



國立臺北科技大學

National Taipei University of Technology

電資學院外國學生專班(iEECS)

**International Graduate Program in Electrical Engineering
and Computer Science (iEECS)**

Master Thesis

**Enhancing Security in O-RAN through Secure
Software Development Lifecycle Analysis:
A Comprehensive Study**

Researcher: Nattapol Raksasook

Advisor: Shiang-Jiun Chen, Ph.D.

July 2024

National Taipei University of Technology

Thesis Verification Form

We hereby recommend that thesis submitted by Nattapol Raksasook entitled 'Enhancing Security in O-RAN through Secure Software Development Lifecycle Analysis: A Comprehensive Study' be accepted as fulfilling the thesis requirement for the Master Degree of the National Taipei University of Technology.

Thesis committee:

Shiang-Jiun Chen

Yi-Wei Ma

Kuo-Chun Wu

Adviser: Shiang-Jiun Chen

Graduate Chair:

Yang-Lang Chang

Date 07/18/2024 (mm/dd/yyyy)

ABSTRACT

Title: Enhancing Security in O-RAN through Secure Software Development Lifecycle

Analysis: A Comprehensive Study

Pages: 121

School: National Taipei University of Technology

Department: International Graduate Program in Electrical Engineering and Computer Science (iEECS)

Time: July, 2024

Degree: Master

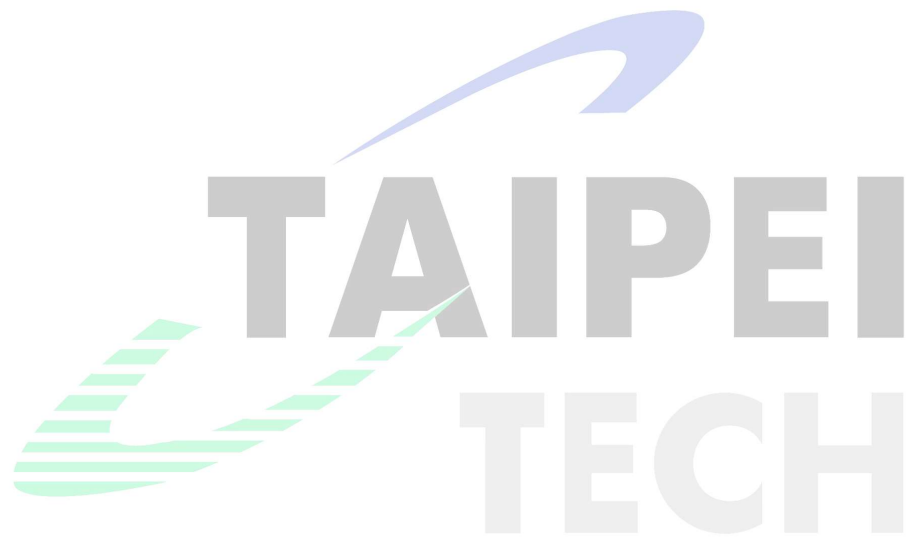
Researcher: Nattapol Raksasook

Advisor: Shiang-Jiun Chen, Ph.D.

Keywords: Open Radio Access Network (O-RAN), Secure Software Development Lifecycle (SSDLC), Software Composition Analysis (SCA), Static Application Security Testing (SAST), Interactive Application Security Testing (IAST), Dynamic Application Security Testing (DAST), Penetration Testing

With the widespread adoption of the Open Radio Access Network (O-RAN) framework, securing its components against potential security breaches has become paramount. This paper delves into the analysis of security vulnerabilities within O-RAN components, analyzing potential risks that might compromise the secrecy, authenticity, and accessibility of vital assets. Utilizing a comprehensive array of security testing tools and methodologies, we perform detailed assessments including vulnerability scanning, penetration testing, and code analysis. By rigorously testing O-RAN components, we aim to pinpoint potential leakage points and security flaws. We then suggest effective remediation strategies and mitigation techniques to address these identified vulnerabilities. By employing sophisticated security assessment tools, we aim to enhance O-RAN security practices, ensuring the resilience and

reliability of O-RAN components against new threats. This research offers valuable insights into the identification, analysis, and resolution of security vulnerabilities within O-RAN, contributing towards fortifying and enhancing the resilience of the O-RAN ecosystem.



