot process in different architectures, but on a high level they follow similar steps before the control is handed off to operating system. After power on, a hardware diagnostic is run after the control is handed over to platform initialization (PI) which then starts the BIOS or UEFI initialization. BIOS or UEFI initializes

hardware and embedded devices and loads the bootloader such as GRUB or Windows Boot Manager. Bootloader's primary function is to load the operating system (OS) into memory, and it starts it by loading the OS kernel. Kernel then initializes system services and drivers required by the OS and with the user space initialization, the OS is ready to serve.

A PC using x86 Architecture with UEFI is booted as illustrated in Figure 20.

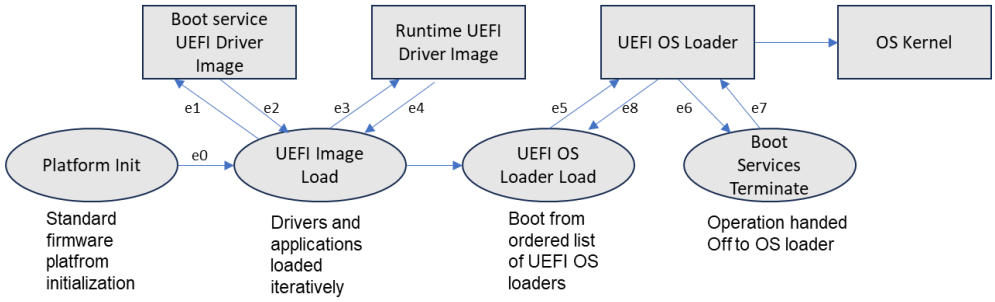

Figure 20. UEFI platform boot process

The above figure presents that sequence of actions for an UEFI platform boot process and the TCG defined events that are measured are marked as e0 to e8. The event 0 is critical as it is the first measurement action in the boot process and made by the platform initialization code (TCG 2021).

Maintaining the integrity of boot process is critical as boot related software components are stored in different area than OS and as those are executed before OS has control. Therefore, it is nearly impossible for OS to detect any malware of the underlying components. This opens an opportunity for rootkits, which are sophisticated and dangerous type of malware. They run in kernel model with the same privilege as operating system and there are diverse types of rootkits that load during different phase of a startup. These are firmware rootkits that overwrote the firmware, bootkits that replace the OS bootloader, kernel rootkits that replace a portion of the OS kernel, and driver rootkits that pretend to be one of the trusted drivers. (Microsoft 2023).

Securing the integrity of the platform is critical and there are countermeasures that can be used against the rootkits. These are Secure Boot, Trusted Boot and Measured Boot, illustrated in Figure 21.

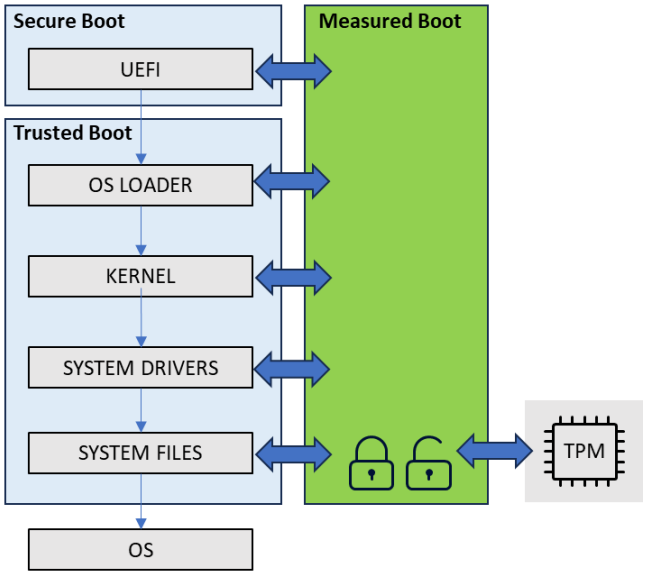

Figure 21. Secure Boot, Trusted Boot and Measured Boot (adapted from Microsoft 2023)

There is confusion within the industry between the secure boot and measured boot and those are often misleadingly used as interchangeable. Secure Boot is a security standard developed by the PC industry to protect the pre-boot phase to ensure that only bootloaders that are trusted by the manufacturer are used in the system boot process. In Secure Boot the digital signature of the bootloader is verified, and the system boot process is only allowed in case of bootloader is trusted. (Microsoft 2023; Kyberturvallisuuskeskus 2020). In case of Windows OS, the trusted OS bootloader would have to be signed by Microsoft or by other vendor in case explicitly approved by the user of the computer.

Trusted Boot is an extension of the Secure Boot and continues the process after OS bootloader has been verified by the firmware. In Trusted Boot the bootloader verifies the digital signature of the kernel before loading it to memory and once kernel oversees the boot process, the kernel verifies the digital signatures of the system drivers and files. If any of the drivers or files are tampered, the bootloader refuses to load the corrupted component. The goal of Trusted Boot is to ensure

that system boot securely and that no unauthorized or tampered code is loaded during the boot process.

Measured Boot goes further from Trusted Boot by measuring the integrity of each component and recording the measuring results into a secure location, a PCR inside a TPM. The measurements together with auditable log allow attesting the system's integrity by a remote validation server at any time. Based on this traceable evidence, a trusted validation server can verify the integrity of the host boot process by comparing received values to expected "good values" and decide whether the host can be trusted.

4.6.6 Remote Attestation

In a complex environment, with hundreds of workloads running or to be instantiated on a host of disaggregated machine of data center, it is essential to know whether the target host can be trusted. Having continuous, up to date knowledge that all the hosts of a system have been verified to be in a good state improves the overall security posture of the system. Conversely, if there are hosts in a system that cannot verified to be in a good state, system policies can be defined for example to allocate reduced privileges or take the host out of service or trigger a process to repair the host. (RFC 9334 2023).

Remote Attestation is procedure that allows for an entity to prove its identity provide measured evidence on its configuration and state, thus making the entity trustworthy. The goal of Remote Attestation is to prove that the host, its configuration, and software running on it, are intact and what they are supposed to be. IETF defines Remote Attestation procedure as where one peer (the "Attester") produces believable information about itself ("Evidence") to enable remote peer (the "Relying Party) to decide whether to consider the Attester a trustworthy peer or not (RFC 9334 2023). Google defines for their cloud services that the attestation is designed to ensure that user data and jobs are only issued to machines that are running their intended boot stack, while still allowing fleet maintenance automation to occur at scale to remediate issues (Google Cloud 2022). One Remote Attestation scenario, where a client requests information whether a device

in a data center can be trusted from an attestation server, is presented in Figure 22.

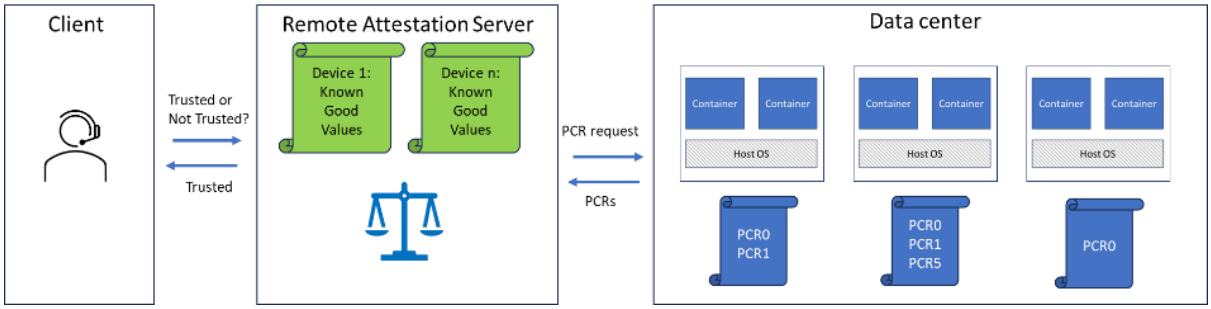

Figure 22. Remote Attestation procedure (adapted from Risto 2023)

In the above scenario, the attestation server requests selected PCRs from the TPMs of the target devices. Values stored in PCRs are results of previous integrity measurements, which are designed to represent the state of a system and its hardware and software configuration. Attestation server compares the received PCR values to known good values and based on those can decide if the target device is trusted or not. The results are then shared with an authorized client, who requested the attestation.

5 BUILDING OPEN RAN EXPERIMENT LABORATORY

The project includes a build-phase with the purpose to evaluate in a laboratory experiment how trust and security can be enforced in cloud-native Open RAN. Cloud-nativeness is widely discussed topic within the IT and Telecom industries, but generally it means applications or services that are designed to run in cloud environment and using cloud computing capabilities such as scalability, elasticity, and resilience. The terms and attributes that are commonly connected to cloud-native are microservices, containers and orchestration.

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications initiated by Google and currently developed by Kubernetes project of Cloud-native Computing Foundation (CNCF) under the Linux Foundation. Kubernetes has been also called Linux kernel of

distributed systems and it has gained de facto status within the industry as container management platform. Figure 23 presents Kubernetes architecture and its main functionalities.

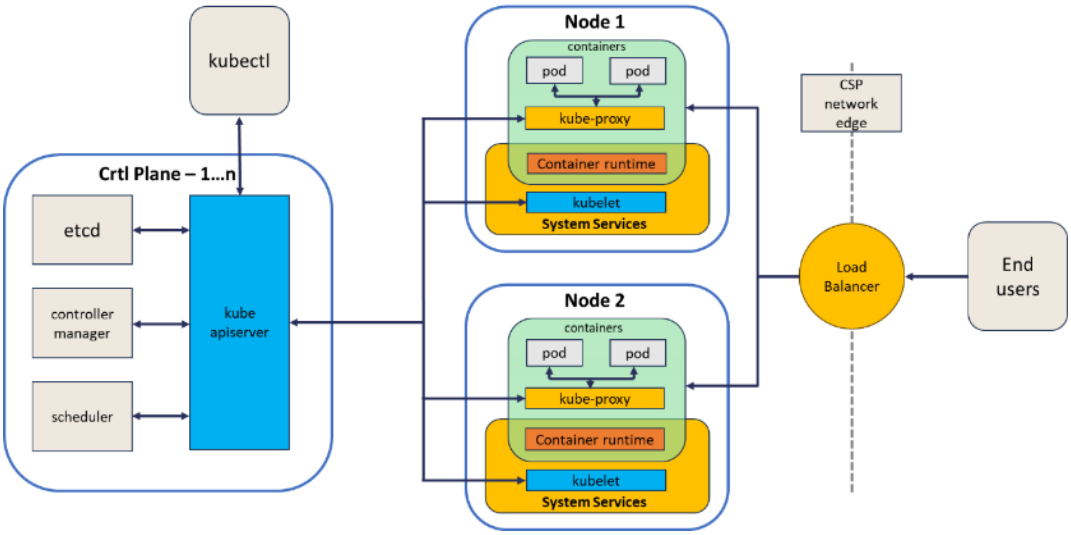

Figure 23. Kubernetes architecture (adapted from Killen 2018)

Kubernetes is deploying containers in units called Pods. A Pod is the smallest work unit in Kubernetes cluster, collection of one or more containers that share the same namespace and volumes. The workloads instantiated as containers are accessed via services that are offered over APIs. The containers, pods, connectivity, and system services are managed by kubectl, which connects over a REST interface to kube apiserver, which authenticates and authorizes all service requests. Kubernetes cluster configuration and secrets are stored in etcd that is the datastore of the cluster. Controller-manager monitors the cluster and initiates control actions to ensure that cluster remains in desired state. Scheduler makes decisions on workload placements by using parameters such as hardware requirements, affinity/anti-affinity rules, labels, system load and other requirements. Kubelet as part of the system service manages the life cycle of a pod and kube-proxy is responsible of the network rules inside a Kubernetes node. Container runtime engines such as Cri-o or Containerd execute and manage containers and act as intermediary between containers and underlying Linux kernel. (Killen 2018).

