Technology
Arts Sciences
TH Köln

5G-FORAN
DFIR IN OPEN RAN

Cologne University of Applied Sciences
Faculty of Information, Media and Electrical Engineering

Research Project

Development of an Adversary Simulation Strategy for a Kubernetes-based Open RAN Deployment

Master of Computer Science and Engineering (M.Sc.)

Author: Jonas Dieterich
Student number: 11135427
E-Mail-Address: arn_jonas.dieterich@th-koeln.de
Examiner: Prof. Dr.-Ing. Andreas Grebe

Cologne, June 18, 2024

Abstract

The research project “IT-Forensics in Open RAN (FORAN)” at the Cologne University of Applied Sciences is a joint effort with the PROCYDE GmbH with the goal of addressing security challenges facing Open Radio Access Network (RAN), specifically in the context of enhancing forensic and incident response capabilities. The project is split into two sub-projects, namely FORAN-ATTACK and FORAN-DFIR. The primary objective of the overall project is to develop a method for analyzing, treating and resolving security incidents in the context of Open RAN, these forensic topics are addressed by FORAN-DFIR. Fulfilling these objectives requires the creation of plausible attack traces, which are developed in the sub-project FORAN-ATTACK. This research project is part FORAN-ATTACK and presents a structured approach to developing an adversary simulation strategy based on a Kubernetes-based Open RAN deployment.

The outcomes of the research project include a comprehensive threat assessment for an Open RAN implementation developed by the O-RAN Software Community (O-RAN SC). Furthermore, the development of a threat model and methodology associated with extending and refining the model are presented in this report. At last, the definition of concrete attack scenarios and means of executing the scenarios based on the recommend tools is outlined. To conclude, the report discusses the findings and results of the research project, reflecting on the limitations and challenges faced, as well as presenting future work, that improves upon the outcomes of this research project.

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
ARP	Address Resolution Protocol
AWS	Amazon Web Services
Azure	Microsoft Azure
BSI	Federal Office for Information Security
CAPEC	Common Attack Pattern Enumeration and Classification
CI/CD	Continous Integrationa and Continous Deployment
CIA	Confidentiality, Integrity and Availability
CIS	Center for Internet Security
CKC	Cyber Kill Chain
CNCF	Cloud Native Computing Foundation
CNI	Container Network Interface
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CSA	Cloud Security Alliance
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DAST	Dynamic Application Security Testing
DevSecOps	Development, Security and Operations
DFD	Data Flow Diagram
DFIR	Digital forensics and incident response
DREAD	Damage, Reproducibility, Exploitability, Affected Users, Discoverability
FCAPS	Fault, Configuration, Accounting, Performance, Security
FiGHT	5G Hierarchy of Threats
FORAN	IT-Forensics in Open RAN
GCP	Google Cloud Platform

gNB	New Radio (NR) Next Generation Node Base
HTTP	Hypertext Transfer Protocol
IoC	Indicator of Compromise
KPI	Key Performance Indicator
MITRE	MITRE Corporation
MITRE ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
MNO	Mobile Network Operator
near-RT-RIC	Near-real-time RAN Intelligent Controller
NF	Network Function
non-RT-RIC	Non-real-time RAN Intelligent Controller
NR	New Radio
NVD	National Vulnerability Database
O-CU	Central Unit
O-DU	Distributed Unit
O-RAN	O-RAN ALLIANCE
O-RAN SC	O-RAN Software Community
O-RU	Radio Unit
OWASP	Open Web Application Security Project
PASTA	Process for Attack Simulation and Threat Analysis
PoC	Proof of Concept
RAM	Random Access Memory
RAN	Radio Access Network
RCE	Remote Code Execution
RIC	RAN Intelligent Controller
SaaS	Software as a Service
SAST	Static Application Security Testing
SDLC	Software Development Life Cycle
SMO	Service Management and Orchestration
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TIP	Telecom Infra Project
TTPs	Tactics, techniques and procedures
UE	User Equipment
VM	Virtual Machine
VMM	Virtual Machine Monitor
WG11	O-RAN Alliance Security Working Group 11

YAML	YAML Ain't Markup Language

1 Introduction

The introduction of Open RAN widens the attack surface of mobile networks and thereby requires the development and improvement of further defensive capabilities. The means by which Open RAN increases the attack surface and the novel techniques and strategies an adversary could pursue to achieve malicious objectives, are explored in this research project. The basis of this analyses is an Open RAN implementation, specifically built and deployed using the container orchestration platform Kubernetes.

1.1 Background

The research project "FORAN" [1] at the Cologne University of Applied Sciences [2] was launched to aid in addressing these security challenges. The project is split into two sub-projects, namely FORAN-ATTACK and FORAN-DFIR. The primary objective of the overall project is to develop a method for analyzing, treating and resolving security incidents in the context of Open RAN, these forensic topics are addressed by FORAN-DFIR.
However, achieving the objectives mentioned above requires the creation of plausible attack traces, which are developed within the scope of this research project and part of the sub-project FORAN-ATTACK. These traces are derived from an in depth technology review and analysis of the components of a verified and functional Open RAN deployment.
The technology review, discovery of vulnerable components and threats to the overall system form the core of a threat model constructed in this project. The threat model is utilized for deriving attack scenarios and generating attack traces for the purpose of forensic analysis and review.

1.2 Motivation

The criticality of cellular networks and relevance to national security cannot be overstated. As Open RAN enhances cellular networks, it forms an integral part of them and security considerations must be taken seriously.

The increased attack surface requires additional attention as new attack vectors demand the development of tailored detection, analysis and remediation capabilities. Anticipating the artifacts, tooling and tactics, techniques and procedures (TTPs) employed by future attackers on this infrastructure forms the foundation for developing defensive and forensic competencies, such as which components require increased visibility and what traces are likely to be created by an attacker taking the path of least resistance. Therefore, this research paper analyzes a functional Open RAN deployment and its underlying technologies, identifying and interconnecting individual threats and vulnerabilities into a comprehensive threat model. Additionally, it derives potential attack scenarios from the threat model aimed at mimicking adversarial techniques applied to Open RAN infrastructure.
Thereby, this work contributes to increasing the resilience of cellular infrastructure, specifically Open RAN deployments, by aiding in the development and verification of forensic capabilities by devising potential TTPs of adversaries targeting Open RAN infrastructure.

1.3 Structure of the Report

The last section of the introduction provides an overview of the upcoming content. The first chapter introduces the topic and context of this project. Moreover, it provides a brief background on the related technologies and initiatives of this research. The second chapter defines the objectives and scope of the research project. The third chapter introduces the theoretical framework and methodologies used in this research. The fourth chapter outlines the principles of threat modeling and how they are applied in this research project. The fifth chapter describes the threat assessment conducted for an Open RAN deployment. The sixth chapter outlines the implementation of a threat model in the form of attack trees. The seventh chapter discusses the attack scenarios and and recommended tooling to conduct adversary simulation testing. The eighth chapter reflects on the findings and results of the research project. The ninth chapter concludes this report.

2 Planning and Definition

This chapter outlines the problem statement this research papers strives to address. Furthermore, the limitation and boundaries of the research are laid out to define the scope of this investigation. At last, specific objectives are presented that are to be achieved by this project.

2.1 Problem Statement

The adoption of Open RAN leads to an increased attack surface that stems from the disaggregation and cloudification of the RAN, with additional open interfaces and multi-vendor interoperability necessitating the development of defensive measures tailored to the new infrastructure landscape. These defensive measures not only entail preventative measures; increased visibility, detection capabilities and response, including retrospective analysis of incidents, are all of high significance in regards to assuring unobstructed operation of the RAN and protecting the integrity and confidentiality of sensitive data. Therefore, analyzing the novel threat landscape and developing a threat model to derive potential attack scenarios, aids in tailoring the defensive and forensic measures to the unique requirements stemming from Open RAN deployments orchestrated by Kubernetes. Moreover, these attack scenarios are designed for automating security testing and verification of forensic and incident response abilities evolved during the research project FORAN.

2.2 Scope

To ensure a focused analysis and research of the problem statement, this section defines the specific scenarios and circumstances that this project will investigate, while also setting clear limitations and boundaries of what is beyond scope.
Threat modelling is a continuous process, especially in regards to software components that are currently in development. Furthermore, anticipating adversaries TTPs without publicly available real-world threat intelligence, significantly increases the complexity. Thereby, the outcome of this project is subject to change as the adoption

of Open RAN rises and more empirical data gathered from operational deployments in the context of production mobile networks operated by Mobile Network Operators (MNOs).
This investigation only assesses one specific release of an Open RAN implementation, namely the I-Release of the O-RAN SC released on the 4th of December 2023 [3]. Moreover, only bare metal Kubernetes deployments are considered. Therefore, deployments in public clouds such as Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure (Azure) are not considered, as well as deployments using container orchestration platforms other than Kubernetes are out of scope. These decision stem from the recommended and supplied deployment instruction from the developers of the Open RAN implementation, the O-RAN SC, themselves [4] [5].

2.3 Objective

The objectives of this research entail a comprehensive technical review of an Open RAN deployment and the relevant technologies that this deployment is based upon. Further, based on this review a meticulous analysis of threats and vulnerabilities is conducted and used to establish a threat model of an Open RAN deployment based on Kubernetes. Subsequently, the threat model is used to derive potential attack scenarios, that form the basis for attack trace generation and adversary simulation. At last, an execution strategy is presented, which includes recommendations for necessary tools and the selection of attack scenarios used for generating attack traces.

3 Theoretical Framework

This chapter discusses the relevant technologies and theoretical concepts that are fundamental to investigating and assessing the threat landscape of a Kubernetes orchestrated Open RAN deployment. Furthermore, the security frameworks and terminology relevant to adversary simulation and digital forensics and incident response (DFIR) are explored.

3.1 Open RAN

The traditional approach of a proprietary RAN based on closed-source software and a single-vendor implementation using custom hardware, has been challenged with the introduction of Open RAN. The goal is to achieve more flexibility and programmability in the RAN architecture. through the disaggregation, virtualization and cloudification of the RAN. The decoupling of hardware and software and improved interoperability, enables a thriving multi-vendor ecosystem based on Commercial Off-The-Shelf (COTS) hardware.
This approach leads to a new architecture 3.1 with additional open interfaces and atomic Network Functions (NFs). These NFsnetwork functions can be controlled and scaled individually using container orchestrators in a microservice architecture. Furthermore, the integration of open-source software and adoption of Continous Integrationa and Continous Deployment (CI/CD) enables a rapid onboarding of new software-defined functionality, in contrast to the slow and elaborate update process of proprietary components implemented in custom hardware. The new architecture encompasses additional NFs to enhance the RAN's intelligence. These NFs are implemented in the form of controllers, that perform operations using control loops with varying timing intervals, depending on the real-time requirements of the functions executed. The intelligence gained by the RICs originates from the extension of the RICs by xApps and rApps. This data-driven control of RAN can further be complemented using machine learning and artificial in conjunction with these apps.

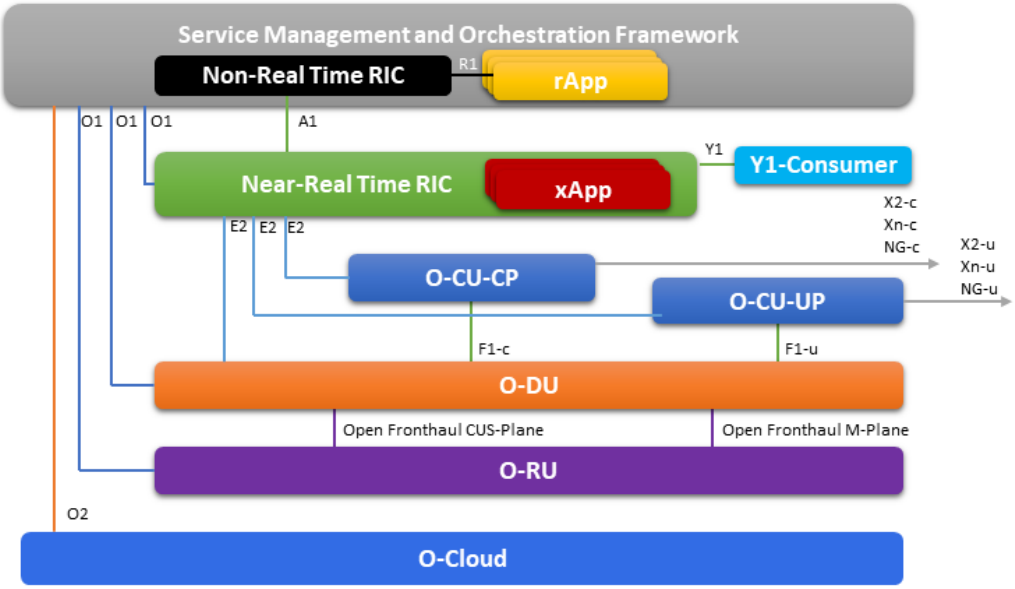

Figure 3.1: O-RAN Architecture [1]

This Open RAN initiative is being led by the O-RAN [6] and the Telecom Infra Project (TIP) [7]. The O-RAN is a partnership made up of numerous telecommunication conglomerates, academic institutions as well as software and hardware vendors. They form the main driver for producing the specifications and standardization documents that current and future Open RAN implementations are based upon. Furthermore, a sub-branch of the O-RAN Alliance, the O-RAN SC, develops an O-RAN compliant reference implementation, with the goal of producing a solution suitable for industrial implementation [8]. The TIP fulfills the role of driving innovation and industry collaboration, to support and increase the adoption of Open RAN. Furthermore, TIP is also an active part in testing and validating Open RAN implementations and conducts field- and commercial trials.

3.2 Kubernetes

The adoption of cloud-native technologies, such as Kubernetes, by the telecommunication industry is a key enabler for the deployment of Open RAN. The trend towards a software-defined RAN architecture, based on open interfaces, open-source software and containerized applications, requires a robust and scalable orchestration platform. Kubernetes supports the deployment and management of containerized applications in a scalable and resilient manner, enabling high availability and fault tolerance, which is essential for critical applications such as RAN deployments.

3.2.1 Virtualization and Containers

The fundamental technology behind cloud native applications is the use of virtualization. Virtualization forms an abstraction layer over the hardware by dividing the existing hardware of the host system and creating multiple individual Central Processing Units (CPUs), blocks of Random Access Memory (RAM) and storage. This abstraction of hardware allows to granularly divide system resources and assign them individually to Virtual Machines (VMs) or containers.