Acknowledgments

The successful completion of this project marks a significant milestone in my personal and academic journey. I am profoundly grateful for the contributions of many individuals whose support and guidance have been invaluable. Due to space constraints, I am unable to mention everyone individually, and I apologize for any omissions.

First, I express my deepest gratitude to my esteemed advisor, Professor Shiang-Jiun Chen. Her warm welcome to international students like myself, coupled with her unwavering support and insightful advice on various aspects of living abroad, adapting to a new environment, studying, gaining work experience, and navigating the intricacies of this project, have greatly enhanced the quality of my work and facilitated a smooth progression.

I sincerely appreciate my lab members, particularly Mr. Yu-Xiang Chen, for his expert advice on the installation of both Non-RT RIC and Near-RT RIC in O-RAN, and for his consistent assistance in resolving issues that arose during the installation process.

Lastly, I am deeply thankful to my family for their constant encouragement and support. Their unwavering belief in me has been the driving force behind my academic endeavors.

In conclusion, the success of this research is attributed to the collective efforts of all the individuals mentioned. I am profoundly indebted to everyone, including those not mentioned, for their invaluable assistance and support, which have been instrumental in the success of this project.

Table of Contents

ABSTRACT	i
Acknowledgments	iii
List of Tables	vi
List of Figures	vii
Chapter 1 Introduction	1
Chapter 2 Background.....	6
2.1 O-RAN ALLIANCE	6
2.2 O-RAN Software Community (OSC)	8
2.3 O-RAN Architecture	10
2.4 O-RAN Components	12
2.4.1 SMO (Service Management and Orchestration Framework).....	12
2.4.2 Non-RT RIC (Non Real Time RAN Intelligent Controller).....	15
2.4.3 Near-RT RIC (Near Real Time RAN Intelligent Controller)	19
2.4.4 O-CU (Open Central Unit).....	25
2.4.5 O-DU (Open Distributed Unit)	27
2.4.6 O-RU (Open Radio Unit)	30
2.4.7 O-Cloud.....	32
2.5 O-RAN Interfaces	33
2.5.1 3GPP interfaces	33
2.5.2 O-RAN interfaces.....	35
2.6 SSDLC and Security Testing Methods.....	36
Chapter 3 Implementation.....	39
3.1 Environment Setup.....	39
3.1.1 Non-RT RIC Setup Process.....	41

3.1.2 Near-RT RIC Setup Process	42
3.2 Testing Process	43
Chapter 4 Results Analysis and Demonstration	50
4.1 Non-RT RIC	50
4.1.1 SCA	50
4.1.2 SAST	61
4.1.3 IAST	72
4.1.4 DAST	76
4.1.5 Pentest	81
4.2 Near-RT RIC	85
4.2.1 SCA	85
4.2.2 SAST	86
4.2.3 IAST	91
4.2.4 DAST	92
4.2.5 Pentest	95
4.3 Demonstration	101
4.3.1 OpenVAS: ICMP Timestamp Disclosure	101
4.3.2 Kube-hunter: Kubernetes Version Disclosure	104
Chapter 5 Conclusion and Future Work	109
References	112