5.1 Setting up the Open RAN experiment

Project plan included a build phase to model an open RAN network to validate the security solution. As Kubernetes is de facto cloud-native solution for managing containers, this project was also based on Kubernetes.

Open RAN networks are managed by SMO, a collection of centralized functionalities responsible of management of the network functions and cloud infrastructure. In a cloud-native deployment, Kubernetes oversees the cloud infrastructure and workload management while SMO focuses on traditional network management. The radio function 5G gNB is split to O-CU, O-DU and O-RU in O-RAN Architecture. Therefore, in this project the Open RAN network was modelled with following functional entities: Kubernetes (kubectl), SMO, O-CU, O-DU and O-RU. Figure 24 illustrates the experiment laboratory setup that was built to validate the problem and to verify the feasibility of the solution.

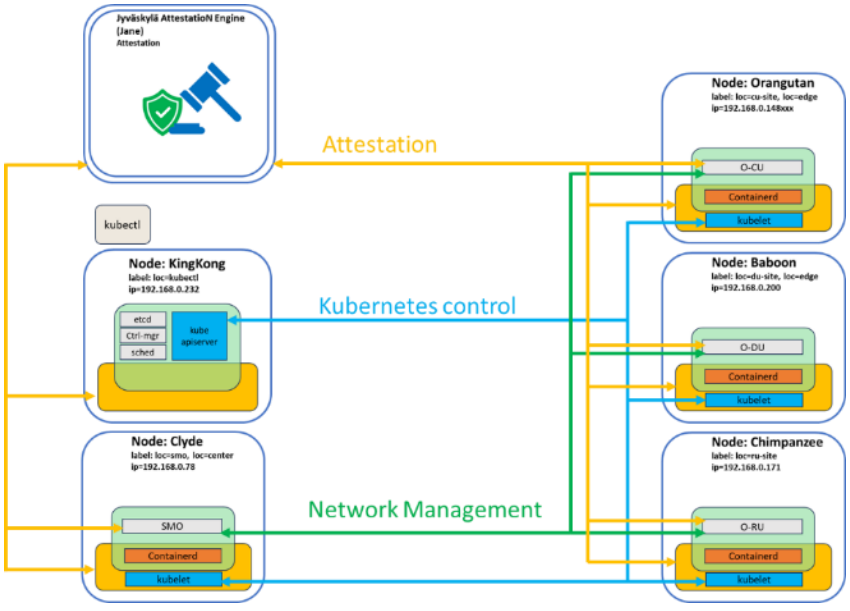

Figure 24. Experiment laboratory setup for cloud-native Open RAN

For Kubernetes cluster, to allow flexibility for experiment and workload placements, five nodes were deployed. Each Kubernetes node was implemented with Raspberry Pi 4 Model B 8 Gb minicomputer running Debian Linus OS. Raspberry Pi's were equipped with TPM2.0 compatible trusted platform module and flashed

with a purpose-build image containing Trust Agent functionality. Trust Agent is an intermediary function that establishes and facilitates the communication between the TPM and Attestation Engine. For remote attestation functionality, Jyväskylä University's JANE attestation Engine was selected. JANE is an open-source project available in Gitlab (JANE 2024) and forked from former A10 Nokia Attestation Engine. JANE implements remote attestation server functionality following the principles documented in IETF RATS specification. JANE attestation server provides mechanisms to attest target devices and provide a judgement on target's trust, based on policies and the known good values expected for the targets.

Kubernetes nodes were setup based on realistic deployment scenario with the exception, that normally the O-RU Radio Unit is based on specialized purpose-built hardware rather than virtualized application instance running on cloud or COTS server. From attestation point of view however, the O-RU as implemented in this laboratory setup (Node: Chimpanzee) allowed attesting the integrity of the server host the same way as it could have been attested in case of real O-RU. For O-CU and O-DU, which implement the higher layer 1 and layer 2 functionalities (O-DU) and Layer 3 functionalities (O-CU), two Kubernetes nodes (Orangutan and Baboon) were allocated. SMO functionality is usually deployed in a central location, typically regional or central cloud center, and in this setup a central node (Clyde) was reserved for that purpose. Kubectl is considered as an independent functionality and could be provided by cloud service provider, hence it was deployed on its own host (kingkong). Figure 25 presents the laboratory setup built to experiment.

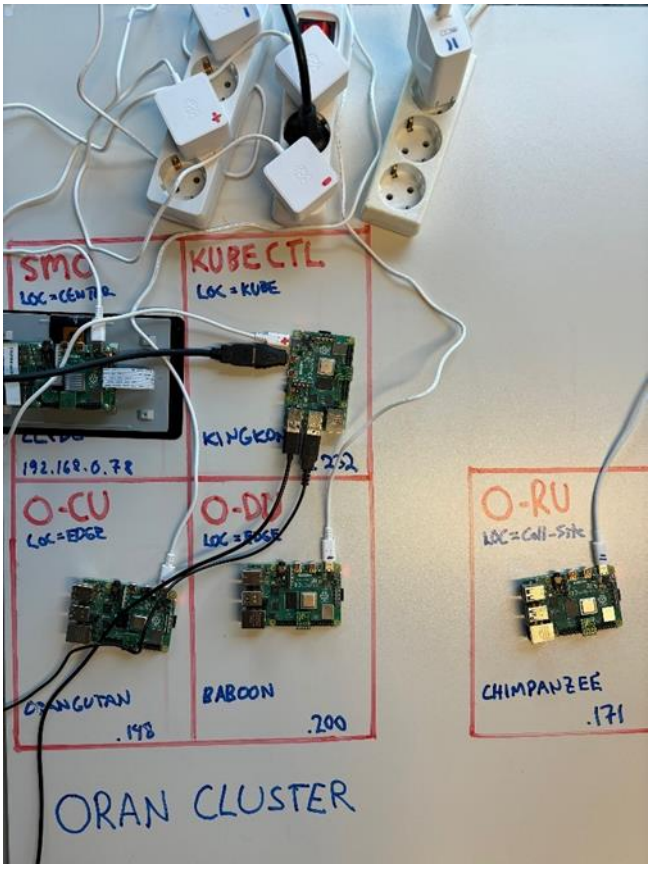

Figure 25. ORAN lab setup

5.2 Setting up Raspberries

Setting up the Kubernetes cluster started with preparing the Raspberry Pi's. Raspberry's OS images located on SD Card were flashed with Raspberry Imaging tool (Figure 26.)

Figure 26. Flashing Raspberry OS to SD cards with Raspberry imaging tool

Flashing process starts by defining the node name and access credentials. In this lab experiment, the planned names for the nodes were named as kingkong, clyde, orangutan, baboon and chimpanzee. Once nodes were named, the next step was to setup the system clock to use external remote network time service, and to re-boot device:

sudo apt install systemd-timesyncd,
sudo systemctl status systemd-timesyncd,
sudo timedatectl set-ntp true,
sudo reboot.

Based on recommendations for Kubernetes Linux installation, memory swap was disabled from each node by editing the dphys-swapfile:

sudo nano /etc/dphys-swapfile
CONF-SWAPSIZE=0
<CTRL+X> and Save

Next step was to define the Linux cgroup-configuration and re-boot. Cgroups is Linux kernel feature that allows to allocate resources use such as CPU, memory, and I/O among a hierarchically ordered groups of processes (Red Hat 2019, section 1.1).

sudo nano /boot/cmdline.txt
+add to end of the line “cgroup_enable=cpuset cgroup_memory=1 cgroup_enable=memory”
<CTRL+X> Save.
sudo reboot

After these configurations, the raspberries were ready to get Kubernetes installed.

5.3 Installing Kubernetes

For Kubernetes, the full k8s solution could have been used, but its footprint is exceptionally large due to several services that were not necessary for Open RAN

lab experiment. Instead, a lightweight Kubernetes version K3S was installed. K3s is a highly available, certified Kubernetes distribution designed for production workloads in unattended, resource-constrained, remote locations or inside IoT appliances. It is packaged as a single <70MB low footprint binary that reduces the dependencies and steps needed to install a production Kubernetes cluster. K3s is optimized for ARM and well suited for Raspberry Pi's. (K3s 2024).

Installing K3s started with installing and configuring the K3s control node. A ready-made script launched from terminal of the targeted host installed the needed components:

```
curl -sfL https://get.k3s.io | sh –
```

This script installed the K3s service with all the needed utilities including kubectl, crictl, ctr, k3s-killall.sh and k3s-uninstall.sh and wrote a Kubernetes configuration (kubeconfig) file to /etc/rancher/k3s/k3s.yaml. This single-node installation on a host "kingkong" would have already been a fully functional Kubernetes cluster, including all the datastore, control-plane, kubelet, and container runtime components necessary to host workload pods, but our deployment was targeted to consist of five nodes.