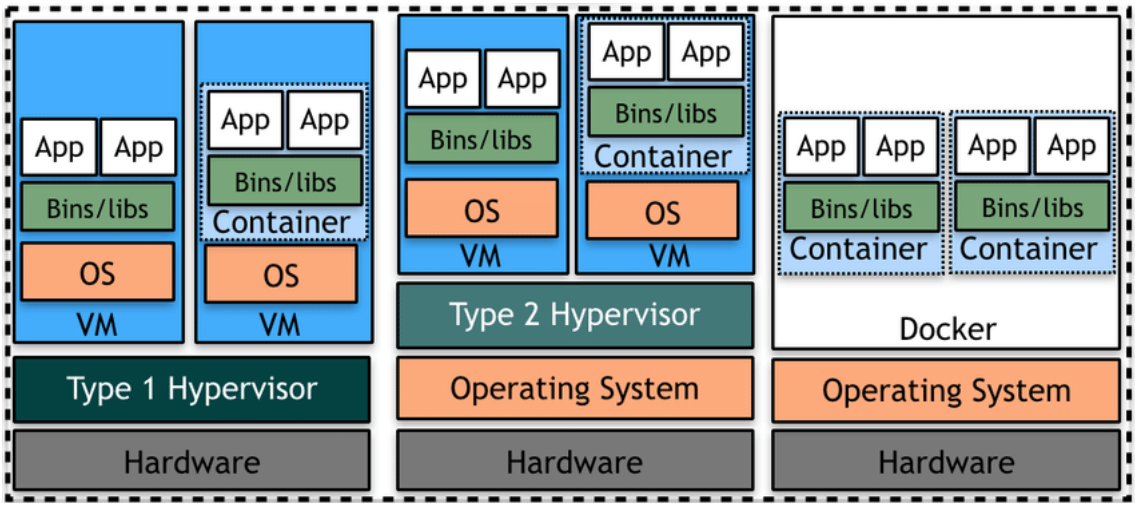

Figure 3.2: Types of Virtualization [9]

VMs operate in an isolated environment with their own operating system, kernel and libraries. This isolation provides a high degree of stability and security, as the failure of one VM does not affect other VMs on the same host system. However, the overhead of running a VM and startup time is higher than that of a container. The component responsible for managing the VM is the hypervisor, also known as the Virtual Machine Monitor (VMM). The hypervisor is responsible for managing the VMs lifecycle and their resources, such as CPU, RAM and storage. There exist two main types of hypervisors, Type 1 and Type 2. Type 1 hypervisors run directly on the host's hardware, while Type 2 hypervisors run on the host's operating system 3.2.

Containers are a lightweight form of virtualization, that provide a way to package applications and its dependencies into a single unit and run them in an isolated yet consistent manner. Containers share the host system's kernel, hence applications running in containers make direct system calls to the host system's kernel. This approach allows for a more efficient use of system resources, however, due to the thinner isolation from the host poses a higher security risk, since a compromise of a container can lead to a compromise of the entire host system. The component

responsible for managing containers is the container runtime, such as containerd which responsible for the container lifecycle (creation, starting, stopping and deletion of containers) and RunC for container execution. The isolation layer of containers is provided by the Linux kernel, through the use of namespaces, cgroups, capabilities. Namespacing provides the isolation from other processes, host networks, users, groups and filesystems, among others. While cgroups allow for the definition of granular resource constraints, in order to manage and restrict the available resources provided by the host system. Lastly capabilites provide a fine-grained control of priviliges, allowing containers to run with the least amount of priviliges required in order for the application to function.
Containers offer additional flexibility in the operation and deployment of software. By decoupling and disaggregating large applications into individually scalable units, isolation between applications is enhanced, as the failure of one container does not affect other containers if no dependencies are involved. Even a failure of an entire node can be compensated by an orchestration platform, since the orchestrator can reschedule all containers to a new node. However, containerization also introduces new security challenges, as containers share the kernel of the host system. Therefore, the isolation between containers and the host system, is comparitively weaker than the isolation between virtual machines and the host system. This degree of isolation can be further minimzed through misconfiguration, such as the use of privileged containers, mounting host paths, adding unnecessary capabilities or running containers as root. All these factors can lead to a container compromise and potentially an escape to the host system. With the criticality of cellular networks in mind, adoption of best practices for container security is essential to mitigate such attack vectors.
The use of virtualization and containerization is being adopted by the telecommunication industry in order cut costs through the use of COTS hardware, as well as scale and manage applications and resources more efficiently. This approach enables the deployment of containerized applications on any platform that supports the container runtime, and supports the shift to a microservices architecture, with the ability to deploy software-defined NFs as containerized application. However, the deployment and operation of single containers at scale requires a container orchestration platform, such as Kubernetes.

3.2.2 Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, management and operation of containers at scale. It offers a various advanced features for orchestrating containers, such as replication, load balancing, automated rollbacks and self-healing capabilites. This makes Kubernetes extremely resilient and ideal for deploying and managing large and complex applications, such as Open RAN. However, the complexity of Kubernetes also introduces additional security challenges which will be addressed in a later chapter. It was invented by Google in 2014 and is now maintained by the Cloud Native Computing Foundation (CNCF). The Kubernetes architecture features a master-slave architecture, with the control plane (master) being responsible for managing the cluster, and the worker nodes (slaves) responsible for executing the containerized workloads.

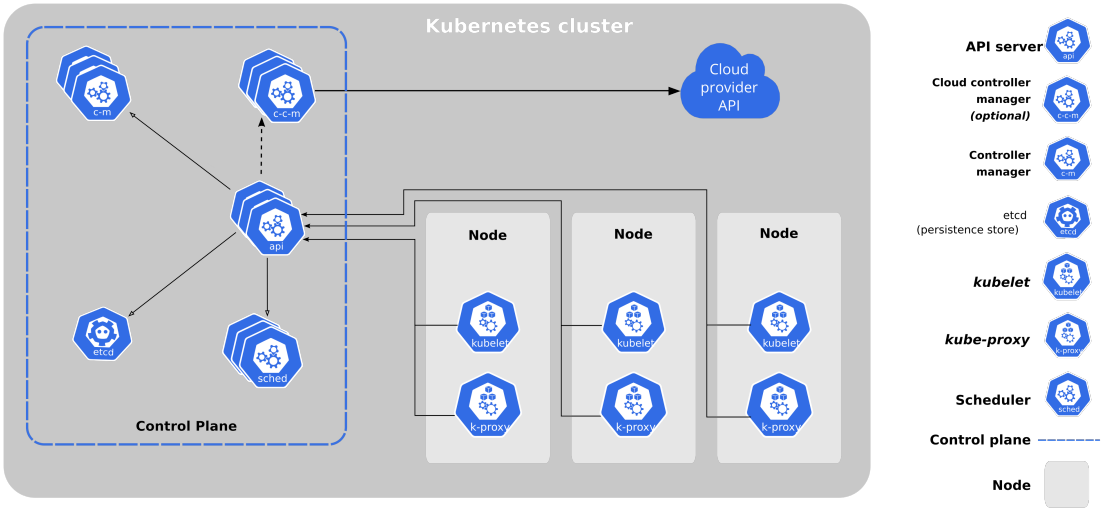

Figure 3.3: Kubernetes Architecture [10]

Control Plane

- **Kube-API:** The Kubernetes Application Programming Interface (API) server is the central component and frontend of the control plane. It exposes the Kubernetes API, which allows for the interaction with the cluster and its resources.
- **etcd:** The etcd component is a distributed key-value store that stores the entire cluster state and configuration data. It serves the purposes of a backing store for Kubernetes and is accessed by the API server to retrieve and write data.

- **Kube-Scheduler:** The Kubernetes scheduler is responsible for scheduling unassigned workloads, in the form of pods, to worker nodes in the cluster. The scheduler process monitors for unscheduled pods and based on the available resources on the worker nodes and the resource requirements of the pod, schedules the pod to the most suitable worker node. Other factors that influence the scheduling decision are for example affinity and anti-affinity rules, as well as data locality.
- **Kube-Controller-Manager:** The Kubernetes controller manager is responsible for running the core controller processes that regulate the cluster. These controllers include the node controller, job controller, ServiceAccount controller, and endpoint controller, among others.
- **Cloud-Controller-Manager:** The cloud controller manager is responsible for interacting and linking the cluster with the cloud provider's API.

Node

- **Kubelet**: The Kubelet is the main process running on each node in the cluster. It is responsible for running containers on the node and ensuring their health.
- **Kube-Proxy**: The Kube-proxy maintains the networking rules on the nodes. These rules dictate the network communications available to the pods from inside or outside the cluster.
- **Container Runtime**: The container runtime is responsible for managing the container lifecycle and executing the actual containers on the node [10].

Operating Kubernetes requires many numerous hardware and software components, such as COTS hardware, a hypervisors, an operating systems for hosting the control plane and worker nodes of the container orchestrator and Kubernetes itself. Moreover, running containerized worloads also requires additional software components, such as container runtimes, that are responsible for managing the containers for example containerd and RunC. This hardware and software extends the threat surface dramatically. Adversarys can track vulnerabilities of all these components on publicly availalbe vulnerability databases, such as the National Vulnerability Database (NVD) database. Furthermore, they may profit of publicly avaialable Proof of Concept (PoC) exploits or use the information to craft their own exploits.
The maintainers of Kubernetes employ fast release cycles, with a new release being published every four months. Furthermore, the support period for each release amounts to merely 14 months [11][12]. These rapid changes in the Kubernetes ecosystem, can lead to the use of unsupported software by MNOs. This in turn may lead to the operation of vulnerable software in mobile networks. Since the discussed

components are mostly open-source, vulnerabilities are publicly disclosed, hence this knowledge can be used by adversaries to develop exploits and compromise the targeted deployments.

3.2.3 CI/CD

The adoption of cloud-native technologies also requires a shift in the development, deployment and operations process. The term CI/CD refers to an ongoing process of integrating new code into repositories, automatically building and testing the code and deploying this code to production. This process also allows for the integration of security tests, such as Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), to ensure that the code meets certain security and compliance requirements before being released into production. The automation of the CI/CD pipeline allows for fast-paced development and deployment of applications, ensuring that application grow and evolve with small changes and updates. This approach reduces the risk of introducing new vulnerabilites as the code is continously validated and tested. Steadiliy integrating minor code changes into a complex system, such as an Open RAN, increases the likelihood of identifying flaws early on. This approach simplifies identifying and addressing code issues and vulnerabilities while also being more cost-effective.

3.2.4 DevSecOps

The term Development, Security and Operations (DevSecOps) refers to the integration and automation of security practices into every stage of the Software Development Life Cycle (SDLC). One of the primary goals is to "shift security left" in regards to the SDLC, since integrating security testing and validation early on in the development process, vastly helps in identifying and mitigating vulnerabilites early on. Therefore, this approach strives to enforce security by design, making it an integral part of the development, deployment and operations procedures and a shared responsibility accross the entire SDLC.
The goal of this investigation is to create a strategy for security testing of an Open RAN deployment, that is based on the principles of DevSecOps. This theoretical framework can aid in identifying and mitigate vulnerabilities, by assessing the threat landscape of the deployment and integrating security testing into SDLC. However, the primary objective is to develop a strategy for conducting adversary simulations, aiding in the creation of DFIR procedures, and advancing capabilities to retospectively identify adversaries and the TTPs they used to achieve their objectives.

3.3 Adversary Simulation

The term adversary simulation refers to the practice of simulating adversarial activity. The main goal of these simulations in the context of this research, is to enhance the detection and incident response capabilities in Open RAN deployments. This is achieved through simulating past and potential future attacks, by identifying potential vulnerabilities and weaknesses, also known as attack vectors, and exploiting them in a safe and controlled manner. This process can be automated in order to continuously test the security posture of an organization, and integrated into the CI/CD pipeline, to pass specifically defined security gates and thereby adhere to a certain level security.
Another category of replicating adversarial activity, is the emulation of real-world adversaries, named adversary emulation. This approach deals with the simulation of the TTPs of adversaries based on authentic artifacts and forensic data gathered by security analysts, that are based on past breaches or attempts thereof. Due to the lack of genuine data of adversarial activity in the context of Open RAN, the focus of this investigation is on the simulation of adversarial activity, rather than the emulation of real-world adversaries [13] [14].
Simulating adversarial activity requires a thorough understanding of the threat landscape, the attack surface and the potential attack paths that adversaries might pursue. This knowledge is essential for the creation of realistic attack scenarios. Therefore creating a comprehensive threat model forms the foundation for the simulation of adversarial activity. Simulating single isolated techniques that adversaries might employ, is known as atomic testing. These tests can be automated and validated. However, since atomic tests only represents a single step in the attack chain, they are not as useful for simulating key behaviours of a potential attacker against its target. On the other hand, engaging red teams to simulate and emulate attack behaviour is costly and cannot be automated. Consequently, micro-simulations are used in the scope of this project to develop attack scenarios that incorperate key techniques employed by adversaries to target and compromise an Open RAN deployment. This approach is advantageous due to the light-weight nature of the simulations, that are suitable for automation, yet incorperate the core techniques used by adversaries. These micro-simulations are developed with the goal to improve the ability to detect and respond to perpetrators, in order to protect the Confidentiality, Integrity and Availability (CIA) of relevant assets in Open RAN.
There exist other forms of adversary simulation, for example penetration testing. However, this approach differs in it's methodology and objectives. Penetration testing has the primary goal of identifying vulnerabilites in a system, software, network or

infrastructure through their expoitation [15].

3.3.1 Assumed Breach Model

The assumed breach model is an approach that presumes adversaries have already gained initial access to the target system, network or infrastructure. This methodology is based on the assumption that the defenses of a system can be overcome and that adversaries are already present within [16]. This model is useful in the context of this project, since the simulation of breaching an outer perimeter and gaining initial foothold requires the introduction of new vulnerabilities into that system or knowledge of existing vulnerabilities, as well as developing or possessing an existent PoC of an exploit. By adopting this model, the primary focus is dedicated to simulating and developing models for adversarial behaviour within the system, as malicous actors will always find or develop the means to gain some extent of direct or indirect access to their target. However, reacting and responding to these threats is essential to protect the CIA of the target's assets, as initial foothold does not directly lead to a full compromise of the target and achievement of the adversaries objectives.

3.3.2 Threat-driven approach

The threat-driven approach is a methodology that has its primary focus on the identification and mitigation of threats, instead of judgig ones security posture by potential vulnerabilities and the implementation of security controls. In the compliance-drive approach organization achieve to pass compliance checks, however, fail to meticulously evaluate the threat landscape that their organization is exposed to. Solely focusing on compliance, does not guarantee that the organization meets a certain level of security and is protected from potential threats. Implementing security controls without a thorough understanding of the threat landscape, or conducting vulnerability scans without verifying if these vulnerabilities are actionable or have a high impact, can lead to a brittle security posture, exposing the organization to a higher risk of compromise.
The threat-driven approach in turn integrates a comprehensive threat analysis and threat intelligence into all phases of the SDLC. This methodlogy ensures that threats are adequately addressed and mitigated, resources efficiently allocated and overall risks minimized. Thereby, attack vectors of high threat potential are prioritized and addressed first. The threat analysis is conducted in a continuous manner, to adapt to an evolving and dynamic threat landscape. This leads to a more resilient security

posture, while also ensuring that resources are allocated efficiently and effectively [17].
This research project aligns with a threat-driven approach, as few vulnerabilities are known and the threat landscape of Open RAN is not well understood. Furthermore, the effectiviness of proposed counter-measures has yet to be validated and the impact of potential threats on the CIA of the Open RAN assets has not been practically studied. Therefore this project attempts to study the said threat landscape, develop a strategy for simulating adversarial activity, and priotize the attack paths that adversaries might pursue based on the anticipated impact on the CIA of the Open RAN assets.