List of Tables

Table 1: SMO risks in O-RAN	14
Table 2: Non-RT-RIC risks.....	17
Table 3: Near-RT-RIC risks.....	23
Table 4: Software Requirements	40
Table 5: Hardware Requirements	41
Table 6: Security Testing Methods Across SSDLC.....	44
Table 7: Security Analysis Tools	44
Table 8: CVSS Severity Score Ranges.....	48
Table 9: Severity Description	51
Table 10: OWASP Dependency Check Results and Solutions.....	53
Table 11: Mend.io Results from Non-RT RIC	59
Table 12: Codacy Total Results from Non-RT RIC	62
Table 13: Codacy Results Focus Security Category from Non-RT RIC	62
Table 14: Aikido Total Results from Non-RT RIC.....	65
Table 15: SonarQube Security Results from Non-RT RIC	70
Table 16: OpenVAS Total Results from Non-RT RIC	74
Table 17: Nessus Total Results from Non-RT RIC	77
Table 18: Metasploit with Nmap Integration Results Details and Solutions	82
Table 19: Aikido Total Results from Near-RT RIC.....	87
Table 20: OpenVAS Total Results from Near-RT RIC.....	91
Table 21: Nessus Total Results from Near-RT RIC	92
Table 22 Tool Alignment with SSDLC Phases for O-RAN	98

List of Figures

Figure 1: The structure of the O-RAN Alliance Technical Steering Committee	7
Figure 2: Architecture of O-RAN.....	11
Figure 3: Non-RT RIC Reference Architecture.....	16
Figure 4: Near-RT RIC Internal Architecture.....	22
Figure 5: Testing Focus in O-RAN Architecture.....	39
Figure 6: The Installation Results of Non-RT RIC	42
Figure 7: The Installation Results of Near-RT RIC.....	42
Figure 8: Secure Software Development Lifecycle (SSDLC)	43
Figure 9: OWASP Dependency Check Total Results from Non-RT RIC	52
Figure 10: OWASP Dependency Check CVE Results Categorize by Severity	53
Figure 11: Embold Total Results from Non-RT RIC	67
Figure 12: SonarQube Total Results from Non-RT RIC	69
Figure 13: Nikto Total Results from Non-RT RIC.....	74
Figure 14: Trivy Total Results from Non-RT RIC	78
Figure 15: Trivy CVE Results Categorize by Severity from Non-RT RIC.....	79
Figure 16: Metasploit with Nmap Integration Total Results from Non-RT RIC	81
Figure 17: Kube-hunter Total Results from Non-RT RIC.....	84
Figure 18: OWASP Dependency Check Total Results from Near-RT RIC.....	85
Figure 19: Embold Total Results from Near-RT RIC	89
Figure 20: SonarQube Total Results from Near-RT RIC	90
Figure 21: Trivy Total Results from Near-RT RIC.....	94
Figure 22: Trivy CVE Results Categorize by Severity from Near-RT RIC	94
Figure 23: Metasploit with Nmap Integration Total Results from Near-RT RIC.....	96
Figure 24: Kube-hunter Total Results from Near-RT RIC	97

Figure 25: OpenVAS Vulnerability Scan Results Before Remediation 101

Figure 26: iptables Rules for Dropping ICMP Timestamp Requests and Replies 102

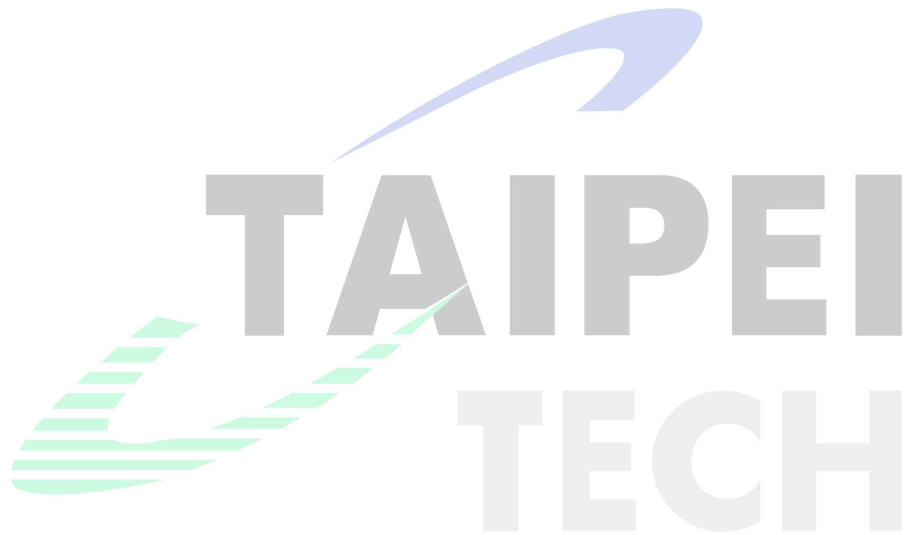
Figure 27: ICMP Timestamp Reply Testing Results..... 102