The next step was to install and configure additional nodes. As control node with kubectl service was already created, the consecutive nodes would be worker nodes. K3s offers a script for installing worker nodes and adding those on existing cluster:

```
curl -sfL https://get.k3s.io | K3S_URL=https://myserver:6443
K3S_TOKEN=mynodetoken sh –
```

The needed token was obtained from the following folder of the K3s server (control node):

```
/var/lib/rancher/k3s/server/node-token
```

Once all the nodes were installed Kubernetes, the result was one control-node and four worker nodes. Kubernetes nodes can be seen on dashboard as presented in Figure 27. Dashboard can be enabled with CLI commands by launching a proxy server that forwards the HTTP requests to Kubernetes API server, which can be accessed from localhost:8001 an admin token:

```
sudo kubectl proxy
sudo k3s kubectl -n Kubernetes-dashboard create token admin-user
```

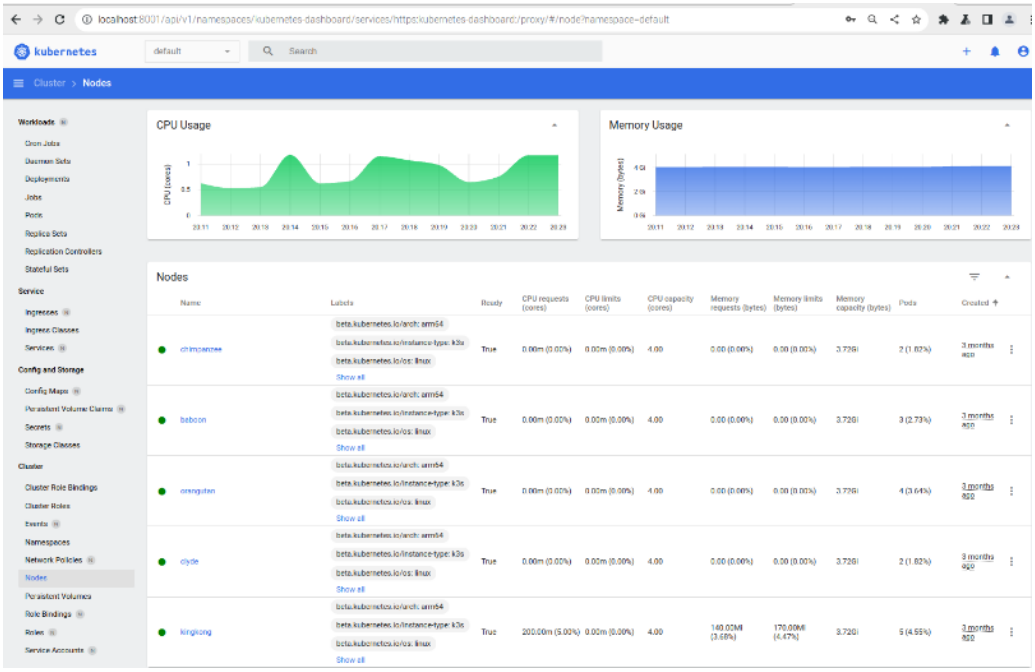

Figure 27. Kubernetes dashboard showing the created nodes.

Kubernetes is based on elasticity, scalability and automation and it is independently aiming to maintain the desired state of the cluster by dynamically scaling up and down resources based on changing conditions in a system. The system is self-healing and balancing the load natively. This means that normally the scheduler of Kubernetes makes all the decisions where to deploy a workload, whether to move the workload from one node to another or whether to allocate more resources. Scheduling can be guided by setting additional constraints e.g. affinity, anti-affinity, compute capabilities etc rules. In this lab experiment, the Kubernetes nodes were assigned with labels, which can be used to guide or “force”

the deployment of a workload to a specific node. Below is a collection of CLI commands that were used for labels:

sudo kubectl label clyde location=smo

sudo kubectl label chimpanzee location=ru-site

sudo kubectl pod orangutan loc=edge

sudo kubectl pod orangutan location=cu-site

sudo kubectl get nodes –show-labels

Figure 28 presents a CLI view on nodes in the Kubernetes cluster and labels associated for each of them.

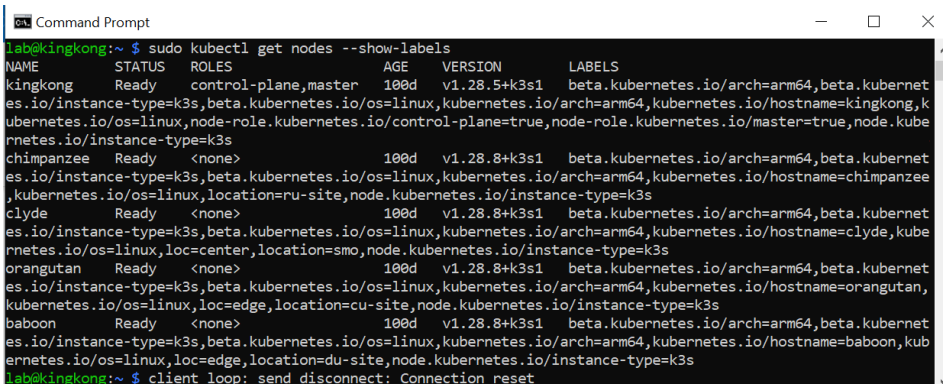

Figure 28. Kubernetes nodes, their status, and labels

Labels associated to nodes can be used in the Kubernetes deployment YAML-file, where attribute NodeSelector: <label_key:label_value> will instruct the Kubernetes scheduler in placement of the workload. One or more labels can be assigned to each node.

5.4 Installing Docker registry

K3s installation is lightweight system and contains only absolutely critical services. If there is need for additional services, those need to be installed separately. Container registry is a service that can be used to manage and share container images. For Kubernetes, typically used registries are Docker and Harbor and this project implemented Docker registry.

Docker was installed from the instructions from Rancher (Rancher 2023):

```
curl https://releases.rancher.com/install-docker/20.10.sh | sh
sudo docker run -d -p 5000:5000 --restart=always –name registry registry:2
```

Installed Docker registry was verified by building “hello-world” image that was successfully built and stored in the registry.

5.5 Building 5G gNB image for Docker

As the scope of this project is to study security of hosts and deployed software packages, it was not required to deploy full 5G base station in each target node. Instead, the experiment modelled the RAN by deploying limited functionalities of 5G gNB and focused only on management plane parts.

In O-RAN architecture the RAN network functions are managed with IETF specified NETCONF/YANG protocol. (O-RAN 2023b). Figure 29 illustrates the NETCONF functionalities located in O-RAN Architecture.

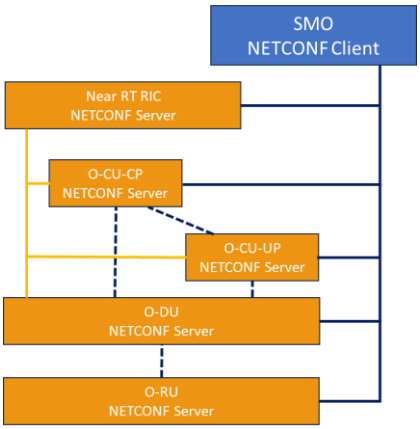

Figure 29. NETCONF client and NETCONF servers in O-RAN Architecture.

NETCONF protocol is designed for NETwork CONFiguration and provides mechanisms to install, configure and delete configurations of network devices. It uses XML (Extensible Markup Language) data encoding for the configuration data and the protocol messages (RFC 6241 2011). NETCONF is independent of the data modeling language and O-RAN uses YANG language for data modeling of the

network configurations. YANG is recommended as it provides advanced features for configuration management (RFC 7950 2016).

Kubernetes nodes in the project were based Raspberry Pi devices with Linux Debian Bullseye distro and processors are based on arm-architecture. This caused challenges in finding a suitable NETCONF client / server application from open-source as majority of available projects are compatible with x64-architecture. There were few candidates supporting arm, but those were relatively old and caused additional problems when building an image. The list of dependencies was extremely long, and no compatible libraries were found, especially for crypto-related functionalities. In the end, after several trial-and-error cycles, a Dockerfile (Appendix 1) based on Ubuntu focal base image (released Apr 2020) required significant number of libraries and dependencies to add, but in the end allowed successfully building an image.

Building of the image was not however the only challenge as the next one was with Docker public and private registries. Netopeer2 image was stored on private registry located on the same host with kubectl. When deploying the pods based on create image, an error response ‘ImagePullError’ was received. Checking the log information from containerd.log indicated that kubectl tried to pull the image from public Docker registry dockerhub.io instead of private one that had been created. Fortunately, Armv7 compatible NETCONF-YANG server/client image was found from Docker hub and taken in use in the experiment lab.

5.6 Instantiating RAN applications

The modelled Open RAN network in the experiment consists of Service Management and Orchestration (SMO) and a gNB disaggregated to O-RU, O-DU and O-CU which all are cloud-natively deployed as containers. Containers are managed by Kubernetes, which would normally make all the decisions on workload placement. As the experiment has cell-site data center (Chimpanzee), edge data centers (Orangutan and Baboon) and Central date centers (Kingkong and Clyde), the setup already suggests that RAN functions should be deployed closer to the (O-

RU) radio site. In the lab experiment, labels were used to guide the RAN functions placement to edge data centers. This was realized by added label ‘location=edge’. Also scheduling of SMO was forced to Kubernetes node clyde with label ‘location=smo’ as the other central cloud node kingkong was reserved for kubectl and good practices do not suggest deploying workloads in the same node. With these constraints, Kubernetes did its workload scheduling as depicted in Figure 30. As it can be seen, SMO was placed on clyde, O-RU management on orangutan and O-CU and O-DU on baboon.

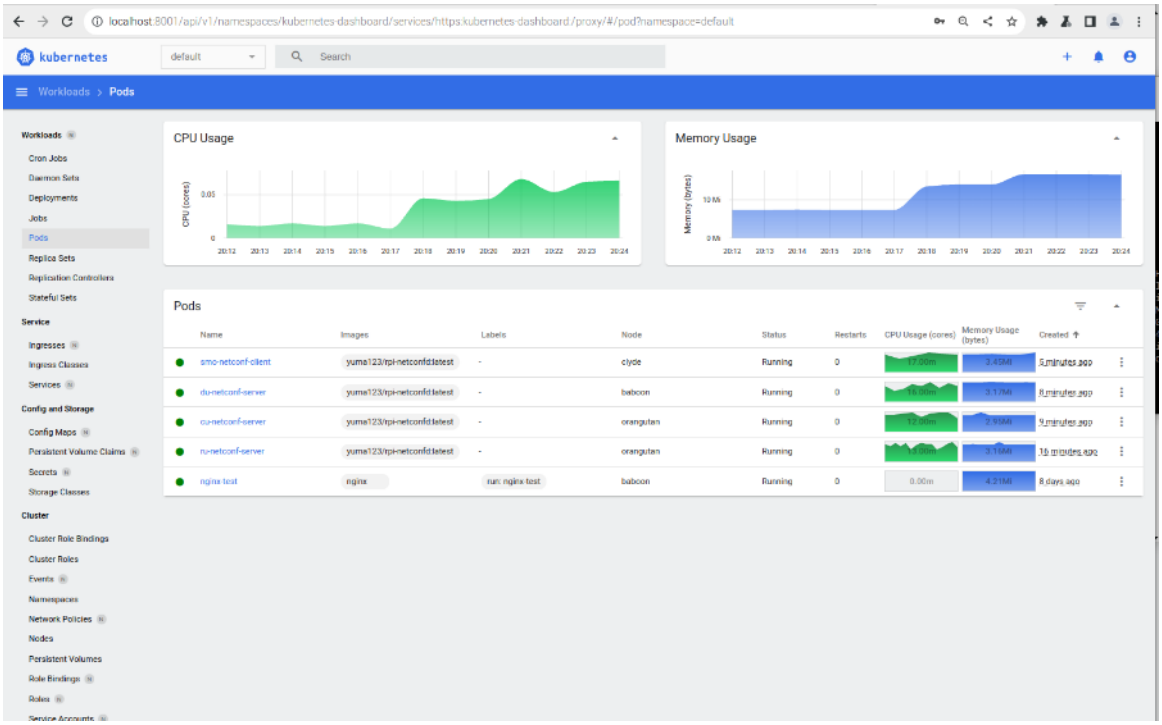

Figure 30. RAN pods deployed at edge data centers as scheduled by Kubernetes.