3.3.3 Threat Detection Engineering

Threat detection enigneering is the practice of developing and refining processes to detect potential threats in a defined environment [18]. The research project FORAN develops a strategy and the necessary capabilities to retrospectively detect and analyse adversarial activity in Open RAN deployments, as a result the practice of threat detection engineering is of essence. The methodology and strategy presented in this paper, serves in advancing this undertaking by analysing and identifying critical assets as well as threats that pose a substantial risk to security of the Open RAN deployment. Furthermore, this report analyses what adverserial events are likely to occur and how these events can be generated in an automated and controlled fashin, to test and validate the detection and response capabilities.
Frameworks that are important in the context of threat detection engineering and especially the process of detection validation are discussed in the following section.

3.3.4 Cyber- and Unified Kill Chain

The term kill chain originates from the military and is used to model the process of an attack. It was adapted to the domain of computer security with the introdcution of the Cyber Kill Chain (CKC) by Lockheed Martin, and describes the stages of an attack from reconnaissance to actions on the objective. The CKC aids in understanding the steps adversaries take to compromise their target, thereby also helps in developing detection strategies. Even though CKC is a linear model and cyber intrusions generally do not follow a linear path, identifying Indicator of Compromises (IoCs) in each stage of the kill chain supports in generating traces for detection validation and incident response [19] [20].
The CKC model comprises seven stages, used to describe the methodology of an

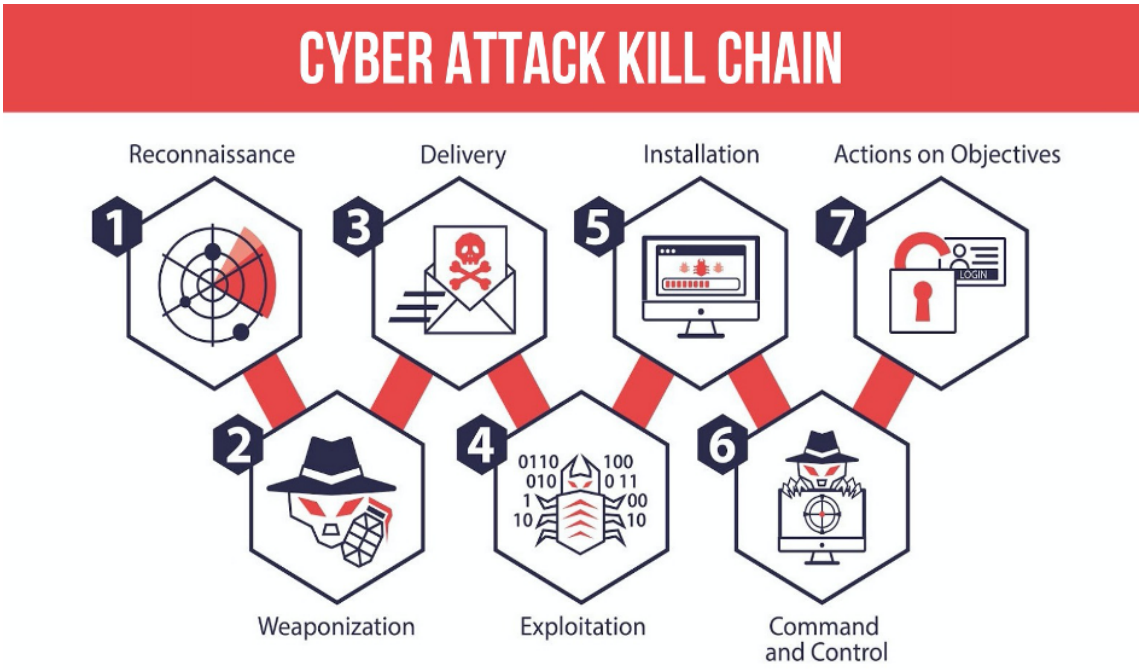

Figure 3.4: Cyber Kill Chain [21]

attack. These phases are depicted in Figure 3.4 above. The cyber kill chain was further modified by Dutch security expert Paul Pols, in order to compensate some of the shortcomings of the original model. The extended model, the Unified Kill Chain, is a more comprehensive model that includes a total of 18 stages. This model addresses more attack vectors that occur within the perimeters an organization, which is critical for validating detection capabilities, as it is challenging to gain visibility into attacks and the preparation thereof outside of an organization's perimeters [22]. Since, the Unified Kill Chain is a more suitable model for the development of detection strategies, it will be used in the context of this research project. The phases of the Unified Kill Chain model are depicted in Figure 3.5 below.

#	Unified Kill Chain
1	Reconnaissance
2	Resource Development
3	Delivery
4	Social Engineering
5	Exploitation
6	Persistence
7	Defense Evasion
8	Command & Control
9	Pivoting
10	Discovery
11	Privilege Escalation
12	Execution
13	Credential Access
14	Lateral Movement
15	Collection
16	Exfiltration
17	Impact
18	Objectives

Figure 3.5: Unified Kill Chain [22]

3.3.5 Pyramid of Pain

The Pyramid of Pain is a model that illustrates the different types of IoCs used to detect and respond to adversarial activities. These IoCs are categorized based on the effectiveness of deterring adversaries. Correspondingly, these categories also represent the degree of difficulty for security analysts and incident responders to build detection capabilites for these distinct type of indicators [23]. The bottom layer of the pyramid depicts the most straightforward indicators to detect, hashes. However, the simplicity of changing hashes, enables adversaries to rapidly adapt to changing circumstances, thereby reducing the effectiviness of the detection measures implemented. The indicators hashes, IP addresses and domain names are simple to detect, therfore, only higher layers of the pyramid are discussed in the context of this research project. The higher layers comprise network/host artifacts, the tooling utilized by adversaries and the TTPs they employ. Implementing and developing capabilites to detect these indicators is complex, as the methodology and tooling arsenal used by adversaries is constantly changing and evolves over time. The

Pyramid of Pain is depicted in Figure 3.6 below.

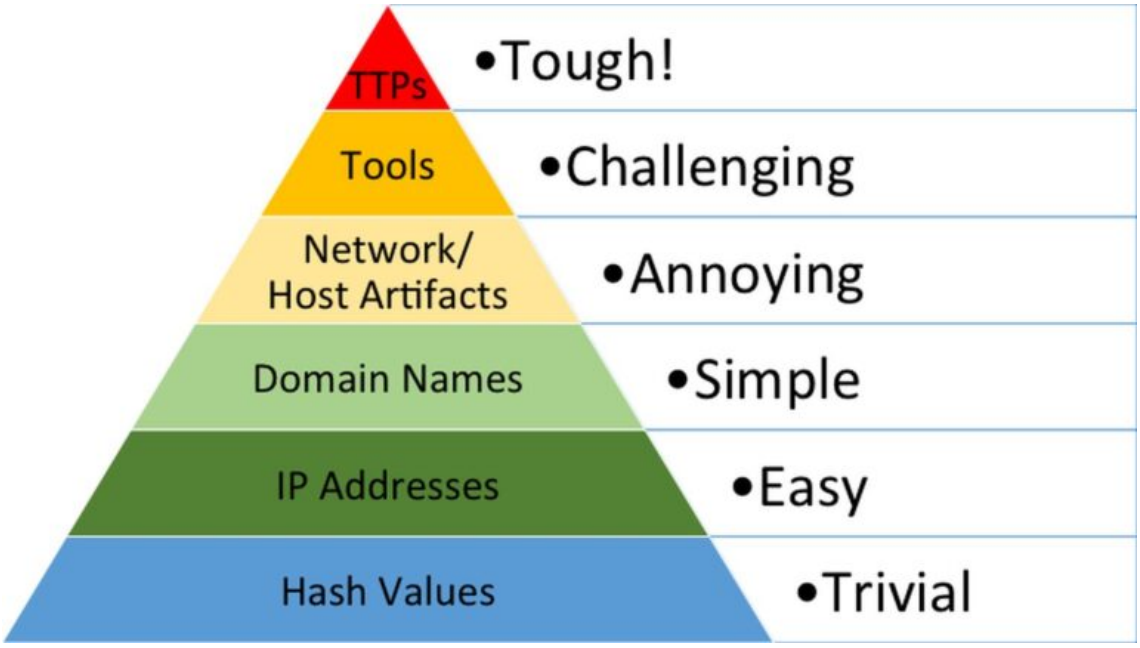

Figure 3.6: Pyramid of Pain [24]

3.3.6 MITRE ATT&CK

The Mitre Corporation is a non-profit organization that manages research and development centers funded by the U.S. government. The areas of focus of the Mitre Corporation are diverse, however one major area of interest is cybersecurity. Mitre maintains and operates various projects and frameworks used by the security community to enhance their security posture. Some of these research projects include the tracking of CVEs and Common Weakness Enumerations (CWEs) as well as the Cyber Analytics Repository (CAR) and the project ENGAGE. However, the framework of most interest to this research project, is the Adversarial Tactics, Techniques, and Common Knowledge (MITRE ATT&CK) framework. This framework is a knowledge base of adversarial TTPs based on real-world threat intelligence, security research, past security breaches and data gathered from publicly available sources. The framework is highly beneficial for modeling and analyzing the behavior of adversaries, to detect as well as emulate their behaviour to enhance ones security posture [25] [26].

MITRE ATT&CK Matrix Components

- **Tactics**: Tactics represent the objectives and strategy employed by adversaries to achieve their higher-level objectives.

- **Techniques**: Techniques exemplify particular implementations of actions, executed to achieve a tactical objective.
- **Sub-Techniques**: Sub-techniques are more granular actions that represent one approach implementing a specific technique.

The MITRE ATT&CK framework is structured in the form of a matrix, with columns representing adversarial tactics and rows representing individual techniques, and sub-techniques. There exists multiple variations of this matrix, some of which are tailored to specific industries of even technologies. These matrices are beneficial in the process of threat modelling and are discussed in more detail later on.

3.3.7 Security Teams

There exist a variety of security teams responsible for different aspects of security within an organization. Employees responsible for security monitoring and analysis, as well as for incident response are categorized as the blue team. The red team consists of employees that perform offensive security operations, such as penetration testing and adversary simulation. The purple team consists of both red- and blue team operators and is responsible for coordinating the activities of the red and blue team, and ensuring efficient and effective collaboration. Furthermore, the purple team aids in knowledge transfer between the red and blue team, to ensure that both teams are kept up-to-date and informed about the latest advances and threats that both teams analyzed [27] [28] [29].

Security Teams

- **Red Team**: The red team is responsible for offensive operations and simulating adversarial activity.
- **Blue Team**: The blue team is in charge of defensive operations and upkeeping the security posture of the organization and its assets.
- **Purple Team**: The purple team oversees the coordination and knowledge transfer between the red and blue team, and is staffed with both red and blue team operators.

3.4 Digital Forensics and Incident Response

The practice of DFIR combines two main fields of expertise, digital forensics and incident response. Digital forensics involves the collection, examination, analysis and reporting of digital evidence to support cyber incident investigations [30]. On

the other hand, incident response entails a set of processes and procedures executed subsequent to the detection of cybersecurity incident. These set of processes aim to minimize the impact of the incident, eradicate the threat and securely restore the affected systems. These objectives ensure business continuity and the protection of the CIA of an organization's assets [31].

Incident Response Lifecycle (NIST)

- **Preparation**: The prepration phase involves the development of incident response capabilities. This includes the creation of incident response plans, policies and playbooks, establishing coomunication guidelines, distributing contact information, building and maintaining a tooling arsenal and training the staff to prepare for actual incidents. Furthrermore, the preparation phase aims to minimize the number of incidents by implementing preventative measures.
- **Detection & Analysis**: This phase comprises the detection and rapid analysis of potential IoCs. The objective is to identify whether an incident occurred, and determine the scope of the incident to initiate a quick response. Furthermore, the potential impact of the incident is assessed, to allocate resource effectively.
- **Containment, Eradication & Recovery**: This phases aims to control the incident, minimize the impact, eradicate the threat and securely restore the affected systems. This stage entails short-term and long-term strategies and procedures to eradicate the threat and limit its impact.
- **Post-Incident Activity**: The post-incident activities consist of reviewing and learning from the past incident. This includes creating thorough documentation and updating existing policies and response plans to more efficiently handle future incidents [31].

This research project focuses on a specific aspect of DFIR, the retrospective detection and analysis of adversarial activity in Open RAN deployments. In particular, the development of a strategy for conducting adversary simulations, to enhance the detection and response capabilities of the organization. Therefore, this research paper aligns with the "Preparation" phase of the incident response lifecycle.

4 Threat Modeling Principles

The primary objective of this chapter is to introduce the concepts and principles of threat modeling. Furthermore, the chapter discusses the value of threat modeling for assessing threats appropriately and the methodology, models and frameworks that aid in this process. Moreover, the groups participating in threat modeling are outlined and what purpose threat modeling serves, especially in the context of this project.
Threat modeling servers a significant role in enhancing an system's resiliency from threats. The dynamic and constantly evolving threat landscape, requires organizations to continuously reassess their risk exposure to implement counter-measures and improve their monitoring and incident response capabilities. Threat modeling aligns with the threat-driven approach, which is a proactive approach to security. It shifts to focus away from merely achieving a certain level of compliance, and instead focuses on identifying and mitigating threats to the system.

4.1 Methodology

Threat modeling is a systematic process to identify, assess, prioritize and address threats that can potentially lead to the compromise of a system's security. The main objective is to identify threats early and prioritize them appropriately. This allows for an efficient allocation of resources and aids in selecting a suitable risk handling strategy. This strategy might entail implementing preventative measures, or to enhance visibility into the system, to detect threats more effectively, or to ignore threats due to their low level of risk. Thereby, the overall impact of threats is minimized, resources utilization and effectiveness maximized and the system's security posture improved.

4.1.1 Threat Modeling Process

The high-level process of threat modeling is outlined below. The process is iterative and continuous and can be tailored to the specific requirements and circumstances of the examined system and security operation utilizing the threat model.