What should be noted is the location of O-RU management in the edge cloud site. As RAN functions consist of multiple different functionalities split to separate containers and potentially deployed on different hosts, to attest a RAN function means that all the components must be measured and attested. This may mean attesting several hosts, as in case of O-RU in the lab experiment. The other observation is the placement of O-CU and O-RU on the same host (orangutan), while O-DU is deployed on its own host. The learning from Kubernetes workload placement is that scheduler has full control on resources, and it may dynamically

make changes on workload deployments. This will also have an impact to attestation, as only Kubernetes has knowledge on hosts where containers are located.

5.7 Remote Attestation

Main goal of this project is to enable trust and security in Open RAN, and research done in this project suggests using remote attestation with TPMs. Attestation is a process to generate and convey claims about trustworthiness characteristics of an entity. Claims are based on evidence including e.g. device's identity, provenance, software configuration, hardware composition, compliance to test suites and supply chain trust. Remote attestation consists of several actions performed by different entities. IETF remote attestation procedures defines three main roles for attestation: the Attester, who produces believable information about itself to enable a remote peer, the Relying Party to decide whether to consider the Attester trustworthy. Remote attestation procedures are facilitated by Verifier, who evaluates the claims provided by the Attester against reference values (known-good-values) by applying predefined set of policies. The result of the evaluation process is the attestation result that is delivered to Relying Party.

The lab experiment decided to install Jyväskylä University's JANE as a remote attestation server. JANE follows the principles of IETF RATS framework and has similar roles implemented. It maintains the known good values about devices and other elements and provides the attestation and validation mechanisms. JANE provides a Graphical User Interface (Figure 31) allowing to define 'Elements' (e.g. devices) to be assessed, 'Policies' to use in assessment and 'Expected Values' (known good values).

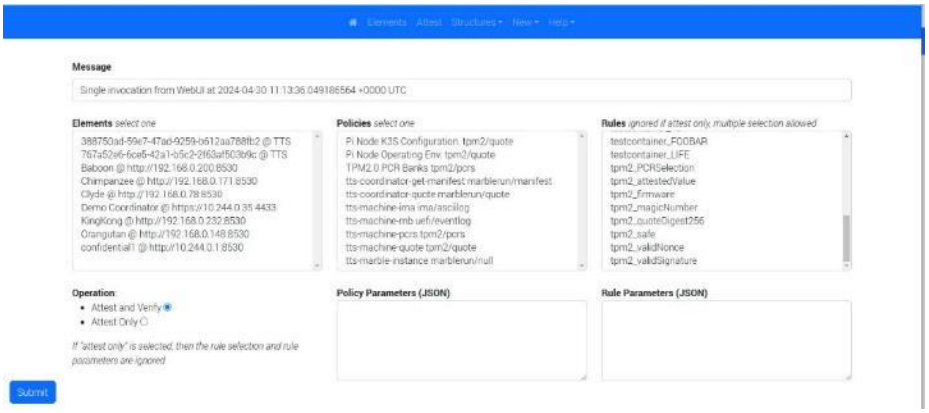

Figure 31. JANE attestation server with elements, policies and rules

As it can be seen from Figure 31 the Open RAN functions are included in the Elements. Three different Policies were defined in this lab setup: Pi Node K3S configuration, Pi Node Operating Environment and TPM PCR banks. Pi Node K3S configuration measures the Kubernetes service system and configuration files, such as k3s, k3s.service and k3s.service.env and stores the results to PCR 4 and 5. Pi Node Operating Environment measures the boot loader and kernel image as well as system configuration related files such as config.txt and cmdline.txt and stores the results to PCRs 0,1 and 7. Figures 32 and 33 demonstrate the Raspberry Pi Node and K3S Policies.

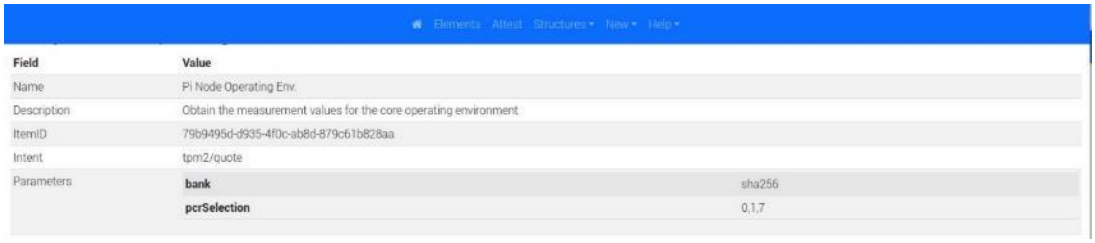

Figure 32. Raspberry Pi Node Operating Environment Policy with PCR 0,1,7

Figure 33. Raspberry Pi K3S Configuration Policy with PCR 0,4,5

Remote attestation server is flexible and allows users to define their own policies based on their devices and environment. This enables to measure any host and its critical system and configuration related files and store results in tamper-proof Platform Configuration Registers of TPM. PCRs can be used for custom purposes, but for certain specific platforms such as UEFI, their use is standardized. The GUI of JANE allows either to attest only, which is useful for reading the stored values, or attest and verify procedure, which compares the evidence to expected values and based on policies provides attestation results. The attestation results can then be used to decide whether to trust an entity. Figure 34 demonstrates attest and verify operation to Kubernetes node clyde where its K3S system and configuration environment has been attested.

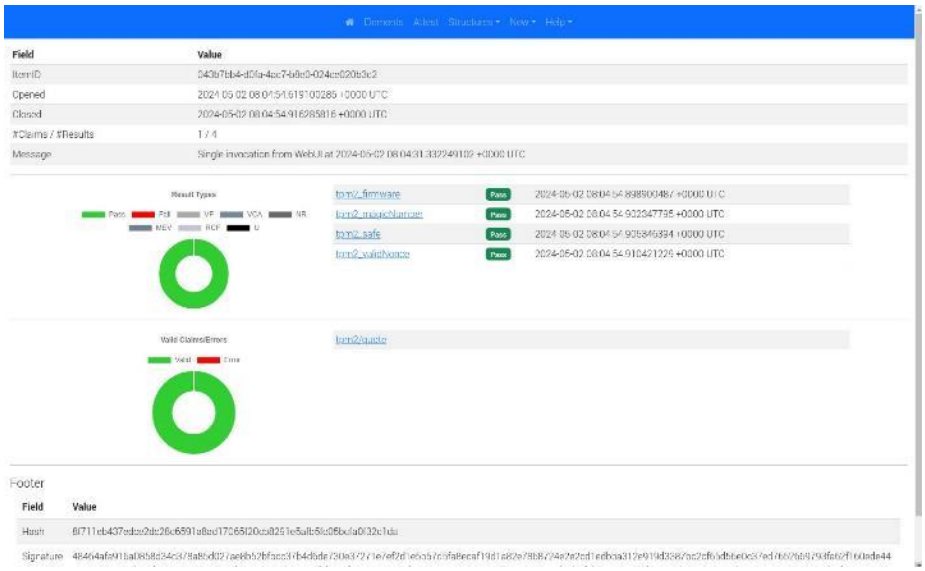

Figure 34. Attestation results of Kubernetes node ‘clyde’ K3S environment

Attestation on K3S environment allows to verify that Kubernetes K3S image as well as the configuration and system files are as expected. In addition to attesting K3S environment, it is critical to also attest and verify the host operating environment. This would include measuring the key boot, system and configuration files and kernel. Figure 35 below shows the attestation results for node clyde’s host operating environment.

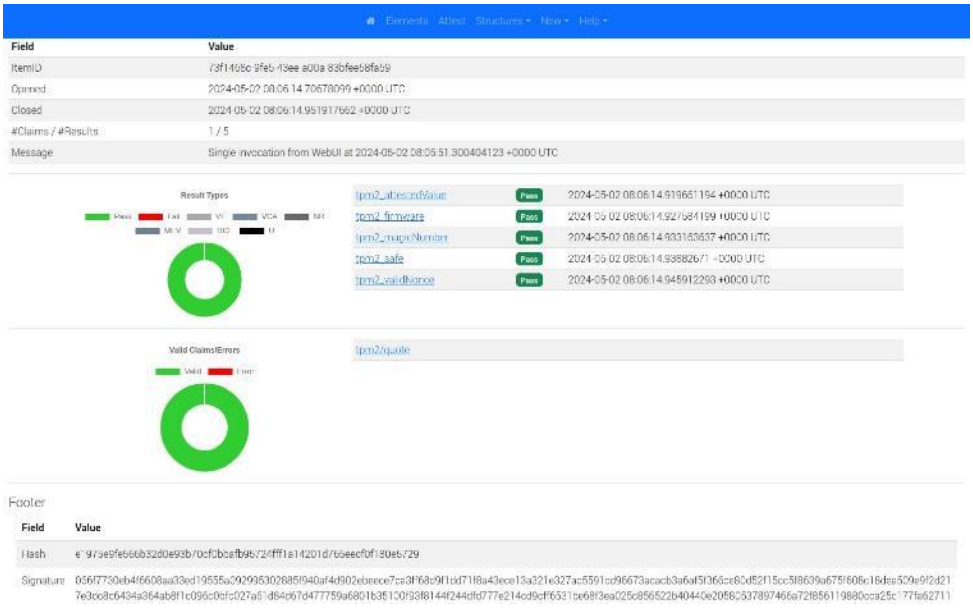

Figure 35. Attestation results of Kubernetes node ‘clyde’ OS environment

Together these attestation results provide reliable cryptographically backed evidence about state of the target host and allows the Relying Party to decide whether to trust the host. A ‘Fail’ result in any of attested rules could trigger predefined policy-based actions such as termination or isolation of the host, forensics analysis, or other appropriate procedures.

As demonstrated above, remote attestation procedures together with use of TPM allows to verify hosts and their key system and configuration related components, and to get assurance about trust on those. As indicated in previous sections, it is not sufficient attest and verify only the hosts, but in addition the remaining parts that are the RAN functions and their components.

Open RAN functions O-RU, O-DU and O-CU are realized with one or more containers instantiated on host(s). To attest and verify a container, it is first necessary to identify the attestable properties of a container and for that it is essential to understand what container is and what it consists of. Container images are result of a build process and for this project also an image for NETCONF Server / Client was built. The inputs for the build process in case of Docker Containers is given in a Dockerfile that can be considered as blueprint of the container (Turcany 2021, p. 25). Dockerfile is a text document that contains all the commands that a user would have to execute on terminal to assemble an image (Dockerfile

2024). Dockerfile starts with a FROM instruction followed by the identity of the base image to be used. In our example (Appendix 1) NETCONF image was built on ubuntu:focal. The next instructions are typically environment variables, installing tools packages and libraries, adding dependencies, and defining permissions and configurations. Each instruction line executed, will add set of changes to previous result, and the changes resulted by instruction are grouped together as layer. This way the building continues in staged, each instruction adding its own layer on top of the previous build stage. (Turcany 2021, p.25). Layers can be observed from the manifest file of an image. Figure 36 provides an example of image manifest, according to OCI specification.


```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.oci.image.manifest.v1+json",
  "config": {
    "mediaType": "application/vnd.oci.image.config.v1+json",
    "digest": "sha256:b5b2b2c507a0944348e0303114d8d93aaaa081732b86451d9bce1f432a537bc7",
    "size": 7023
  },
  "layers": [
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "digest": "sha256:9834876dcfb05cb167a5c24953eba58c4ac89b1adf57f28f2f9d09af107ee8f0",
      "size": 32654
    },
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "digest": "sha256:3c3a4604a545cdc127456d94e421cd355bca5b528f4a9c1905b15da2eb4a4c6b",
      "size": 16724
    },
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad867736",
      "size": 73109
    }
  ],
  "subject": {
    "mediaType": "application/vnd.oci.image.manifest.v1+json",
    "digest": "sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a4333501270",
    "size": 7682
  },
  "annotations":{
    "com.example.key1": "value1",
    "com.example.key2": "value2"
  }
}
```


Figure 36. Example Image Manifest (OCI 2024)

Manifest example above contains image configuration, three layers and image manifest, each associated with sha-256 digest uniquely identifying the property.

In addition to manifest, the image itself can be identified by its sha-256 digest. Image digest is an immutable characteristic generated during the build process and uniquely presents the image. When pulling an image by its digest, exactly same

image is received every time. Figure 37 presents the images and their digests of the lab experiment.

```
root@kingkong:/var/lib/docker/containers# cat /var/lib/docker/image/overlay2/repositories.json
{"Repositories":
        {

        "netopeer-evo2":
        {"netopeer-evo2:latest":"sha256:2c2c721cc3397e47e15277789900011afc5adbb1c64e199f5a6ad8da9e3c50e3"}

        "registry":
        {"registry:2":"sha256:20b02a79df1e04210278fa01c2f0386a50d5118808176a8601acd1149afcfbfc"
        "registry@sha256:d5f2fb0940fe9371b6b026b9b66ad08d8ab7b0d56b6ee8d5c71cb9b45a374307"
        "sha256:20b02a79df1e04210278fa01c2f0386a50d5118808176a8601acd1149afcfbfc"},

        "sysrepo/sysrepo-netopeer2":
        {"sysrepo/sysrepo-netopeer2:latest":"sha256:976d1dcf6c0cc031f87377c287365272b3351a45589be3771f041b2a00df230f"
        "sysrepo/sysrepo-netopeer2@sha256:36cc1d841c97f118d62f775024d60fb1ce210c3bca6b32135c1a9e5a358a1cc9"
        "sha256:976d1dcf6c0cc031f87377c287365272b3351a45589be3771f041b2a00df230f"}

        "ubuntu":
        {"ubuntu:20.04":"sha256:f1b701af35d60a285f85ca02ccbff130ed7b830bd2c31661c3c80c05cd3da89c",
        "ubuntu:cosmic":"sha256:4170043ccad8a83f17f6486517fb7c474ead8bc1b97fb1e3cdd834b86ba3bae3"
        "ubuntu@sha256:7d657275047118bb77b052c4c0ae43e8a289ca2879ebfa78a703c93aa8fd686c"
        "sha256:4170043ccad8a83f17f6486517fb7c474ead8bc1b97fb1e3cdd834b86ba3bae3"
        "ubuntu@sha256:bb1c41682308d7040f74d103022816d41c50d7b0c89e9d706a74b4e548636e54"
        "sha256:f1b701af35d60a285f85ca02ccbff130ed7b830bd2c31661c3c80c05cd3da89c"}
        }
}
root@kingkong:/var/lib/docker/containers#
```

Figure 37. Images and their checksums in private Docker registry

Digests associated with images can be used in attestation as they uniquely identify the image, and therefore ensuring the integrity. For container image attestation, the image and manifest contain several components, each identifiable by their digest. As manifest describes the image and its components in detail and each component associated with sha-256 digest, the manifest can be used in attestation of the container. However, as the supply-chain attacks are in heavy increase, it is important to consider the integrity and security of the build environment as well. Therefore, to secure the integrity of a container, would require including claims both on image manifest and on building environment (Turcany 2021). Figure 38 illustrates procedure for container image attestation.

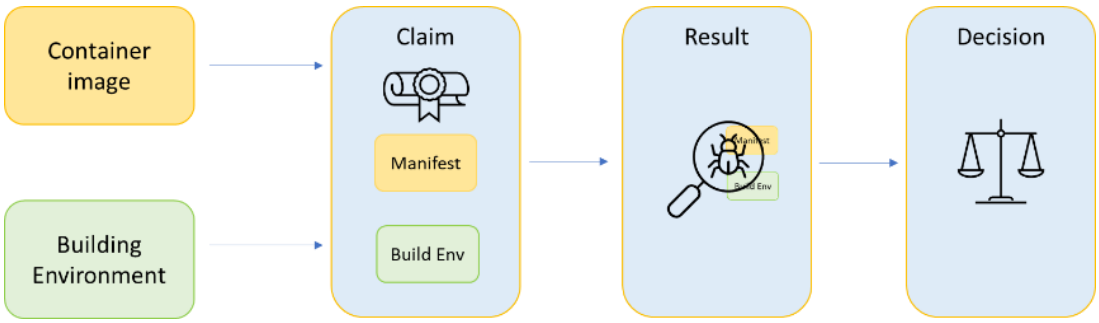

Figure 38. Remote attestation for containers (adapted from Turcany 2021)

Attestation on container instances can be already supported by certain configurations such as containerd inspect. The inspected information should include the following information: creation and starting timestamp, image reference, capabilities, pid, starting command and its arguments, existing environment variable, mounted volumes, resources limits and usage, security context, capabilities, cgroups path (Turcany 2021).

5.8 Remote Attestation and Kubernetes

As discussed, for RAN function to be trusted, it is essential to attest and verify all the hosts and software components related. In case of cloud-native, Kubernetes is de facto solution to manage deploying containerized applications in target cloud infrastructure. Kubernetes is based on elasticity and automation, and it supports e.g. self-healing, load balancing, horizontal scaling, network storage orchestration and automated rollouts and rollbacks. Kubernetes manages also the life cycle of containers, and its scheduler may at any instance relocate the container to another node. Figure 39 illustrates Kubernetes knowledge about the node and its location in the network. Kubernetes also knows the containers hosted by this node, but those are not visible in this figure.

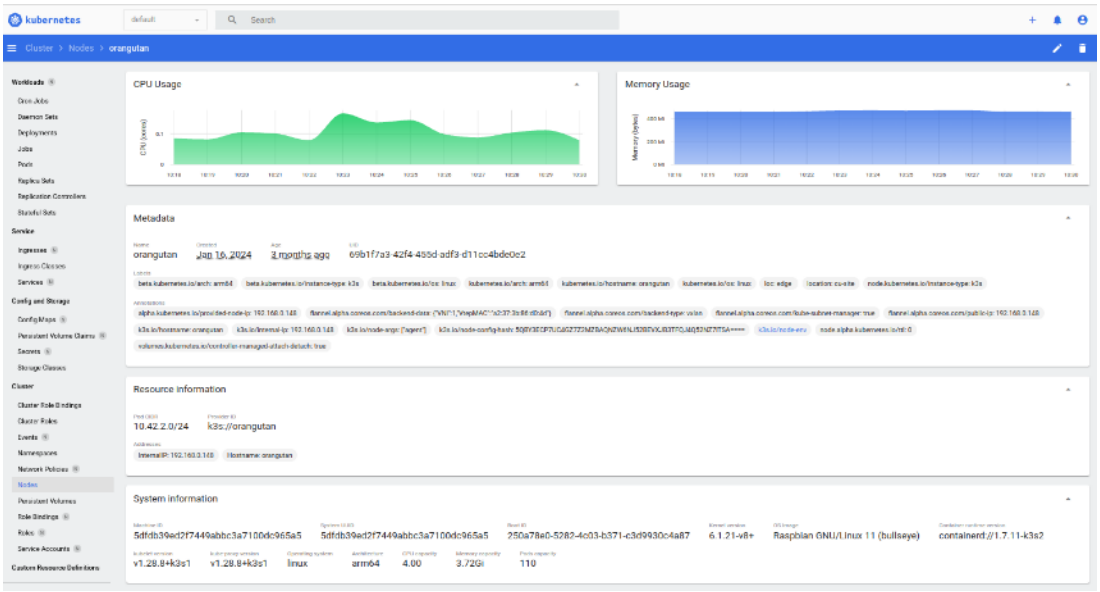

Figure 39. Kubernetes node info from dashboard

As it can be seen from the figure, the node IP address is 192.168.0.148 while pod has been allocated address 10.42.2.0/24. It should be noted that pod address is meant for pod-to-pod communication within a cluster and is not visible outside. As Kubernetes is the only entity that has information about containers, their network locations and hosts they are running, it means that to attest a RAN function either requires tight coordination with Kubernetes or alternatively, attestation should be performed via Kubernetes. As remote attestation in Kubernetes deployment can be realized multiple diverse ways, finding the optimal solution should be researched further.