High-Level Process

- **Objective**: The initial step involves defining the purpose of the threat model. This includes the definition of how the threat model is integrated into future development processes or security operations and the expected outcomes of using the threat model in the defined area of application.
- **Scope and Identification of Assets**: This step comprises the definition of the scope of the threat model and the identification of the relevant assets. This covers the definition of the system's boundaries as well as specifying the level of depth that is required to model the system. The term assets represents all the components that make up, interact or are processed by the system. Assets are the entities critical to the system's functioning and operation, therefore it is essential to identify and prioritize them. In order to adequately identify relevant assets, system modeling is used to represent the systems composition and internal structure. In system modeling, a graphical representation of the system is developed, that represents the system's architecture, including all the components, data flows, data stores and interaction with internal and external systems. A widely adopted approach to system modeling is the use of DFDs. At last, this stage outlines the relevant stakeholders involved in the threat modeling process.
- **Identify and Assess Threats**: The subsequent step of the process involves the identification of potential threats to the system, with the objective of determining as many threats as possible, especially ones that pose a high risk to the system's security. At first, potential threat actors are defined, these actors are classified according their varying level of expertise and sophistication in conducting attacks. Furthermore, since both internal and external threat actors must be considered, the degree of access they have to the system is also specified. Thereafter, potential threats are identified and classified according to a threat modeling framework such as STRIDE, explained in a later section. The classification and prioritization of threats is important in order to assess the risk and potential impact of threats appropriately. Subsequent to identifying and ranking individual threats, a tactical approach can be utilized to develop more complex contexts by chaining threats, using models such as the CKC. These attack scenarios can be formally represented in attack trees.
- **Develop and Implement Mitigation Strategies**: Following the thorough identification and assessment of threats, the next step is to develop suitable mitigation strategies. These are the standard risk handling strategies such as risk acceptance, risk avoidance, risk transfer and risk mitigation. The

mitigation strategies are selected based on several factors, two of which are impact and likelihood of the threat occurring. There exist multiple threat model frameworks that cover different factors used to assess the severity of threats.

- **Monitor, Review and Iterate**: The last stage of the threat modeling process is to monitor, review and iterate the implemented threat model. Due to the ever-evolving threat landscape in a product that is undergoing changes and deployed in production, new threats are introduced or have not yet been identified. This is due to the high complexity of larger software-based systems. Furthermore, as the system changes, updating the system models and documentation to reflect the changes is of utmost importance [32] [33].

The subsequent subsections discuss two main resources that provide insightful guidance on threat modeling, the 'Threat Modeling Manifesto' and the Open Web Application Security Project (OWASP).

Threat Modeling Manifesto

The Threat Modeling Manifesto was established by a community of experts in threat modeling, ranging from security professionals to academics and authors. The objective of the collective is to educate and improve security through the adoption of threat modeling. The Threat Modeling Manifesto provides a set of principles and guidelines for threat modeling. First and foremost, it defines the key questions to ask when conducting threat modeling:

- What are we working on?
- What can go wrong?
- What are we going to do about it?
- Did we do a good job?

Furthermore, the manifesto emphasizes key values and principles, such as "Doing threat modeling over talking about it" or "People and collaboration over processes, methodologies, and tools". The recommended best practices and pitfalls or patterns and anti-patterns as defined by the collective, tremendously help in guiding the threat modeling process as well as ensuring that time is not wasted on unnecessary activities [34].

OWASP Guidelines

The OWASP provides a comprehensive guide to threat modeling, with valuable resources such as the 'Threat Modeling Process', 'Threat Modeling Cheat Sheet' and

the system modeling tool 'OWASP Threat Dragon'. By defining the threat modeling process, recommending frameworks and providing an example implementation of the threat modeling process, this resource serves as beneficial guidance for organizations to implement a structured approach to threat modeling [32] [33] [35] [36].

Terminology

This section provides an overview of the key terms and concepts relevant to threat modeling and adversary simulation in this research project.

- **Threat Vector**: A path utilized by a threat actors to attempt to compromise a network or system.
- **Threat Surface**: The sum of all threat vectors that can be utilized by a threat actor to compromise a network or system.
- **Threat Matrix**: A structured matrix listing tactics and techniques used by threat actors to target a specific system, technology or environment.
- **Threat Scenario**: A chain of threats interacting with the target environment or network to achieve a specific malicious objective.
- **Threat Profile**: A comprehensive summary of threat actors that features their objectives, capabilities and methodology used to target a system.

4.2 Frameworks

This section covers popular threat modeling frameworks that are utilized in the industry. These frameworks help in establishing a structured approach to threat modeling and provide a systematic way to identify, assess and prioritize threats.

4.2.1 STRIDE

The STRIDE model is a threat modeling framework that was developed by Microsoft. The framework is used to assess threats according six categories, where each category represents the violation a specific security property of the modeled system. THereby, each threat is classified on the security properties it violates [37].

Category	Violates	Examples
Spoofing	Authenticity	Impersonation of a legitimate entity using stolen credentials.
Tampering	Integrity	Malicious modifications of information.
Repudiation	Non-repudiability	The removal or forgery of information to avoid attribution of a malicious action.
Information Disclosure	Confidentiality	Access or exfiltration of secret information.
Denial of Service	Availability	Rendering a system, service or data inaccessible.
Elevation of Privileges	Authorization	The process of gaining more access and permission to access additional resources, systems or networks.

Table 4.1: STRIDE categories and the respective security properties they violate. [32]

4.2.2 MITRE ATT&CK

The MITRE ATT&CK framework is a comprehensive knowledge base of adversarial tactics, techniques and procedures based on real-world analyses of incidents and threat intelligence. The framework was already discussed in section 3.3.6.

4.2.3 Attack Trees

The concept of attack tree was introduced by Bruce Schneier in 1999. Attack trees are a formal and methodical approach for describing the security of a system. The attack tree represents a hierarchical structure of threats that lead to a defined adversary objective, depicted as the root node. The leaf nodes of the tree portray the various threats posed to the system. Thereby, an attack tree represent various attack scenarios and strategies that may be employed by an adversary to compromise a system. A single path through the tree represents a specific attack scenario, named attack path in this context. Individual nodes of the tree represent the various techniques an adversary employs to traverse along the tree towards the root node. There exist different types of nodes, namely AND nodes and OR nodes. AND nodes require all child nodes to be true to reach the parent node, while OR nodes require only one child to be true to reach the parent node. Attack trees are a valuable tool for threat modeling, as they enable the representation of complex attack scenarios in a structured and and simple manner [38].

4.2.4 PASTA

Process for Attack Simulation and Threat Analysis (PASTA) is a risk-centric approach to threat modeling and was presented in 2015 by Tony Ucedavelez and Marco Morana. The framework is based on a seven-step process outline below.

- **Define Objectives**
- **Define the Technical Scope**
- **Application Decomposition and Analysis**
- **Threat Analysis**
- **Weakness and Vulnerability Analysis**
- **Attack Modeling & Simulation**
- **Risk Analysis & Management**

The focus of a risk-based approach ensures that the most impactful threats are identified and addressed. Furthermore, PASTA can be utilized to model non-traditional threats, such as the disclosure of confidential information through social media. Moreover, the simulation of attacks aids in security testing and verifying mitigation strategies, but also enables the improvement of incident response capabilities [39] [40].

4.2.5 DREAD

The Damage, Reproducibility, Exploitability, Affected Users, Discoverability (DREAD) model is a risk assessment model that was developed and utilized by Microsoft. The model is used to assess the severity of a threat using five categories, where each category is rate on a scale from 0 to 10. The categories are presented below.

- **Damage**
- **Reproducibility**
- **Exploitability**
- **Affected Users**
- **Discoverability**

The scores are summed up to allow for a meticulous prioritization and assessment of each threat. The model has its limitations, due to the subjective nature of the ratings. However, it is still valuable in threat analysis and can aid in the relative prioritization of threats [41] [42].

5 Threat Assessment

This chapter outlines the architecture and threat landscape of an Open RAN implementation based on the O-RAN Alliance specifications and Kubernetes as container orchestrator.

5.1 O-RAN SC Architecture

The O-RAN Software Community (SC) is a branch of the O-RAN Alliance, that actively develops a reference implementation using the O-RAN specifications and architecture, as mentioned in 3.1. This implementation has the purpose of serving as a reference implementation for industry deployments [8].
The O-RAN SC I-Release is the current release of the O-RAN SC and was released in December 2023 [43]. This release is the key focus of this research project. The following section provides a rough overview of the architecture and most significant components and interfaces of the O-RAN SC implementation.

O-RAN Components

- **O-Cloud**: The O-Cloud encompasses physical infrastructure nodes that host the O-RAN components, as well as the software used to manage and orchestrate them. This platform can be based on OpenStack or Kubernetes using COTS hardware. There exist varies modes of deployment, such as private cloud (on-premise), public cloud, or hybrid cloud. Furthermore, the are multiple disparate scenarios that describe the location and distribution of these components across the O-Cloud nodes, in order to fulfill the stringent latency requirements [44].
- **Service Management and Orchestration (SMO)**: The SMO is responsible for the management, orchestration and monitoring of O-RAN components. It also hosts the non-RT-RIC and thereby provides the intelligence and policies for non-real-time procedures within the RAN. Moreover, the SMO conducts Fault, Configuration, Accounting, Performance, Security (FCAPS) operations and is in charge of network configuration and tracking of resources. Furthermore, the SMO interacts with the O-Cloud through the O2 interface, to align and manage

the infrastructure resources with the operational requirements of deployed NFs [45].

- **non-RT-RIC**: The non-RT-RIC is responsible for the management and optimization of RAN functionality that do not require real-time processing. The control loops run with timescales above one second. These tasks entail providing intelligence and guidance to the near-RT-RIC, and advanced data analytics through the use of AI/ML models to predict and optimize RAN operations. The functionality of non-RT-RIC can be augmented through the use of rApps, that aid in the data-driven analysis and optimization [45].
- **near-RT-RIC**: The near-RT-RIC is in charge of real-time optimization and control of the RAN components, with control loops operating within a timescale of 10 milliseconds to one second. The near-RT-RIC's functionality can be extended through the deployment of xApps. These applications take care of specific optimization functions, such as load balancing or hand-over procedures. The near-RT-RIC is connected to the RAN nodes through the E2 interface, used to gather telemetry data and enforce optimization decisions.
- **Central Unit (O-CU), Distributed Unit (O-DU) and Radio Unit (O-RU)**: The disaggregated base station (NR Next Generation Node Base (gNB)) components are represented by the O-CU, O-DU and O-RU. The O-CU handles higher-level protocols, responsible for central control and management. The O-DU conducts real-time data processing, while the O-RU is in charge of handling radio signals [45].

Interfaces

- **E2:** The E2 interface connects the near-RT-RIC with the RAN nodes, also known as E2-nodes. This allows for real-time data collection and processing using the subsecond control loops in the near-RT-RIC. These control loops are utilized to optimize and exercise control over the RAN resources.
- **A1:** The A1 interface interconnects the near-RT-RIC with the non-RT-RIC. This enables the near-RT-RIC to receive policies and instructions on how to employ and use it's intelligence for optimizing RAN nodes.
- **O1:** The O1 interface connects the SMO with the near-RT-RIC and the disaggregated gNodeB components. This open interface is utilized for network management, such as functions from the FCAPS framework.
- **O2:** The O2 interface connects the SMO with the O-Cloud. This interface serves the purpose of managing and provisioning virtualized NFs. This entails operations such as monitoring, scaling and ensuring redundancy of the deployed

NFs [45] [46].

5.2 Laboratory Attack Environment

The exist multiple valid approaches to deploying Open RAN, therefore, since no single deployment model is agreed upon and applicable in all circumstances, only one is practically considered in this research. The deployment model followed for this research is based on the O-RAN SC I-Release, orchestrated with Kubernetes in a private cloud (on-premise). This laboratory environment is set up to simulate an authentic Open RAN deployment to evaluate the threat landscape under the given deployment conditions. The disaggregated base station components are not yet part of the evaluation, due to time constraints and missing integration in the laboratory environment. However, the final deployment model will include these components to cover the extended attack surface.
The laboratory environment runs on a Dell PowerEdge R650xs server featuring the Type-1 hypervisor, VMware ESXi 7. The O-RAN I-Release is distributed among two virtual machines, each featuring a Kubernetes cluster for orchestration purposes. One cluster operates the O-RAN near-RT-RIC and the other the non-RT-RIC, both are connected via the A1 interface, which is based on a RESTful Hypertext Transfer Protocol (HTTP)-API. A representation including additional systems for attack simulation and DFIR utilized in the FORAN project are depicted in the figure 5.1.

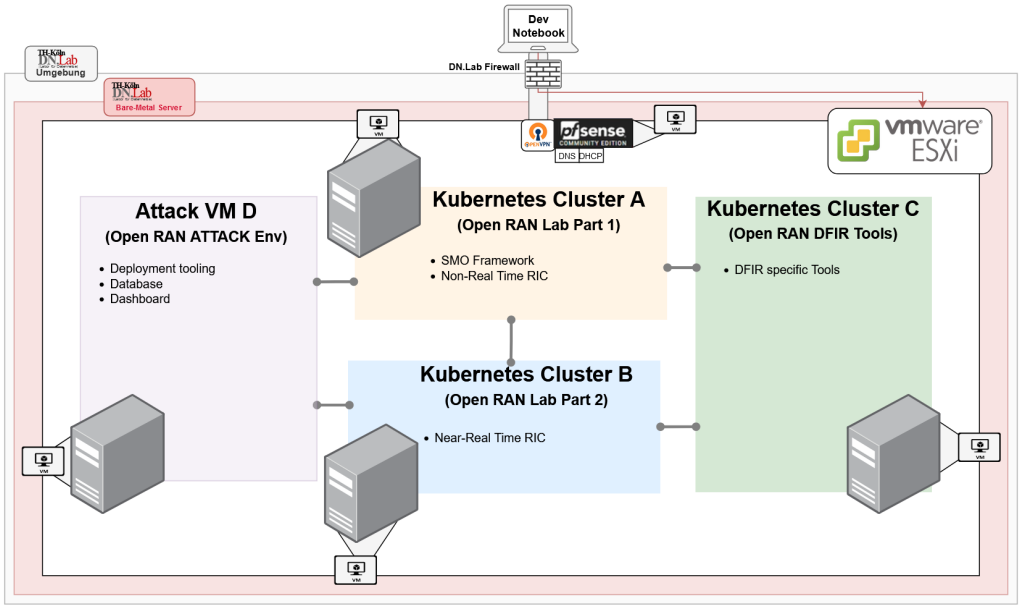

Figure 5.1: Laboratory Environment Setup [1]

5.3 Threat Actors

Assessing the threat landscape of the O-RAN I-Release entails a definition of relevant threat actors. These actors are categorized into internal and external threat actors. Furthermore, the scope of this research centers the focus at the environment of the O-Cloud and the near-RT-RIC and non-RT-RIC components. Establishing access to these components is complex from an external approach, especially when considering a private cloud deployment. Therefore, the threat actors are primarily internal, with the exception detailed in the subsequent section.

The O-RAN Alliance comprises multiple working groups, each with a specialized area of expertise. The O-RAN Alliance Security Working Group 11 (WG11) specializes in the security aspects of O-RAN and is in charge of defining security requirements and specifications. These specifications comprise a threat model including the identification of relevant threat actors and stakeholders involved in the O-RAN ecosystem [47]. Furthermore, the Federal Office for Information Security (BSI) has conducted research in the area of Open RAN and also determined several categories of malicious actors, that form the basis of the threat actor taxonomy presented in the next section [48].