5.9 Automation by connecting attestation to kubectl

Kubernetes and JANE attestation server both offer APIs that can be used to consume their services. Kubernetes cluster related management activities are performed via kubectl offered by Kubernetes API server while JANE attestation server API allows querying information, attesting devices as well as attesting and verifying. By connecting these services together, automated scripts can be made to verify the status of hosts and their environment and use the results of the verification to manage Kubernetes nodes and their workload scheduling. A short experiment was made with a bash script to demonstrate the feasibility of this automation. Figure 40 presents the script created.

```
#!/bin/sh -x

ELEMENT=3e658489-7d12-4d5f-8f69-1a3a02552ee4
POLICY=c8a3b52f-bc81-4711-a0a4-2396b81e002

EN="$(curl -s -X GET -k -H "Content-Type: application/json" https://192.168.0.217:8520/element/$ELEMENT | jq -r .name)"
PN="$(curl -s -X GET -k -H "Content-Type: application/json" https://192.168.0.217:8520/policy/$POLICY | jq -r .name)"
EN=$(echo "$EN" | tr '[:upper:]' '[:lower:]')

echo Applying $PN to $EN


#open session
SESSION="$(curl -s -X POST -k -H "Content-Type: application/json" https://192.168.0.217:8520/session --data '{"message":"ZZZZZ test from curl"}' | jq -r .itemid)"

#attest element
CLAIMID="$(curl -s -X POST -k -H "Content-Type: application/json" https://192.168.0.217:8520/attest --data '{"cid":"'$ELEMENT'","pid":"'$POLICY'","sid":"'$SESSION'"}' | jq -r .itemid)"

ATTEST1="$(curl -s -X POST -k -H "Content-Type: application/json" https://192.168.0.217:8520/verify --data '{"cid":"'$CLAIMID'","sid":"'$SESSION'","rule":"tpm2_attestedValue","parameters":{}}' | jq .)"
ATTEST2="$(curl -s -X POST -k -H "Content-Type: application/json" https://192.168.0.217:8520/verify --data '{"cid":"'$CLAIMID'","sid":"'$SESSION'","rule":"tpm2_firmware","parameters":{}}' | jq .)"

#close session
curl -X DELETE -k https://192.168.0.217:8520/session/$SESSION

#Print stuff
echo Session was $SESSION with claim $CLAIMID with RESULTS $ATTEST1 $ATTEST2
#curl -s -X GET -k -H "Content-Type: application/json" https://192.168.0.217:8520/claim/$CLAIMID | jq .

if [ $ATTEST1 -eq 0 ] && [ $ATTEST2 -eq 0 ];
then

echo host $EN PASSED
sudo kubectl label nodes $EN trusted=yes
else
echo host $EN FAILED
sudo  kubectl label nodes $EN trusted=no
fi
```

Figure 40. Bash-script to enable automation by connecting attestation to kubectl

The above script is using one of the Kubernetes nodes, identified by its element number (EN), as a target. The selected Policy “K3S operating environment” is identified with its policy number (PN). JANE attestation server requires a session to be established for attesting and that is created first, followed by creating a claim and receiving it. Attestation is performed for two rules, TPM2 attested values and TPM2 firmware, which check that Kubernetes environment is as expected, and that firmware returns expected value. The results attested are verified and attestation result is used to trigger kubectl node labeling action.
If attestation passes, kubectl sets a label with key:value par ‘trusted=yes’ and if it fails, kubectl configures the node untrusted by setting the label ‘trusted=no’. Based on these labels, Kubernetes scheduler may take further actions, for example relocated workloads from untrusted hosts, trigger forensics and reconfiguration actions, and set constraints for future workload placements that label ‘trusted’ must be ‘yes’.

6 CONCLUSIONS

As it has been outlined in this research, Open RAN together with cloudification adds complexity and increases the attack surface of mobile network. The added complexity increases the importance of trust and new approaches and technologies need to be taken in use. This project has discussed the concept of trust and explained how it can be used to enhance security posture of cloud-native Open RAN. Section 4 walks through the Open RAN network and its trust ontologies. A step-by-step approach starting from host server and container pair, ending to full Open RAN system is demonstrated. Key aspect is to measure the state of the assets and computing environment, and based on that decide if a RAN function can be trusted. Further the use of remote attestation together with trusted platform module, and their key characteristics are presented. Section 5 shares the results from experimental laboratory where the concept discussed in sections 3 and 4 are validated. The main conclusions of this project are:

- State of the assets and computing environment must be monitored and verified.

- Remote attestation together with Trusted Platform Modules enables measuring and verifying the targets with the required integrity.
- In Open RAN system, remote attestation is recommended to be placed in central location, e.g. to co-locate with SMO.
- Remote Attestation of Cloud-native based RAN functions requires either tight coordination with Kubernetes or should be done via Kubernetes. This should be studied further.

7 DISSEMINATION OF RESULTS

A survey on industry and key stakeholders indicates that there are various activities on supply chain security, remote attestation, and hardware-based security for trusted computing. These activities can be explained with technological development (cloudification, artificial intelligence etc.), tensions in global political environment and with dramatically changed security landscape. The lack of trust is compensated by creating technologies that will adhere to principles of zero-trust and allow verifying the trust state of a system.

Solutions to secure the future communications networks are required and this project has proven that remote attestation provides solution that can be used to cater integrity and security of Open RAN systems. Global organizations have ongoing activities that offer opportunities to contribute from this research. IETF is working on Remote Attestation Procedures (IETF RATS) and has several activities ongoing. US NIST has done research on trusted container platforms outlining that the most current security control implementations are not coherent and consistent, and that the foundation of layered security approach is the security of physical platform (NIST IR8320 2021). Their activities are assumed to continue, and it is important to monitor their proceedings.

However, as this project focuses on Open RAN system and especially follows the Architectural definitions of O-RAN ALLIANCE, it is essential to review their activities.

O-RAN ALLIANCE

O-RAN ALLIANCE WG11 has several activities related remote attestation and use of hardware-based root of trust. WG11 Security Requirements Specification v.08.00 contains several references to TPMs including the above NIST IR 8320 and ETSI NFV Report on Attestation Technologies and Practices for Secure Deployments. This specification also lists various requirements related to root of trust, remote attestation, or trusted platform modules. As example it lists the following requirements for Chain of Trust for O-Cloud (O-RAN 2023d):

- The O-Cloud platform shall support a root of trust that verifies the integrity of every relevant component in the O-Cloud platform.
- It shall be possible to attest an O-RAN Application through the full attestation chain from the hardware layer through the virtualization layer to the O-RAN Application layer.
- The chain of trust shall be built from measurements stored in a hardware root of trust.
- The chain of trust shall be built from measurements stored in a software root of trust for scenarios where a hardware root of trust is not feasible or available.
- A remote attestation service (AS) should be supported for providing additional benefits beside verifying O-Cloud platform integrity by CoT. The remote AS should collect O-Cloud platform configurations and integrity measurements from data center servers at a O-Cloud service provider via a trust agent service running on the O-Cloud platform servers. O-Cloud service provider is responsible for defining allowlisted trust policies. These policies should include information and expected measurements for desired platform CoT technologies. The collected data is compared and verified against the policies, and a report is generated to record the relevant trust information in the AS database. The remote AS should be extended to include O-RAN Applications integrity.

The requirements are not limited to O-Cloud, but the document also defines, that SW packages need to be protected, including verifying the integrity of application packages. With regards to remote attestation functionality, WG11 has already listed potential commercially available candidate solutions and considered deployment options for example within SMO (O-RAN 2023d, p. 92).

As O-RAN ALLIANCE is the organization that drives Open RAN development and as it already has identified the importance of trust and remote attestation, it offers excellent opportunity to contribute based on findings from this research.

CNCF

Cloud-native Computing Foundation as part of Linux Foundation is the organization that hosts the technical development work of Kubernetes. Kubernetes is de facto cloud-native orchestration solution and researched and experimented also in this project. The findings of this research backed by the experiments reveal that Kubernetes is only entity that has visibility to workloads and their network locations. As it has been identified critical to attest all the components and host related to RAN function, the attestation must be coordinated with or performed via Kubernetes. Adding remote attestation to cloud-native environment, would likely have an impact on Kubernetes, and new functionalities would have to be designed and contributed. A quick review on the github repository of CNCF technical advisory group on security reveals that there are no direct solutions specified, but that the problem has been identified and solutions are needed (CNCF 2024b). In their published whitepaper they have several references indicating the need for ensuring the provenance and integrity of applications during different phases of its life cycle:

> Security concerns within this landscape are complex because of the explicit focus on rapid development and deployment.
>
> This complexity requires a paradigm shift to protect applications by migrating from a purely perimeter-based approach to one where security moves closer to dynamic workloads that are identified based on attributes and metadata (e.g. labels and tags).
>
> Cloud-native computing is highly complex and continually evolving. Without core components to make compute utilization occur, organizations cannot ensure workloads are secure.
>
> In order for security to span all layers of container platforms and services, a hardware root of trust based in a Trusted Platform Module

> (TPM) or virtual TPM can be used. The chain of trust rooted in hardware can be extended to the OS kernel and its components to enable cryptographic verification of trusted boot, system images, container runtimes, and container images, and so on.
>
> End to end attestations may be used to validate the processes used by software creators and suppliers. These attestations should be added to every step in the software supply chain.
>
> A secure CI/CD system should generate SBOMs and attestations. Attestations should include the CI step's process, environment, materials, and products. Evidence should be cryptographically verified when possible. The software producer should use trusted documents such as signed meta-data documents and signed payloads to verify the authenticity and integrity of the built environment.
> (CNCF 2022)

Security is high on the agenda of CNCF, and they have already identified the problem and potential solutions. As there is need for remote attestation and root of trust, CNCF would be ideal organization for developing the results of this project further to realization.

OTHER BUSINESS OPPORTUNITIES

Driven by frameworks such as Zero-Trust, there is increasing demand for continuous monitoring, logging, and having detailed knowledge of the state of the assets and underlying compute systems. Remote attestation provides provenance and integrity protected measurement results from the underlying hardware and its configuration and from software assets running in cloud infrastructure. Further attestation can be extended to cover measurable characteristics from the build environment. In this project, remote attestation was connected to Kubernetes scheduling, demonstrating clear benefits in secure automation. Similarly remote attestation can be connected to other systems including the SIEM or forensics

services. Exploring those are topic for further research, but it is believed that those could provide new business opportunities in the era when the trust state of our systems needs to be continuously monitored.

REFERENCES

3GPP TS 38.401. NG-RAN Architecture description. V18.1.0. Available at: https://www.3gpp.org/ftp/Specs/archive/38_series/38.401/38401-i10.zip

Arthur, W. Challener, D. Goldman, K. 2015. A Practical Guide to TPM 2.0. Apress, Berkeley, CA. Available at: https://doi.org/10.1007/978-1-4302-6584-9_9. [Accessed 20.4.2024]

ATIS Alliance for Telecommunications Industry Solutions. 2023. Enhanced Zero Trust and 5G. White Paper. Available at: https://www.atis.org/resources/enhanced-zero-trust-and-5g/

Barr, W. 2020. Attorney General William P. Barr Delivers the Keynote Address at the Department of Justice's China Initiative Conference. US Department of Commerce. Available at: https://www.justice.gov/opa/speech/attorney-general-william-p-barr-delivers-keynote-address-department-justices-china [Accessed 16.4.2024]

Backlund, A. 2000. The definition of system. Kybernetes, Vol. 29 No. 4, pp. 444-451. Available at: https://doi.org/10.1108/03684920010322055 [Accessed 3.5.2024]

Barber, B. 1983. The Logic and Limits of Trust. New Brunswick, NJ. Rutgers University Press.

Bundesamt für Sicherheit (BSI). 2022. Open RAN Risk Analysis. Study paper. Available at: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/5G/5GRAN-Risk-Analysis.pdf?__blob=publicationFile&v=7 [Accessed 16.4.2024]

Bursell, M. 2022. Trust in computer systems and the cloud. John Wiley & sons.

CNCF. 2022. Cloud-native Security Whitepaper. Github article. Cloud-native Computing Foundation Available at: https://github.com/cncf/tag-security/blob/main/security-whitepaper/v2/cloud-native-security-whitepaper.md [Accessed 3.5.2024]

CNCF. 2024a. Kubernetes open-source project. Web page. Cloud-native Computing Foundation. Available at: https://www.cncf.io/projects/Kubernetes/ [Accessed 17.04.2024]

CNCF. 2024b. Technical Advisory Group Security Publications. Github repository. Cloud-native Computing Foundation. Available at: https://github.com/cncf/tag-security/blob/main/PUBLICATIONS.md [Accessed 3.5.2024]

Connett, B. O'Halloran, B. 2018. Systems Engineering Design: Architecting Trustworthiness in Cyber Physical Systems Using an Extended Aggregated Modality.

U.S. Naval Postgraduate School. Available at: https://www.sciencedirect.com/science/article/pii/S1877050918319598

Cybernews. 2024. Attack on Swedish datacenter shocks multiple businesses. Web article. Available at: https://cybernews.com/news/tietoevry-ransomware-attack-disrupts-businesses/ [Accessed 16.4.2024]

DIGITALEUROPE. 2023. Recommendations for a more ambitious Cyber Defence Policy. Available at: https://www.digitaleurope.org/resources/digitaleuropes-recommendations-for-a-more-ambitious-eu-cyber-defence-policy/ [Accessed 3.5.2024]

Dockerfile. 2024. Docker docs. Webpage. Available at: https://docs.docker.com/reference/dockerfile/ [Accessed 2.5.2024]

EO 14028. 2021. Executive Order on Improving the Nation's Cybersecurity. The President of the USA. Available at: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

Ericsson. 2023. Zero Trust Architecture for advancing mobile network security operations. White Paper. Available at: https://www.ericsson.com/en/reports-and-papers/white-papers/network-resilience-through-zero-trust-architecture

Gambetta, D. 1988. Can we trust trust? In Gambetta, D. (Ed.) Trust: Making and Breaking Cooperative Relations, pp. 213-237. New York: Basil Blackwell

Google Cloud. 2022. Remote Attestation of disaggregated machines. Web page. Available at: https://cloud.google.com/docs/security/remote-attestation [Accessed 23.4.2024]

Harbor. 2024. Open-source cloud-native registry. Available at: https://github.com/goharbor/harbor [Accessed 17.04.2024]

Hardin, R. 2002. Trust and trustworthiness. Russell Sage Foundation.

Hutchins, E. Cloppert, M. Amin, R. 2011. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains Lockheed Martin Corporation. Available at: https://community.mis.temple.edu/mis5208sp2016/files/2015/01/iciw2011.pdf

Infante, D. Rancer, A. Womack, D. 1997. Building Communications Theory. Waveland Press Inc

Irwin, T. 1999. Aristotle - Nicomachean Ethics Second Edition. Hackett Publishing Company Inc.

ISA. 2017. Ukrainian power grids cyberattack. Web article. Information Society of Automation. Available at: https://www.isa.org/intech-home/2017/march-april/features/ukrainian-power-grids-cyberattack [Accessed 16.4.2024]

Jalava, J. 2006. Trust as a Decision, The Problems and Functions of Trust in Luhmannian Systems Theory. Research Report. University of Helsinki, Department of Social Sciences, Department of Social Policy. Available at: http://urn.fi/URN:ISBN:952-10-2855-6

JANE. 2024. Jyväskylä AttestatioN Engine. Jyväskylä University. Available at: https://gitlab.jyu.fi/ijoliver/jane [Accessed 24.04.2024]

K3s. 2024. Lightweight Kubernetes. Project web page. Available at: https://k3s.io/ [Accessed 28.4.2024]

Killen, B. 2018. Introduction to Kubernetes Workshop. Internet2: 2018 Technology Exchange. Orlando, FL. Available at: https://speakerdeck.com/mrbobbytables/introduction-to-Kubernetes-workshop [Accessed 25.4.2024]

Kivinen, S. 2023. Integrating Innovations into a Company's Strategy Through a Concept of Sustainable Futures Business Design, Laurea University of Applied Sciences.

Kyberturvallisuuskeskus. 2020. Secure Boot – haavoittuvuus käyttöjärjestelmissä. Web article. Available at: https://www.kyberturvallisuuskeskus.fi/fi/secure-boot-haavoittuvuus-kayttojarjestelmissa [Accessed 23.4.2024]

Liyanage, M. Braeken, A. Shahabuddin, S. Ranaweera, P. 2023. Open RAN security: Challenges and opportunities. Journal of Network and Computer Applications, Volume 214, May 2023. Available at: https://www.sciencedirect.com/science/article/pii/S1084804523000401 [Accessed 3.5.2024]

Luhmann, N. 1968. Vertrauen, Ein Mechanismus der Reduktion sozialer Komplexität. UTB Uni-Taschenbücher.

Luhmann, N. 1988. 'Familiarity, Confidence, Trust: Problems and Alternatives', in Diego Gambetta (ed.) Trust: Making and Breaking of Cooperative Relations. Oxford: Blackwell, 94-107.

Luhmann, N. 1979. Trust and Power. John Wiley & Sons.

Marsh, S. 1994. Formalising Trust as a Computational Concept. Doctoral thesis. University of Stirling. Available at: https://www.cs.stir.ac.uk/~kjt/techreps/pdf/TR133.pdf

Microsoft Build. 2023. Secure the Windows boot process. Available at: https://learn.microsoft.com/en-us/windows/security/operating-system-security/system-security/secure-the-windows-10-boot-process. [Accessed 22.4.2024]

MITRE Corporation. 2024a. MITRE ATT&CK Cloud Matrix. Available at: https://attack.mitre.org/matrices/enterprise/cloud/

MITRE Corporation. 2024b. MITRE ATT&CK Containers Matrix. Available at: https://attack.mitre.org/matrices/enterprise/containers/

MITRE Corporation. 2024c. MITRE ATT&CK Matrix. Available at: https://attack.mitre.org/

Moilanen, T., Ojasalo, K., & Ritalahti, J. 2022. Methods for development work: New kinds of competencies in business operations. Helsinki: Books on demand.

MSRC. 2024. Microsoft Azure Kubernetes Service Confidential Container Elevation of Privilege Vulnerability. Microsoft Security Response Center. Available at: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-29990 [Accessed 17.04.2024]

NIST IR8320B. 2021. Hardware-Enabled Security: Policy-Based Governance in Trusted Container Platforms. US National Institute of Standards and Technology. NIST Interim Report. Available at: https://doi.org/10.6028/NIST.IR.8320B-draft

NIST NVD. 2024. National Vulnerability Database. Available at: https://nvd.nist.gov/vuln [Accessed 17.04.2024]

NIST SP 800-152 A Profile for U.S Federal Cryptographic Key Management Systems. 2015. US National Institute of Standards and Technology, NIST Special Publication. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-152.pdf

NIST SP 800-160 Volume 2 Revision 1 Developing Cyber-Resilient Systems. 2021. National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v2r1.pdf

NIST SP 800-161 Rev. 1. 2022. Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations. National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161r1.pdf

NIST SP 800-207. 2020. Zero Trust Architecture. US National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

NIST 800-37 Revision 2 Risk Management Framework for Information Systems and Organizations. 2018. National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf

NIST SP 800-95 Guide to Secure Web Services. 2007. National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-95.pdf

Norton. 2023. Danish cloud hosting companies CloudNordic and Azero ransomware attack leads to total data loss. Web article. Available at: https://community.norton.com/en/forums/danish-cloud-hosting-companies-cloudnordic-and-azero-ransomware-attack-leads-total-data-loss [Accessed 14.5.2024]

OCI. 2024. OCI Image Manifest Specification. Available at: https://github.com/opencontainers/image-spec/blob/main/manifest.md [Accessed 2.5.2024]

Ojasalo, K. Koskelo, M. Nousiainen, A. 2015. Foresight and Service Design Boosting Dynamic Capabilities in Service Innovation. Laurea Universities of Applied Sciences. Available at: http://dx.doi.org/10.1007/978-1-4471-6590-3_10 [Accessed 14.5.2024]

Oliver, I., Kuure, P., Sedkowski, W. & Sommer, T. 2023. Ontologising Trustworthy in the Telecommunications Domain. Available at: https://arxiv.org/pdf/2311.15839.pdf [Accessed 16.04.2024]

O-RAN ALLIANCE. 2023a. O-RAN Architecture Description v11.00. 2023. O-RAN ALLIANCE Technical Specification. Available at: https://orandownloadsweb.azurewebsites.net/specifications [Accessed 29.04.2024]

O-RAN ALLIANCE. 2023b. O-RAN Operations and Maintenance Architecture v11.00. Available at: https://orandownloadsweb.azurewebsites.net/specifications [Accessed 29.04.2024]

O-RAN ALLIANCE. 2023c. O-RAN Security Threat Modeling and Risk Assessment 2.0. O-RAN ALLIANCE Technical Specification. Available at: https://orandownloadsweb.azurewebsites.net/specifications

O-RAN ALLIANCE. 2023d. O-RAN Security Requirements Specification v8.00. Available at: https://orandownloadsweb.azurewebsites.net/specifications

OWASP. 2024. Kubernetes Security Cheat Sheet. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Kubernetes_Security_Cheat_Sheet.html [Accessed 17.04.2024]

PortSwigger. 2024. Server-side request forgery. Available at: https://portswigger.net/web-security/ssrf [Accessed 5.3.2024]

Proudler, G. Chen, L. Dalton, C. 2015. Trusted Computing Platforms. Available at: https://doi.org/10.1007/978-3-319-08744-3. [Accessed 21.04.2024]

Rancher. 2023. Installing Docker. Available at: https://rancher-manager.docs.rancher.com/getting-started/installation-and-upgrade/installation-requirements/install-docker [Accessed 28.4.2024]

Red Hat. 2019. Resource management guide – Red Hat Enterprise Linux 7. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/resource_management_guide [Accessed 28.4.2024]

UK Ministry of Defense. 2021. Red Teaming Handbook 3rd Edition. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1027158/20210625-Red_Teaming_Handbook.pdf [Accessed 3.5.2024]

RATS. 2024. Remote Attestation Procedures. Internet Engineering Task Force (IETF). Series of specifications. Available at: https://datatracker.ietf.org/wg/rats/about/

RFC 4949, 2013. Internet Security Glossary of Terms, Volume 2. Internet Engineering Task Force. Available at: https://datatracker.ietf.org/doc/rfc4949/

RFC 6241. 2011. Network Configuration Protocol (NETCONF). Internet Engineering Task Force. Available at: https://datatracker.ietf.org/doc/rfc6241/

RFC 7950. 2016. The YANG 1.1 Data Modeling Language. Internet Engineering Task Force. Available at: https://datatracker.ietf.org/doc/rfc7950/

RFC 9334. Remote Attestation procedureS (RATS) Architecture. 2023. Internet Engineering Task Force. Available at: https://datatracker.ietf.org/doc/rfc9334/ [Accessed 17.04.2024]

RFC 9397, Trusted Execution Environment Provisioning Architecture, Internet Engineering Task Force (IETF). 2023. Available at: https://datatracker.ietf.org/doc/rfc9397/

Risto, R. 2023. Forensics from Trusted Computing and Remote Attestation. Master's thesis. Oulun Yliopisto. Available at: https://urn.fi/URN:NBN:fi:oulu-202306152531

Schneider, F. Bellovin, S. Inouye, A. 1999. Building trustworthy systems: Lessons from the ptn and internet. IEEE Internet Computing Volume 3 Issue 6. pp. 64–72. Available at: https://doi.org/10.1109/4236.807013

Simon, H. A. 1996. The Architecture of Complexity: Hierarchic Systems. The Sciences of the Artificial. pp.183-216. MIT Press.