5.3.1 Threat Actor Taxonomy

This section details a taxonomy of threat actors deemed relevant in the context of the O-RAN SC I-Release. The taxonomy categorizes threat actors based on their internal or external entry points and the broad level of access they possess to the O-RAN infrastructure. The following table 5.1 provides an overview of the identified threat actors.

Threat Actor	**Internal/External**	**Access**
Insider Admin	Internal	Medium
RAN Operator	Internal	High
Cloud Service Provider	External	Medium
xApp/rApp developer	External	Medium

Table 5.1: Threat Actors in the O-RAN SC I-Release ecosystem

Threat Actor Description

- **Insider Admin:** This category of threat actors is internal and possesses access to a subset of the O-RAN infrastructure. For example, an insider admin that administers the near-RT-RIC component. Moreover, a network admin with access to the traffic passing between O-RAN components is included into this category. This actor has medium level of access, indicating a lack of access to all components within the ecosystem. However, this actor may cause significant damage, due to the ability to compromise or disrupt entire nodes under their control.
- **RAN Operator:** The RAN operator is an internal threat actor with high level of access to the O-RAN infrastructure. The RAN operator may access any component within the O-RAN ecosystem, and can therefore impact the entire infrastructure.
- **Cloud Service Provider:** The cloud service provider is an external threat actor with medium level of access to the O-RAN infrastructure. This actor has the ability to affect the availability the entire O-Cloud, however, not access data or individual components within the O-RAN ecosystem. Furthermore, the cloud service provider may create or delete nodes that host O-RAN NFs.
- **xApp/rApp Developer:** The xApp/rApp developer is an external threat actor with medium level of access to the O-RAN infrastructure. This actor has the ability to remotely execute code on O-RAN components. Specifically, Remote Code Execution (RCE) is achieved through the release and adoption of malicious software in the form of xApps impacting the near-RT-RIC and

rApps affecting the non-RT-RIC.

5.4 Threat Frameworks

This sections details relevant threat frameworks, that are utilized to describe and classify threats. One of the most prominent frameworks, the MITRE ATT&CK, was already discussed in section 3.3.6.

5.4.1 CVE

The CVE framework is an initiative maintained by the MITRE Cooperation and other partners, to consistently describe and classify vulnerabilities. The catalog of vulnerabilities is publicly accessible and provides organizations with a standardized approach for addressing and processing vulnerabilities that affect their software and hardware ecosystem [49].

5.4.2 CWE

The CWE framework is a community-driven initiative, also maintained by the MITRE Cooperation, that provides a standardized approach to describe and categorize weaknesses in software and hardware systems. The list aids developers and architects in identifying potential weaknesses that could lead to vulnerabilities in their products, prior to release and deployment. Therefore, this framework aims to reduce the introduction of vulnerabilities by supporting comprehensive analysis of systems in the early stages of development [50].

5.4.3 CVSS

The Common Vulnerability Scoring System (CVSS) framework is an open standard for measuring the severity of vulnerabilities. This facilitates the process of prioritizing and implementing the according risk treatment strategies, upon identification of exposure to a particular CVE. The severity is assessed according to several metrics that are scored between zero and ten. The framework is maintained by FIRST Inc., and the current release version 4.0 [51] [52].

5.4.4 CAPEC

The Common Attack Pattern Enumeration and Classification (CAPEC) framework is a public catalog describing known attack patters used by threat actors to target

and exploit systems. The catalog is maintained by the MITRE Cooperation and supports cyber analysts to improve their understanding on how adversaries operate, as well as red teamers to enhance their effectiveness in simulating attacks [53].

5.5 Threat Enumeration

This section outlines the results of a comprehensive threat discovery relevant to the O-RAN SC I-Release implementation.

5.5.1 Threat Matrices and Collections

The following collection of threats are employed to enumerate and identify threats relevant to O-RAN SC I-Release.

Threat Matrices and Collections

- **Microsoft Threat Matrix for Kubernetes:** The Microsoft Threat Matrix for Kubernetes is a comprehensive collection of TTPs that target Kubernetes applications. The threat matrix is based on MITRE ATT&CK and aids in understanding how threat actors operate. This helps in the process of building capabilities and scenarios for attack simulation, as these TTPs can be replicated, such as with the sample implementation of techniques by the company Redguard AG [54] [55].
- **Containers Matrix:** The Containers Matrix is part of the MITRE ATT&CK framework and covers the TTPs leveraged against containerized applications. The matrix covers a total of nine tactics and 39 techniques utilized by threat actors to compromise containers [56].
- **MITRE FiGHT:** The MITRE 5G Hierarchy of Threats (FiGHT) framework is a collection of TTPs known to target 5G networks. The TTPs are based on research, publicly available information and real-world attacks. The framework is also based on MITRE ATT&CK and aids in understanding the threat landscape of 5G systems, as well as how adversaries approach and exploit these systems [57].
- **Cloud Matrix:** The Cloud Matrix is part of the MITRE ATT&CK framework and covers the TTPs leveraged against cloud-native environments. As the deployment models of O-RAN are based on differing cloud deployments (such as public-, hybrid- and private cloud), this matrix provides a thorough understanding of how these threats apply in these circumstances [58].

- **OWASP Top 10** The OWASP foundation maintains prominent collections of threats that pose the highest risk to a particular application or ecosystem. The OWASP Kubernetes Top 10 and Docker Top 10 further aid in identifying the highest risk threats in the containerized and Kubernetes orchestrated O-RAN I-Release deployment [59] [60] [61].

5.5.2 Related work

This section discusses related work conducted in the area of threat analysis in Open RAN. The work is based on research papers, WG11 and publications by Trend Micro. The WG11 of the O-RAN Alliance has conducted research in the area of threat modeling and analyzing threats in Open RAN. The working group has identified numerous threats that span all assets part of the O-RAN ecosystem. The most important threat categories relevant to this research are threats affecting the O-Cloud. These categories are generic threats (T-GEN), threats affecting VMs and containers (T-VM-C), threats regarding VM and container images and threats concerning the virtualization layer (T-VL) such as the hypervisor and the container runtime. Furthermore, threats concerning O-Cloud administration (T-ADMIN), the near-RT-RIC (T-NEAR-RT), non-RT-RIC (T-NON-RT) and open source software (T-OPENSRC) are deemed relevant as well. However, the threats are analyzed in a broader context and lack meticulous techniques that make these threats actionable for the purpose of adversary simulation [47].
This lack of threat mapping on to specific adversarial techniques is covered in the paper "Toward Securing the 6G Transition" by Klement et. al [44]. The research outlines a method for mapping O-RAN threat IDs onto MITRE ATT&CK techniques, including the calculation of severity metrics based on the CVSS framework. The metrics include impact, exploitability and a base score. Furthermore, the overall severity, likelihood and risk of the threats is assessed. Furthermore, a second paper by Klement et al. named "Securing the Open RAN Infrastructure, discusses the practical vulnerability assessment of the O-RAN SC implementation in the context of Kubernetes [62]. The paper discusses assessment techniques, such as SAST, deployment auditing, penetration testing and runtime security. An security evaluation of the near-RT-RIC illustrates that 792 potential vulnerabilities affect the O-RAN SC's implementation, with 13 of the critical vulnerabilities being exploitable in the deployment.
The company Trend Micro has conducted research specifically addressing the security of xApps in the O-RAN ecosystem. Their research has identified multiple vulnerabilities in the O-RAN SC G- and I-Release which are listed in the subsequent section

on known vulnerabilities. The discovered vulnerabilities can cause the disruption of E2 services. Through exploitation, a rogue xApp is able to eavesdrop and intercept E2 traffic, or cause complete unavailability of E2 services. As a consequence, the concept of a rogue xApp has been introduced into the MITRE FiGHT knowledge base [63] [64].

5.5.3 ORAN SC Vulnerabilities

There exist multiple known vulnerabilities in the O-RAN SC I-Release implementation, depicted in the following table 5.2. A full description of these vulnerabilities is listed in Appendix A using data from the CVE public database [65].

CVE	Vendors	Products	Updated
CVE-2024-34473	O-RAN-SC	I-Release	2024-05-06
CVE-2024-34044	O-RAN-SC	I-Release	2024-04-30
CVE-2024-34047	O-RAN-SC	I-Release	2024-04-30
CVE-2024-34045	O-RAN-SC	I-Release	2024-04-30
CVE-2024-34048	O-RAN-SC	I-Release	2024-04-30
CVE-2024-34043	O-RAN-SC	I-Release	2024-04-30
CVE-2024-34046	O-RAN-SC	I-Release	2024-04-30

Table 5.2: Known CVEs affecting the O-RAN SC I-Release implementation [65]

The older O-RAN SC G-Release has also been affected by multiple security vulnerabilities, which are scored using the CVSS framework, as shown in table 5.3.

CVE	Vendors	Products	Updated	CVSS v3
CVE-2023-42358	O-RAN-SC	Ric-plt-e2mgr	2024-01-10	7.7 HIGH
CVE-2023-41627	O-RAN-SC	Ric Message Router	2023-12-14	7.5 HIGH
CVE-2023-40998	O-RAN-SC	Ric Message Router	2023-12-14	7.5 HIGH
CVE-2023-40997	O-RAN-SC	Ric Message Router	2023-12-14	7.5 HIGH
CVE-2023-41628	O-RAN-SC	E2	2023-12-10	7.5 HIGH

Table 5.3: Known CVEs affecting the O-RAN SC G-Release implementation [65]

However, there exist no public PoCs for these vulnerabilities at the time of writing this report, and therefore the impact of these vulnerabilities cannot be reproduced and independently assessed. Furthermore, without available PoCs, attack simulation addressing these vulnerabilities is limited.

5.5.4 Practical Security Assessment

Assessing threats practically in the laboratory environment with the goal of utilizing discovered vulnerabilities and weaknesses in adversary simulation is a complex task.

However, in the realms of Kubernetes and containerized applications, there exist several tools to scan and assess the security posture of the O-RAN SC I-Release deployment. The tools kube-bench and Trivy by Aqua Security as well as kubescape maintained by the CNCF, are excellent tools to identify misconfigurations and vulnerabilities in Kubernetes and their associated workloads. The integration with the Center for Internet Security (CIS) benchmark, compliance frameworks and vulnerability scans of images and filesystem allow for a broad and comprehensive threat analysis of the O-RAN SC I-Release deployment [66] [67] [68] [69].

6 Threat Model Implementation

This chapter presents the implementation of a threat modeling constructs for a Kubernetes-based Open RAN implementation based on the O-RAN SC I-Release implementation, for the purpose of adversary simulation.

6.1 Scope

The scope of the threat model centers around the O-RAN infrastructure in terms of the O-Cloud. The deployment model of a private cloud using the container orchestrator Kubernetes is considered. The primary components evaluated, are the Kubernetes orchestrated near-RT-RIC and the non-RT-RIC, container runtime, images and other software components integrated into the deployment. Out of scope are the SMO and disaggregated gNB components including exposed interfaces, as well as hardware components and User Equipments (UEs).

6.2 Adversary Objectives

The definition of adversary objectives is required in order to model potential attack paths that aim at achieving of these goals. The list of high-level objectives is loosely based on the MITRE ATT&CK framework, however considered in the context O-RAN deployment. The objectives are defined as follows:

High-level Adversary Objectives

- **Disruption of Service:** This objective aims at disrupting the operations of the O-RAN deployment, by targeting the near-RT-RIC and non-RT-RIC components. The objective may be achieved by targeting single services or the entire deployment, as well as targeting the gNB components, through the near-RT-RIC.
- **Data Theft:** The objective of data theft is achieved by targeting the near-RT-RIC and non-RT-RIC components or data in transit, in order to exfiltrate

confidential data. The data may include RAN configuration data, Key Performance Indicators (KPIs), credentials such as keys, certificates or service account tokens, as well as network data and UE data.

- **Data Destruction:** This objective entails the destruction or manipulation of data with the goal of rendering the O-RAN deployment inoperable.
- **Reputation Damage:** The objective of reputation damage is achieved through impacting the availability, integrity or confidentiality of the O-RAN deployment. Therefore, the objective may be attained by fulfilling one or multiple of the goals defined above.
- **Espionage:** The objective of espionage aims at gaining knowledge of the RAN operation, in order to prepare further attacks, damage the reputation of the operator or to gain an economic advantage.

6.3 Initial Access Assumption

The goal of the threat model is to theoretically structure adversarial attack patterns by threat actors with a varying degree of access to the O-RAN infrastructure. This follows the assume-breach model discussed in section 3.3.1. Therefore, the means of accomplishing initial access are subordinate to the attack patterns employed upon gaining initial access.

6.3.1 Entry Points

This section outlines the defined entry points presuming the assumed-breach scenario. The initial access assumptions align with the threat actors defined in the previous chapter. These threat actors are either internal to RAN operations, thus have some degree of access to the infrastructure, including networks, hosts or code and image repositories. Furthermore, the xApp/rApp developer is considered, that has the ability to indirectly deploy and execute code in the near-RT-RIC and non-RT-RIC components.

Entry Points

- **Compromised Credentials:** A straightforward yet plausible method of gaining access to the O-RAN infrastructure is through the malicious use of valid, potentially overpermissive or compromised credentials. This may include stolen or accidentally disclosed credentials, such as developer or operator credentials in the form of keys, certificates or service account tokens. Plausibility of this

method is reinforced by the Cloud Security Alliance (CSA) report, that analyzed the top threats to cloud computing, and identified insufficient credential, access and key management as the number one security issue in cloud environments [70].

- **Compromised Container Image:** The use of supply chain attacks, such as backdooring a container image utilized in the target infrastructure is another plausible method of gaining initial access and considered in this threat model.
- **Vulnerable publicly facing API/application:** The existence of a publicly exposed API or service allows adversaries to enumerate and potentially exploit the given threat vector. This is especially critical in public cloud deployments that may lack dedicated private access to all the deployed and operated components.
- **xApp/rApp:** The xApp and rApp applications classify as unique supply chain attacks to O-RAN deployments. They can be considered as potential entry points, as they are developed externally, and deployed in the near-RT-RIC and non-RT-RIC respectively. There exist few standardized security prerequisites in the onboarding and development process of such applications, such as least-privilege access. Therefore, these components enable adversaries to gain an initial foothold into the O-RAN infrastructure or cause extensive damage, as discussed in section 5.5.2.

6.4 Tooling Review

The process of threat modeling is supported by multiple tools that facilitate the creation of threat models. This includes tooling to depict the architecture, DFDs, as well as threats and attack trees. The following tools are considered for the threat model implementation:

Threat Model Tooling

- **Threat Dragon:** Threat Dragon is an open-source threat modeling tool maintained by the OWASP foundation. The tool assists in the creation of threat model diagrams employing values and principles of the Threat Modeling Manifesto discussed in 4.1.1. Furthermore, the tool supports assessing individual threats according to multiple threat modeling frameworks, such as STRIDE. The main use-case of Threat Dragon in this research is the creation of DFDs. The tools comes in two variants, a web-based application and a desktop application [71].