Stickdorn, M. 2018. This Is service design doing. O'Reilly Media.

Tate, J., Happ, M. (2017). Qualitative Secondary Analysis: A Case Examplar. Journal of Pediatric Health Care. Research Methods Volume 32, Issue 3. 308-312, May 2018. Available at: https://www.jpedhc.org/article/S0891-5245(17)30327-9/fulltext [Accessed 5.3.2024]

TCG. 2016. Trusted Platform Module Library Part 1: Architecture. Trusted Computing Group. Available at: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Rev-2.0-Part-1-Architecture-01.38.pdf [Accessed 22.4.2024]

TCG. 2017. Glossary. Trusted Computing Group. Available at: https://trustedcomputinggroup.org/wp-content/uploads/TCG-Glossary-V1.1-Rev-1.0.pdf [Accessed 21.04.2024]

TCG. 2019a. TPM 2.0 Library Technical Specification. Trusted Computing Group. Available at: https://trustedcomputinggroup.org/resource/tpm-library-specification/ [Accessed 17.04.2024]

TCG. 2019b TPM 2.0 A brief introduction. Trusted Computing Group. Available at: https://trustedcomputinggroup.org/wp-content/uploads/2019_TCG_TPM2_BriefOverview_DR02web.pdf [Accessed 17.04.2024]

TCG. 2021. TCG PC Client Platform Firmware Profile Specification. Trusted Computing Group. Available at: https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClient_PFP_r1p05_v23_pub.pdf [Accessed 22.04.2024]

Thompson, J. D. 1967. Organizations in action: Social science bases of administrative theory. McGraw-Hill.

Turcany, V. 2021. Providing Trusted Computing Services for Multi-access Edge Cloud Computing. Master's Thesis. Aalto University. Available at https://urn.fi/URN:NBN:fi:aalto-202108298549 [Accessed 30.04.2024]

Unit 42. 2019. Critical Vulnerability in Harbor Enables Privilege Escalation from Zero to Admin. Available at: https://unit42.paloaltonetworks.com/critical-vulnerability-in-harbor-enables-privilege-escalation-from-zero-to-admin-cve-2019-16097/ [Accessed 17.04.2024]

Uslaner, M. 2002. The Moral Foundations of Trust. University of Maryland. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=824504 [Accessed 5.3.2024]

Watson, W. 2011. Trust, but verify: Reagan, Gorbachev and the INF Treaty. Western Michigan University. Available at: https://scholarworks.wmich.edu/cgi/viewcontent.cgi?article=1045&context=hilltopreview [Accessed 5.3.2024]

Yin, R. K. 2016. Qualitative research from start to finish. Second edition. New York: The Guilford Press.

APPENDIX 1: DOCKERFILE FOR NETOPEER2

```
FROM ubuntu:focal as build

#In case of "is not valid yet (invalid for another"-repo terror,run this 'sudo hwclock
--hctosys' before building
ENV TZ=Europe/Helsinki
RUN ln -snf /usr/share/zoneinfo/Europe/Helsinki /etc/localtime
RUN DEBIAN_FRONTEND=noninteractive apt-get update -y --allow-downgrades
--allow-remove-essential --allow-change-held-packages
RUN \
    apt-get update && apt-get install -y \
    # general tools
    git \
    vim \
    curl \
    gnupg \
    apt-transport-https \
    python3 \
    python3-pip wget nano bash libz-dev python cmake libmbedtls-dev gcc g++ \
    libssl-dev libev-dev protobuf-c-compiler libprotobuf-c-dev libcmocka-dev \
    swig libavl-dev pkg-config libpcre2-dev libffi-dev rustc libstd-rust-dev

RUN \
    apt-get update && apt-get install -y \
    cmake \
    build-essential \
    supervisor \
    libpcre3-dev \
    pkg-config \
    libavl-dev \
    libev-dev \
    libprotobuf-c-dev \
```

```
    protobuf-c-compiler \
    libssl-dev \
    swig \
    python-dev \
    python3-dev \
    python3-wheel \
    libcurl4-openssl-dev \
    libxslt-dev \
    libxml2-dev \
    libtool \
    libtool-bin \
    python-setuptools \
    libreadline-dev \
    python-libxml2 \
    python-lxml \
    python3-lxml \
    libprotobuf-dev && \
    apt-get install -y libtool libtool-bin libxml2-dev libxslt1-dev libcurl4-openssl-dev
xsltproc python-setuptools cmake zlib1g-dev libssl-dev pkg-config libreadline-dev
&& \
    apt-get install -y bison libboost-thread-dev libboost-thread1.67-dev autoconf
automake screen && \
    apt-get install -y libffi-dev || echo 'livffi-dev does not exist'

# include the latest version of rustup in the image

RUN curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh -s -- -y

ENV PATH="/root/.cargo/bin:${PATH}"

RUN rustup install stable && rustup default stable && rustup component add
rustfmt clippy
RUN \
```

```
    pip3 install ipython ncclient

# add netconf user

RUN \
    adduser --system netconf && \
    echo "netconf:netconf" | chpasswd

# generate ssh keys for netconf user
RUN \
    mkdir -p /home/netconf/.ssh && \
    ssh-keygen -A && \
    ssh-keygen -t dsa -P '' -f /home/netconf/.ssh/id_dsa && \
    cat /home/netconf/.ssh/id_dsa.pub > /home/netconf/.ssh/authorized_keys

# use /opt/dev as working directory
RUN mkdir /opt/dev
WORKDIR /opt/dev

# pyang
#RUN \
#      cd /opt/dev && \
#      git clone https://github.com/mbj4668/pyang.git && \
#      cd pyang && \
#      git checkout cca321ef0c6ddf82c77c12aca8301bcfdfd5b7d3 && \
#      python setup.py install

# libssh
RUN \
      cd /opt/dev && \
      git clone https://git.libssh.org/projects/libssh.git libssh && \
      cd libssh && \
      # git checkout afa4021ded6e58da4ee4d01dbf4e503d3711d002 && \
```

```
    git checkout libssh-0.8.7 && \
    mkdir build && cd build && \
    cmake .. && \
    make -j6 && \
    make install && \
    ldconfig


# libyang
RUN \
    cd /opt/dev && \
    git clone https://github.com/CESNET/libyang.git && \
    cd libyang && \
    git checkout 85d09f3bdf5ea01ea2e01deb384b2b0dde057e3f && \
    git checkout v0.16-r3 && \
    mkdir build && cd build && \
    cmake .. && \
    make -j6  && \
    make install && \
    ldconfig

# protobuf
#RUN \
#    cd /opt/dev && \
#    git clone https://github.com/protocolbuffers/protobuf.git && \
#    cd protobuf && \
#    #git checkout ff3891dab1b1f462d90a68666d14f57eb5fea34f && \
#    git submodule update --init --recursive && \
#    sh autogen.sh && \
#    ./configure && \
#    make -j6 && \
#    make install && \
#    ldconfig
```

```
# protobuf-c
#RUN \
#     cd /opt/dev && \
#     git clone https://github.com/protobuf-c/protobuf-c.git && \
#     cd protobuf-c && \
#     git checkout dac1a65feac4ad72f612aab99f487056fbcf5c1a && \
#     sh autogen.sh && \
#     ./configure --disable-protoc && \
#     # Think in the end we did
#     #./configure && \
#     make -j6 && \
#     make install && \
#     ldconfig


# sysrepo
RUN \
    cd /opt/dev && \
    git clone https://github.com/sysrepo/sysrepo.git && \
    cd sysrepo && \
    # git checkout 724a62fa830df7fcb2736b1ec41b320abe5064d2 && \
    git checkout v0.7.7 && \
    mkdir build_python3 && \
    cd build_python3 && \
    cmake -DREPOSITORY_LOC=/sysrepo -DGEN_PYTHON_VERSION=3 ..
&& \
    make -j6 && \
    make install && \
    ldconfig

# libnetconf2
RUN \
```

```
    git clone https://github.com/CESNET/libnetconf2.git && \
    cd libnetconf2 && mkdir build && cd build && \
    # git checkout 54ba1c7a1dbd85f3e700c1629ced8e4b52bac4ec && \
    git checkout v0.12-r1 && \
    cmake .. && \
    make -j6 && \
    make install && \
    ldconfig

# keystore
RUN \
    cd /opt/dev && \
    git clone https://github.com/CESNET/Netopeer2.git && \
    cd Netopeer2 && \
    # git checkout d3ae5423847cbfc67c844ad19288744701bd47a4 && \
    git checkout v0.7-r1 && \
    cd keystored && mkdir build && cd build && \
    cmake .. && \
    make -j6  && \
    make install && \
    ldconfig

# netopeer2
RUN \
    cd /opt/dev && \
    cd Netopeer2/server && mkdir build && cd build && \
    #git checkout d3ae5423847cbfc67c844ad19288744701bd47a4 && \
    git checkout v0.7-r1 && \
    cmake .. && \
    make -j6 && \
    make install && \
    cd ../../cli && mkdir build && cd build && \
    cmake -DCMAKE_BUILD_TYPE:String="Debug" .. && \
```

```
    make  && \
    make install


RUN \
  apt-get clean && \
  apt-get autoclean && \
  rm -fr /opt/dev
```