- **Threagile:** Threagile is an open-source agile threat modeling toolkit that supports the declarative definition of system architectures and assets using the YAML Ain't Markup Language (YAML) format. The use of YAML for threat model definitions, aids in collaboration and version control as well as in testing and verification of the security posture of the assessed system. This allows for the automated generation of threat models and integration of threat modeling into the SDLC. Moreover, Threagile features the identification of potential risks and threats, supplying hardening recommendations and generating reports in various formats. The toolkit offers multiple other features for threat assessment, however, these are out of scope for this report. The primary application of Threagile in this research, is the modeling of attack trees through the integration with the tool AttackTree, to derive potential attack patterns and scenarios that can be utilized for adversary simulation purposes [72] [73].
- **AttackTree:** The Software as a Service (SaaS) tool AttackTree is a web-based application that supports the creation of comprehensive attack trees. The tool is integrated with Threagile, allowing for the creation of extensive threat models that feature threat- and risk assessment, mitigation and control mechanisms and reporting. The modeling tool features the definition of threat actors, controls and attacks in a graphical format. Furthermore, attacks can be classified according to multiple metrics, including the STRIDE framework, attack complexity and likelihood as well as impact and associated risk. The primary use-case of AttackTree in this research is the creation of attack trees that depict potential attack paths and scenarios that adversaries may employ to achieve their objectives [74].

6.4.1 System Modeling

This section outlines the system modeling of the O-RAN I-Release deployment, as deployed in the laboratory environment. The system modeling is based on the concept of DFD, using the modeling tool Threat Dragon by the OWASP foundation [71]. The DFD is a graphical representation of the system architecture and actors, depicting the interaction of components with a defined degree of detail. A high-level view of the Open RAN deployment, a level 1 DFD (DFD-1) is illustrated in the figure below.

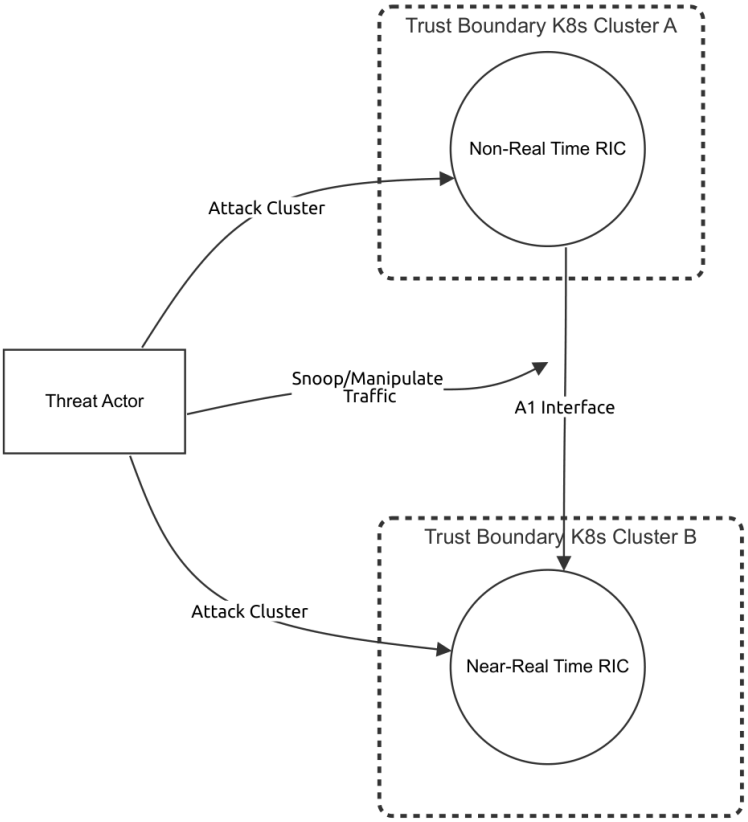

Figure 6.1: DFD-1 of the O-RAN RICs

More detailed DFDs are depicted in Appendix B and Appendix C, illustrating the internal structure of the Kubernetes orchestrated RICs. The DFD models are based the threat models published by the the CNCF Financial User Group [75]. The created models depict the interaction of the individual Kubernetes components, trust boundaries, as well as the individual pods and their segregation into different namespaces. These models aid in identifying potential threat vectors and attack paths, by modeling system components and their interactions with an extended level of detail. Depending on the type of threats considered and the structure of the deployment, varying levels of DFDs are created, to model and represent the systems appropriately for analysis purposes.

6.5 Attack Trees

This sections highlights multiple subtrees and attack paths of the developed attack tree, to visualize attacks on the O-RAN I-Release deployment. The modeled attack tree is based on the threat assessment, system modeling, entry points and defined

adversary objectives. The entire attack tree is not depicted in this report, due to its complexity and size.

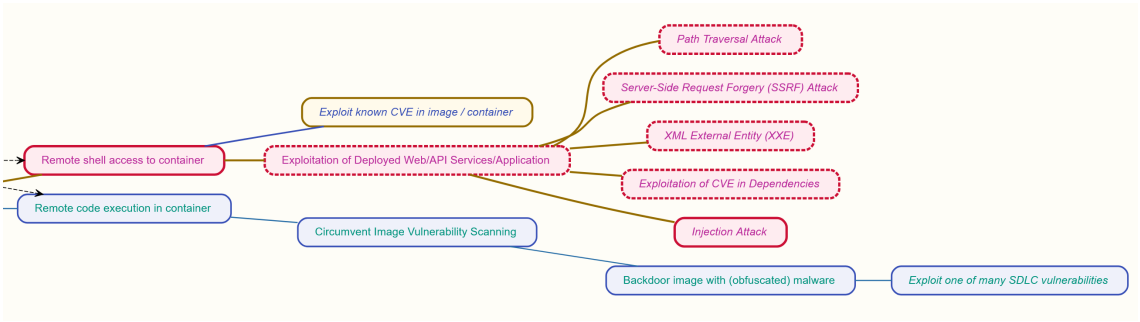

Figure 6.2: Attack Tree: Container Access

The first subtree 6.2 of the model depicts the means of gaining RCE or shell access to a container. The means of achieving this, include uploading a malicious container image to an image repository that hosts valid images for the deployment. This is achieved by exploiting one of the various weaknesses or vulnerabilities in the SDLC. The uploaded image then passes vulnerability scans, through obfuscation of the backdoor and through missing security gates and comprehensive scanning of the image. The second path in the subtree comprises the exploitation of an exposed API or service, that allows for the execution of arbitrary code or commands and spawning of a shell inside the container. A third option of gaining RCE is through the exploitation of known vulnerability in the valid container image, that was deployed without mitigating this vulnerability.

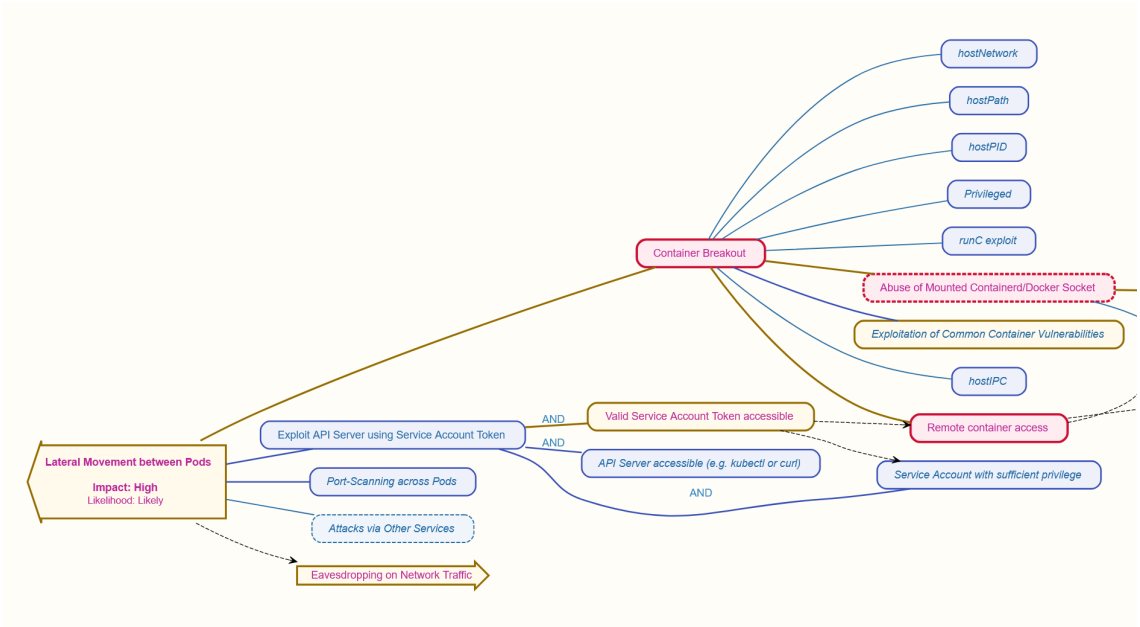

Figure 6.3: Attack Tree: Lateral Movement

The second subtree 6.3 of the model depicts the lateral movement tactics of an adversary upon gaining container access. The first path involves escaping a container. This is achieved through various means, such as a privileged container, or access to particular host namespace, that enables the interaction with a host system and facilitate an escape. Another path in the subtree depicts the discovery of valid credentials in the form of keys or service accounts tokens, that enable access to other services or systems within the deployment. Furthermore, an adversary may conduct reconnaissance on the network, to identify further services and systems that can be targeted for lateral movement. This can be achieved through port scanning or eavesdropping on the network.

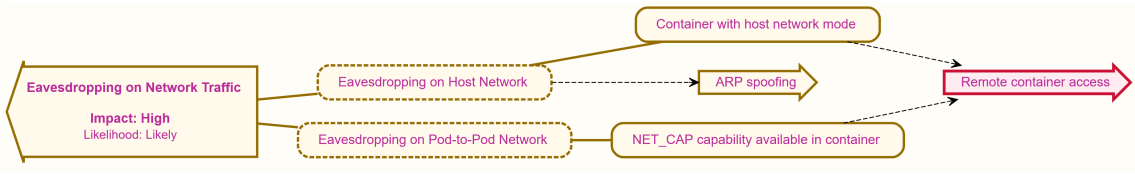

Figure 6.4: Attack Tree: Eavesdropping

The next subtree 6.4 of the model depicts techniques employed by adversaries to eavesdrop on the network. Upon gaining access to a container, the threat actor enumerates the capabilities and privileges of the container. If the container has the NET_CAP capability, the attacker will be able to eavesdrop traffic on Pod-to-Pod networks. This capability is not uncommon, as sidecar containers in service meshes require this capability for various network-related functions, or containers implementing functionality of the Container Network Interface (CNI). The attack may also be able to eavesdrop on the host network, if the container runs in privileged mode or runs in the host network mode. This enables an attacker to eavesdrop on all interfaces that the host is connected to. The last path in the subtree depicts eavesdropping on the host network using Address Resolution Protocol (ARP) spoofing. This technique enables privileged containers to redirect all traffic via the container, allowing the attacker not only to eavesdrop on the network, but also to manipulate and inject traffic.

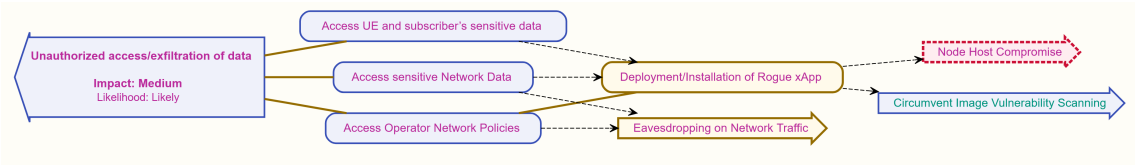

Figure 6.5: Attack Tree: Data Access and Exfiltration

The last subtree depicts the paths for unauthorized data access and exfiltration

tactics of an adversary. The most straightforward means of achieving this is through a node host compromise. This may be possible through the exploitation of a kernel vulnerability, but also through a container escape as mentioned previously. Furthermore, with the use of xApps and rApps in the RIC components, an adversary may gain access to sensitive data through the deployment of a rogue xApp/rApp. This goal may be attained by uploading a malicious xApp/rApp to the image store and circumventing the security scan or by exploiting a vulnerability in an xApp/rApp that was developed and deployed without appropriate security measures. Either scenario enables the adversary to access sensitive data, such as potential UE or subscriber data, network data or network policies and configuration data [76].

7 Attack Scenarios and Strategy

This chapter provides an overview of attack scenarios and strategies developed for adversary simulation in the context of this research project.

7.1 Attack Scenarios

This section outlines a series of attack scenarios that can be used to simulate adversarial activities in the context of the O-RAN SC I-Release implementation and its Kubernetes environment. The scenarios are based on attack chains to simulate realistic and complete attack patterns, potentially employed by threat actors targeting Open RAN implementations and the underlying infrastructure. The first set of scenarios center around threats identified by the WG11 of the O-RAN Alliance, while the second set of scenarios are based on threats identified in the Microsoft Threat Matrix for Kubernetes discussed in section 5.5.1.

7.1.1 WG11 based Threat Scenarios

Attack Scenario: T-GEN-02

Malicious Use of Exposed Kubernetes API

Objectives: Exfiltrate sensitive data by abusing exposed Kubernetes API

1. **Initial Access**: Access Kubernetes API server using valid credentials.
2. **Execution**: Uses kubectl to list and describe sensitive resources within the cluster.
3. **Privilege Escalation**: Creates new roles and bindings using kubectl to gain additional privileges.
4. **Discovery**: Lists secrets, config maps, and other sensitive data using kubectl.
5. **Collection**: Reads and copies sensitive data from discovery process using kubectl.
6. **Exfiltration**: Transfers the copied data to an external server.

Attack Scenario: T-VM-C-01

Abuse of Privileged Calico Pod for Data Exfiltration from Near-RT RIC

Objectives: Exfiltrate sensitive data by abusing a privileged Calico pod

1. **Initial Access**: Gains access to the privileged Calico pod within the Kubernetes cluster.
2. **Execution**: Uses the Calico pod's elevated privileges to execute commands on the host system.
3. **Privilege Escalation**: Escalates privileges from the Calico pod to gain root access on the host system running Near-RT RIC.
4. **Discovery**: Scans the host system to identify sensitive data and critical files related to Near-RT RIC.
5. **Collection**: Accesses and copies sensitive data from the host system.
6. **Exfiltration**: Transfers the copied data to an external server controlled by the adversary.

Attack Scenario: T-IMG-01

Container Image Tampering

Objectives: Exfiltrate sensitive data by tampering with container images

1. **Initial Access**: Gains access to the container image repository using valid credentials.
2. **Execution**: Uploads a tampered container image with a backdoor to the repository.
3. **Deployment**: The tampered image is deployed within the Kubernetes cluster, targeting Near-RT RIC components.
4. **Privilege Escalation**: The malicious code exploits the environment to gain elevated privileges on the host system running Near-RT RIC.
5. **Discovery**: Scans the host system to find sensitive data related to Near-RT RIC.
6. **Collection**: Copies sensitive data from the host system.
7. **Exfiltration**: Transfers the copied data to an external server.

Attack Scenario: T-IMG-03

Secrets Disclosure in Container Image

Objectives: Exfiltrate sensitive data by exploiting secrets disclosure in a container image

1. **Initial Access**: Gains access to the container image repository using valid credentials.
2. **Execution**: Identifies a container image containing hardcoded secrets or credentials.
3. **Deployment**: Compromised image deployed within the Kubernetes cluster, targeting Near-RT RIC components.
4. **Privilege Escalation**: Uses the disclosed secrets to gain elevated privileges within the cluster.
5. **Discovery**: Scans the cluster to find additional sensitive data related to Near-RT RIC.
6. **Collection**: Copies the sensitive data from the cluster.
7. **Exfiltration**: Transfers the copied data to an external server.

7.1.2 Threat Matrix for Kubernetes based Scenarios

Attack Scenario: Threat Matrix Kubernetes 01

Exfiltrate Secrets from Cluster

Requirements: Attacker has obtained access to the cluster

1. **Discovery**:
   - The attacker lists privileged pods to select target.
2. **Credential Access**:
   - The attacker retrieves a service account token by displaying the secrets.
3. **Privilege Escalation**:
   a) Role Bindings bind Service Account to a higher privilege role (cluster-admin).
   b) The attacker gets a shell on one of the listed privileged containers.
4. **Persistence**:
   - The attacker creates Daemon Set that functions as backdoor container.
5. **Defense Evasion**:

a) Attacker names the backdoor pod as they were created by the existing controllers.
b) Attackers clear container logs and Kubernetes events.

6. **Impact**: The attacker obtains all secrets from the cluster and exfiltrates them.

Attack Scenario: Threat Matrix Kubernetes 02

Exfiltrate Data from Host Filesystem

Requirements: Attacker has gained access to the cluster and has rights to deploy a privileged pod.

1. **Privilege Escalation**:
   a) The attacker deploys a privileged pod and obtains shell on the container.
   b) The attacker uses the privileged container to mount the host's file system.
   c) Once mounted, the attacker explores directories and files on the host machine.
2. **Impact**:
   - The attack exfiltrates the files in /etc/kubernetes.
3. **Persistence**:
   a) Role Bindings bind Service Account to a higher privilege role (cluster-admin).
   b) The attacker gets a shell on one of the listed privileged containers.
4. **Persistence**:
   - The attacker creates a CronJob that adds the attack SSH to the authorized_keys.

7.2 Execution Strategy

This section discusses the execution strategy for the attack scenarios outlined above. The strategy involves the use of suitable tools and implementation of techniques to simulate adversarial activities in the O-RAN SC I-Release implementation.

7.2.1 Attack Tooling and Automation Frameworks

In the course of the research project FORAN a number of tools have been identified that can be used to simulate adversarial activities. Futhermore, a framework named

"Clusterforce" has been developed to automate the execution of atomic tests and attack scenarios. The framework also entails logging and analysis capabilities to evaluate the effectiveness of the adversarial simulations.
The following figure depicts the deployment of the tools and the "Clusterforce" framework in the testing environment.

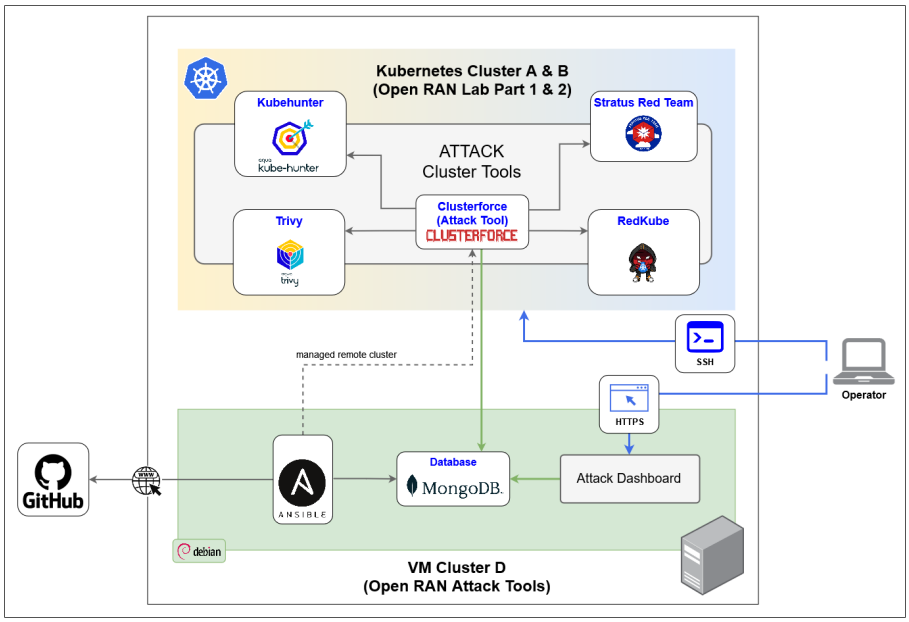

Figure 7.1: Attack Simulation Environment with Clusterforce [1]

The deployment of the framework and according tools is automated using Ansible playbooks, in order to facilitate in the setup and removal process of the associated components. However, one of the downsides of the current implementation is the relatively large set of dependencies required in the setup process.

7.2.2 CALDERA

The MITRE Corporation (MITRE) Cooperation has developed a comprehensive tool for security testing and adversary emulation named "CALDERA". This tool allows for the definition threat agents, extensive adversary profiles and the creation of attack scenarios based on custom or pre-defined TTPs [77].
The execution of scenarios is less noisy compared to Clusterforce, since a single custom reverse shell serves as the entry and starting point for the execution of the entire pre-defined scenario. This avoids the need to installing additional software and dependencies on the target system. Furthermore, the tool features extensive analytics and reporting, C2 communication capabilities and a modular design that leverages the large plugin library offered with CALDERA [78].

8 Discussion

This chapter reflects on the findings and results of the research project covered in this report.

8.1 Assessment of Objectives

The defined objectives of conducting a comprehensive technical review and threat assessment of an Open RAN deployment based on Kubernetes were achieved. This analysis yielded various potential threats and vulnerabilities that are relevant to O-RAN SC I-Release deployment. Furthermore, the analysis was used to establish a threat model, primarily in the form of attack trees, that depict various attack scenarios and techniques that could be executed against the Open RAN deployment. Furthermore, the definition of concrete attack scenarios and means of executing the scenarios based on the recommend tools was outlined.

8.2 Limitations and Challenges

The presented findings are subject to several limitations, that are discussed in the following section. Furthermore, some of the encountered challenges that were faced in conducting the research are discussed.
One of the main limitations of this research project are the lack of real-world data of incidents in Open RAN deployments. Therefore, potential attack scenarios and techniques can only be anticipated, however, may not be fully authentic. Furthermore, the reference implementation supplied by the O-RAN SC is in active development, with a focus on adding features and functionality, instead of supplying a secure and hardened deployment. Therefore, some of the identified threat vectors are likely to be mitigated in operational deployments or do not apply due to the redesign of the reference implementation in future updates and releases. Moreover, the reference implementation may not be fully representative of implementations of Open RAN, as vendors may adapt or develop their own implementations. Furthermore, the lack of disaggregated gNB components and UEs in the laboratory setup, implies deficient

interaction and traffic generation in the deployed environment, that would normally occur in an operational Open RAN deployment.
A key challenge faced in this research concerns the lack of publicized and implemented PoC attacks against Open RAN deployments. This increases the complexity in recommending and rating the impact and applicability of adversarial techniques and scenarios, as implementing and validating some of the techniques requires the development of exploits.

8.3 Practical Implications

The results of this research project form the basis of advancing adversarial simulation in the context of the FORAN project. The developed attack scenarios are to be used in validating and enhancing the forensic and incident response capabilities developed throughout the FORAN project. Furthermore, the insights gained from this research, can be applied to improve the attack strategy and tooling arsenal of the FORAN project, and serve as the foundation for developing and testing more applicable attack scenarios.

8.4 Future Work

The outcomes of this research project can be applied in several ways as well as extended and improved upon. At first, the developed attack scenarios can be implemented and refined to be used in the FORAN project. Furthermore, the threat assessment can be used to develop and implement more atomic techniques tailored to specific Open RAN vulnerabilities. In addition, the development of rogue xApps can be considered, and used to simulate malicious behavior in the Open RAN deployment. Moreover, the extension of the laboratory environment to include disaggregated gNB components and UEs can be considered, to increase the authenticity of the deployed environment.

9 Conclusion

This research project portrayed a structured approach to developing an adversary simulation strategy based on a Kubernetes-based Open RAN deployment. The report covered the techniques and methodologies for assessing and identifying potential threats and using the result to establish a threat model based on attack trees. These attack trees were used to define concrete attack scenarios and strategies that could be implemented and executed in an Open RAN deployment, for the purpose of threat simulation. Furthermore, the methodology and tools used for conducting the adversary simulation were outlined, and the results of the research were discussed. The research project was conducted in the context of the FORAN project, and the results will be used to enhance the attack simulation strategy with the goal of advancing the forensic and incident response capabilities in the Open RAN domain.

Appendix A

CVE	Vendors	Products	Updated	Description
CVE-2024-34473	O-RAN-SC	I-Release	2024-05-06	An issue was discovered in appmgr in O-RAN Near-RT RIC I-Release. An attacker could register an unintended RMR message type during xApp registration to disrupt other service components.
CVE-2024-34044	O-RAN-SC	I-Release	2024-04-30	The O-RAN E2T I-Release buildPrometheusList function can have a NULL pointer dereference because peerInfo can be NULL.
CVE-2024-34047	O-RAN-SC	I-Release	2024-04-30	O-RAN RIC I-Release e2mgr lacks array size checks in RicServiceUpdateHandler.
CVE-2024-34045	O-RAN-SC	I-Release	2024-04-30	The O-RAN E2T I-Release Prometheus metric Increment function can crash in sctpThread.cpp.
CVE-2024-34048	O-RAN-SC	I-Release	2024-04-30	O-RAN RIC I-Release e2mgr lacks array size checks in E2nodeConfigUpdateNotificationHandler.
CVE-2024-34043	O-RAN-SC	I-Release	2024-04-30	O-RAN RICAPP kpimon-go I-Release has a segmentation violation via a certain E2AP-PDU message.
CVE-2024-34046	O-RAN-SC	I-Release	2024-04-30	The O-RAN E2T I-Release Prometheus metric Increment function can crash in sctpThread.cpp.

Table 1: Public CVEs affecting the O-RAN Software Community I-Release implementation [65]

Appendix B

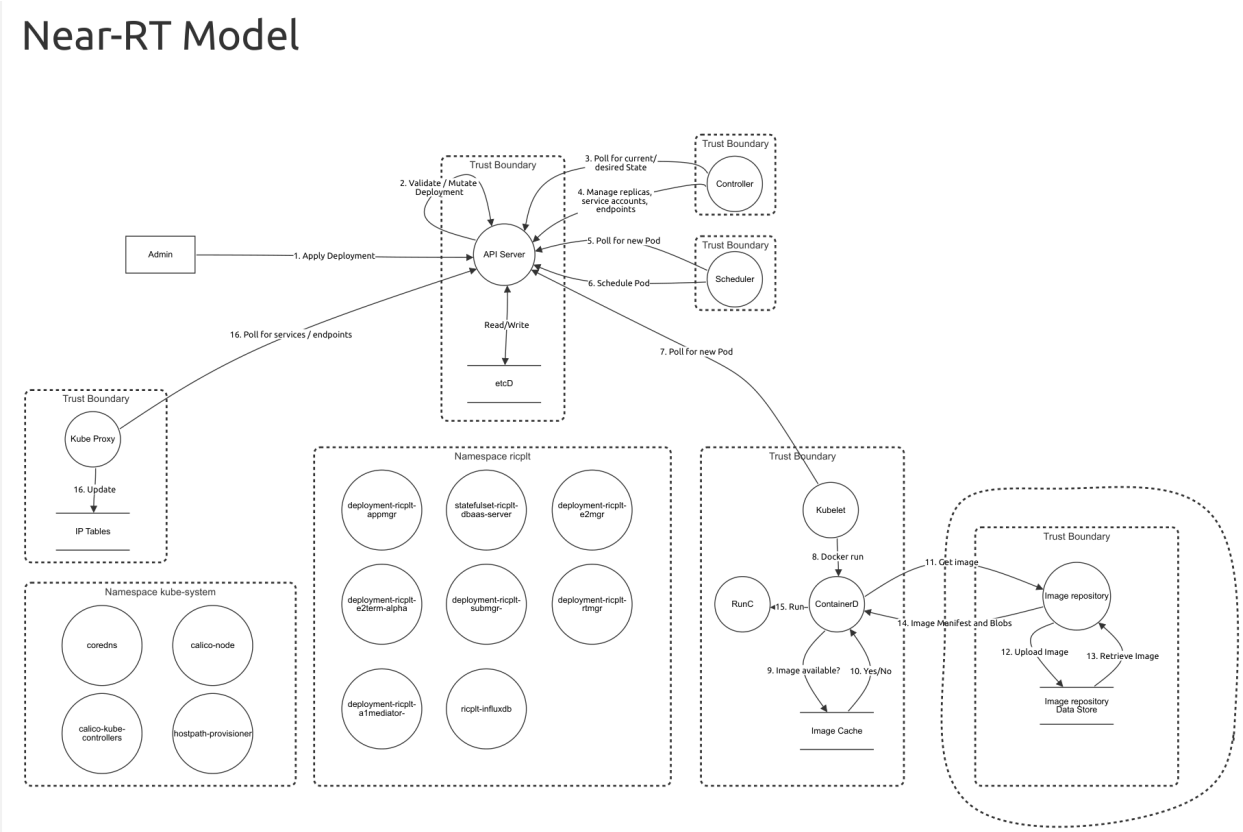

Figure 1: near-RT-RIC Level 2 Data Flow Diagram

Appendix C

Non-RT Model

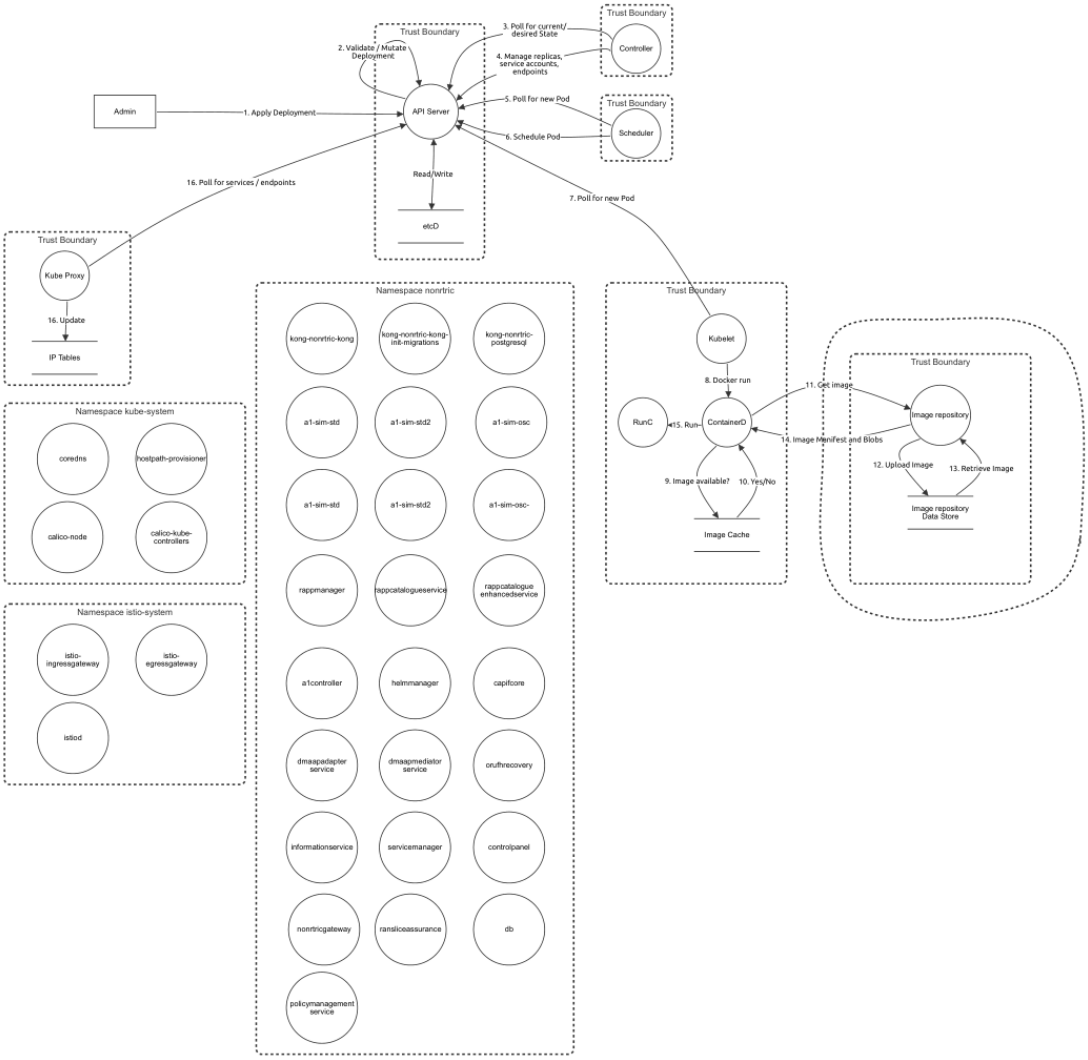

Figure 2: non-RT-RIC Level 2 Data Flow Diagram

Bibliography

[1] *5G-FORAN*, https://www.5g-foran.com/. (visited on 27 May 2024).

[2] *TH Köln*, https://www.th-koeln.de/startseite_16.php. (visited on 27 May 2024).

[3] *Release-Notes — oran master documentation*, https://docs.o-ran-sc.org/en/latest/release-notes.html. (visited on 29 May 2024).

[4] *2023-12-19 Release I - RIC Platform - Confluence*, https://wiki.o-ran-sc.org/display/RICP/2023-12-19+Release+I. (visited on 29 May 2024).

[5] *Installation Guides — oran master documentation*, https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/installation-guides.html. (visited on 29 May 2024).

[6] *O-RAN ALLIANCE e.V*, https://www.o-ran.org/. (visited on 31 May 2024).

[7] *TIP*, https://telecominfraproject.com/openran/. (visited on 31 May 2024).

[8] *O-RAN Software Community*, https://o-ran-sc.org/. (visited on 31 May 2024).

[9] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in Cloud Computing: State of the Art and Research Challenges," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 430–447, March 2018, ISSN: 1939-1374. DOI: 10.1109/TSC.2017.2711009. (visited on 03 Jun. 2024).

[10] *Kubernetes Components*, https://kubernetes.io/docs/concepts/overview/components/. (visited on 03 Jun. 2024).

[11] *Kubernetes Release Cycle*, https://kubernetes.io/releases/release/. (visited on 03 Jun. 2024).

[12] *Patch Releases*, https://kubernetes.io/releases/patch-releases/, April 2025. (visited on 03 Jun. 2024).

[13] Catalin, *Adversary Emulation vs Simulation | Redpoint Cyber*, https://www.redpointcyber.com/emu vs-simulation/, January 2023. (visited on 04 Jun. 2024).

[14] "Adversary simulation: Put your incident response programs to the test,"

[15] *Penetration Testing | U.S. Department of the Interior*, https://www.doi.gov/ocio/customers/p testing, Site Page, Jun. 2015. (visited on 05 Jun. 2024).

[16] *Definitions | Red Team Development and Operations*, https://redteam.guide/docs/definitions. (visited on 05 Jun. 2024).

[17] M. Muckin and S. C. Fitch, "A Threat-Driven Approach to Cyber Security," 2019.

[18] *Practical Threat Detection Engineering: A hands-on guide to planning, developing, and validating detection capabilities | Packt Publishing books | IEEE Xplore*, https://ieeexplore.ieee.org/document/10251369/metrics#metrics. (visited on 05 Jun. 2024).

[19] *Cyber Kill Chain® | Lockheed Martin*, https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. (visited on 05 Jun. 2024).

[20] *Cyber Kill Chain®*, https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html. (visited on 05 Jun. 2024).

[21] *Cyber Attack Kill Chain - Perchè è importante conoscerla*, https://f3rm1.cloud/learning/cyber-attack-kill-chain—perch——importante-conoscerla. (visited on 05 Jun. 2024).

[22] P. Pols, *The Unified Kill Chain*, http://www.unifiedkillchain.com. (visited on 05 Jun. 2024).

[23] Davidjbianco, *Enterprise Detection & Response: The Pyramid of Pain*, March 2013. (visited on 06 Jun. 2024).

[24] *Pyramid of Pain*, https://www.attackiq.com/glossary/pyramid-of-pain/. (visited on 06 Jun. 2024).

[25] *MITRE ATT&CK®*, https://attack.mitre.org/. (visited on 09 Jun. 2024).

[26] *Our Impact | MITRE*, https://www.mitre.org/our-impact. (visited on 09 Jun. 2024).

[27] M. Nicholls, *What is Purple Teaming? How Can it Strengthen Security?* https://www.redscan.com/news/purple-teaming-can-strengthen-cyber-security/, November 2023. (visited on 06 Jun. 2024).

[28] *Red Team VS Blue Team: What's the Difference? - CrowdStrike*, https://www.crowdstrike.com 101/red-team-vs-blue-team/. (visited on 06 Jun. 2024).

[29] *Was ist ein Purple Team? | CrowdStrike*, https://www.crowdstrike.de/cybersecurity-101/purple-teaming/. (visited on 06 Jun. 2024).

[30] K Kent, S Chevalier, T Grance, and H Dang, "Guide to integrating forensic techniques into incident response," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-86, 2006, NIST SP 800–86. DOI: 10.6028/NIST.SP.800-86. (visited on 06 Jun. 2024).

[31] P. Cichonski, T. Millar, T. Grance, and K. Scarfone, "Computer Security Incident Handling Guide : Recommendations of the National Institute of Standards and Technology," National Institute of Standards and Technology, Tech. Rep. NIST SP 800-61r2, August 2012, NIST SP 800–61r2. DOI: 10.6028/NIST.SP.800-61r2. (visited on 06 Jun. 2024).

[32] *Threat Modeling - OWASP Cheat Sheet Series*, https://cheatsheetseries.owasp.org/cheatsheets/Thre Modeling_Cheat_Sheet.html. (visited on 09 Jun. 2024).

[33] *Threat Modeling Process | OWASP Foundation*, https://owasp.org/www-community/Threat_-Modeling_Process. (visited on 09 Jun. 2024).

[34] *Threat Modeling Manifesto*, https://www.threatmodelingmanifesto.org/. (visited on 09 Jun. 2024).

[35] *Threat Modeling | OWASP Foundation*, https://owasp.org/www-community/Threat_-Modeling. (visited on 07 Jun. 2024).

[36] *OWASP Threat Dragon | OWASP Foundation*, https://owasp.org/www-project-threat-dragon/. (visited on 09 Jun. 2024).

[37] Archiveddocs, *The STRIDE Threat Model*, https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20), November 2009. (visited on 09 Jun. 2024).

[38] *Academic: Attack Trees - Schneier on Security*, https://www.schneier.com/academic/archives/1999/ trees.html. (visited on 09 Jun. 2024).

[39] T. UcedaVelez, "Risk Centric Threat Modeling,"

[40] *Threat Modeling*, https://handbook.gitlab.com/handbook/security/product-security/application-security/threat-modeling/. (visited on 09 Jun. 2024).

[41] A. Shostack, "Experiences Threat Modeling at Microsoft,"

[42] DOMARS, *Threat Modeling for Drivers - Windows drivers*, https://learn.microsoft.com/en-us/windows-hardware/drivers/driversecurity/threat-modeling-for-drivers, August 2023. (visited on 10 Jun. 2024).

[43] *Release-Notes — oran master documentation*, https://docs.o-ran-sc.org/en/latest/release-notes.html. (visited on 10 Jun. 2024).

[44] F. Klement, W. Liu, and S. Katzenbeisser, "Toward Securing the 6G Transition: A Comprehensive Empirical Method to Analyze Threats in O-RAN Environments," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 420–431, February 2024, ISSN: 0733-8716, 1558-0008. DOI: 10.1109/JSAC.2023.3339172. (visited on 31 May 2024).

[45] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023, ISSN: 1553-877X, 2373-745X. DOI: 10.1109/COMST.2023.3239220. (visited on 14 Jun. 2024).

[46] M. Alavirad, U. S. Hashmi, M. Mansour, *et al.*, "O-RAN architecture, interfaces, and standardization: Study and application to user intelligent admission control," *Frontiers in Communications and Networks*, vol. 4, March 2023, ISSN: 2673-530X. DOI: 10.3389/frcmn.2023.1127039. (visited on 14 Jun. 2024).

[47] O-RAN Alliance WG11, *Threat-Modeling.O-R003-v02.00*, https://specifications.o-ran.org/specifications. (visited on 16 Jun. 2024).

[48] "Open RAN Risk Analysis,"

[49] *Overview | CVE*, https://www.cve.org/About/Overview. (visited on 16 Jun. 2024).

[50] *CWE - About CWE*, https://cwe.mitre.org/about/index.html. (visited on 16 Jun. 2024).

[51] *Common Vulnerability Scoring System SIG*, https://www.first.org/cvss/. (visited on 16 Jun. 2024).

[52] *NVD - Vulnerability Metrics*, https://nvd.nist.gov/vuln-metrics. (visited on 16 Jun. 2024).

[53] *CAPEC - About CAPEC*, https://capec.mitre.org/about/index.html. (visited on 16 Jun. 2024).

[54] *About - Threat Matrix for Kubernetes*, https://microsoft.github.io/Threat-Matrix-for-Kubernetes/about/. (visited on 16 Jun. 2024).

[55] *Kubernetes Threat Matrix*, https://kubernetes-threat-matrix.redguard.ch/. (visited on 16 Jun. 2024).

[56] *Matrix - Enterprise | MITRE ATT&CK®*, https://attack.mitre.org/matrices/enterprise/conta (visited on 16 Jun. 2024).

[57] *MITRE | FiGHT™*, https://fight.mitre.org/. (visited on 16 Jun. 2024).

[58] *Matrix - Enterprise | MITRE ATT&CK®*, https://attack.mitre.org/matrices/enterprise/cloud/. (visited on 16 Jun. 2024).

[59] *OWASP Kubernetes Top Ten | OWASP Foundation*, https://owasp.org/www-project-kubernetes-top-ten/. (visited on 16 Jun. 2024).

[60] *OWASP Kubernetes Security Testing Guide | OWASP Foundation*, https://owasp.org/www-project-kubernetes-security-testing-guide/. (visited on 16 Jun. 2024).

[61] *OWASP Docker Top 10 | OWASP Foundation*, https://owasp.org/www-project-docker-top-10/. (visited on 16 Jun. 2024).

[62] F. Klement, A. Brighente, M. Polese, M. Conti, and S. Katzenbeisser, *Securing the Open RAN Infrastructure: Exploring Vulnerabilities in Kubernetes Deployments*, May 2024. arXiv: 2405.01888 [cs]. (visited on 31 May 2024).

[63] *Open RAN: Attack of the xApps - Security News*, https://www.trendmicro.com/vinfo/us/security/ne and-exploits/open-ran-attack-of-the-xapps. (visited on 14 Jun. 2024).

[64] *Opening Critical Infrastructure: The Current State of Open RAN Security*, https://www.trendmicro.com/en_us/research/23/l/the-current-state-of-open-ran-security.html, December 2023. (visited on 17 Jun. 2024).

[65] *Vulnerabilities (CVE) - OpenCVE*, https://www.opencve.io/cve?cvss=&search=o-ran+. (visited on 15 Jun. 2024).

[66] *Kubescape*, https://kubescape.io/. (visited on 17 Jun. 2024).

[67] *Aquasecurity/kube-bench*, Aqua Security, Jun. 2024. (visited on 17 Jun. 2024).

[68] *Aquasecurity/trivy*, Aqua Security, Jun. 2024. (visited on 17 Jun. 2024).

[69] *Overview - Trivy*, https://aquasecurity.github.io/trivy/v0.52/. (visited on 17 Jun. 2024).

[70] *Top Threats to Cloud Computing: Pandemic 11 Deep Dive | CSA*, https://cloudsecurityalliance.org/a threats-to-cloud-computing-pandemic-eleven-deep-dive. (visited on 17 Jun. 2024).

[71] *OWASP Threat Dragon | OWASP Foundation*, https://owasp.org/www-project-threat-dragon/. (visited on 17 Jun. 2024).

[72] *Threagile — Agile Threat Modeling Toolkit*, https://threagile.io/. (visited on 17 Jun. 2024).

[73] C. Schneider, "Threagile: Agile Threat Modeling with Open- Source Tools from within your IDE,"

[74] *AttackTree: Model, Simulate, Defend*, https://attacktree.online. (visited on 17 Jun. 2024).

[75] *Financial-user-group/projects/k8s-threat-model at main · cncf/financial-user-group · GitHub*, https://github.com/cncf/financial-user-group/tree/main/projects/k8s-threat-model. (visited on 17 Jun. 2024).

[76] *Radio control manipulation via rogue xApps | MITRE FiGHT™*, https://fight.mitre.org/techni (visited on 17 Jun. 2024).

[77] *Caldera*, https://caldera.mitre.org/. (visited on 18 Jun. 2024).

[78] *Welcome to MITRE Caldera's documentation! — caldera documentation*, https://caldera.readthedocs.io/en/latest/index.html. (visited on 18 Jun. 2024).