Figure 28: ICMP Timestamp Request Testing Results..... 102

Figure 29: OpenVAS Vulnerability Scan Results After Remediation 103

Figure 30: Kubernetes API Server Configuration with Debugging Handlers Disabled 106

Figure 31: Kube-hunter Vulnerability Scan Results After Remediation..... 107



Chapter 1 Introduction

The rapid evolution of the telecommunications industry and the growing demand for seamless connectivity have been driven by the rise of groundbreaking technologies like 5G networks [1]. As the future of cellular telecommunication advances, 5G is set to transform how we connect and engage with the digital world. Central to this transformation is the Open Radio Access Network (O-RAN) [2], which represents a significant shift in network architecture. O-RAN's open and interoperable design allows for greater flexibility and innovation in 5G deployments, offering benefits such as enhanced network security, reduced costs, and the ability to accommodate a broader range of suppliers. This paradigm shift is expected to drive advancements across various sectors, from telecommunications to IoT applications, solidifying O-RAN's role as a cornerstone of future 5G infrastructure. By leveraging software-defined principles, virtualization, and disaggregation, O-RAN empowers operators to realize the full potential of 5G by offering greater flexibility, scalability, and cost-efficiency compared to traditional proprietary systems. This architectural transformation enables operators to optimize network deployments, enhance service delivery, and accommodate a variety of use cases that require rapid and responsive connectivity.

O-RAN utilizes the principles of Software Defined Networking (SDN) and Network Function Virtualization (NFV) to enhance flexibility, scalability, and cost-effectiveness within the radio access network sector. This is accomplished by separating hardware from software, O-RAN disaggregates the traditional monolithic base station into smaller, interoperable functional modules, while SDN provides a centralized, software-defined control plane for dynamic management and orchestration. NFV enables the virtualization of network functions, allowing for efficient resource allocation and rapid deployment of virtualized functions. Together, SDN, NFV, and O-RAN enable operators to build a flexible, agile, and cost-effective infrastructure that supports diverse 5G use cases and fosters innovation in the

telecommunications industry [3].

Open Radio Access Network (O-RAN) [2] architecture is designed to revolutionize the RAN industry, making it more open, adaptable, and intelligent. Security analyses of O-RAN highlight its architectural blueprint, risk areas, threat actors, and potential threats, emphasizing the need for increased security measures. These analyses address vulnerabilities, such as the communication interface between network controllers, and propose mitigation strategies. Additionally, they explore the security and privacy challenges posed by Open RAN, discuss relevant standardization efforts, and emphasize the importance of secure design. The virtualized and open nature of O-RAN introduces new risks, requiring comprehensive risk assessments, security analyses, and testing to ensure information security. Overall, these studies stress the significance of prioritizing security to ensure the future security and sustainability of O-RAN networks.

The O-RAN interface [4] refers to the connection and interaction between different network controllers within the architecture. These controllers are responsible for managing and coordinating various functions and components of the RAN. In O-RAN, the network controllers play an essential role in enabling communication between different elements of the RAN, such as base stations, virtualized network functions, and centralized management systems. They ensure proper coordination and control of the RAN operations, including the management, configuration, and optimization of radio resources. The interface for communication between these network controllers is an indispensable element within the O-RAN framework. It facilitates the exchange of control signals, management information, and data among the controllers. This interface allows them to collaborate and coordinate their actions to ensure seamless operation and efficient management of the RAN.

However, the communication interface can also be a potential point of vulnerability if not properly secured [5]. An attacker who gains unauthorized access or manipulates the communication interface may disrupt the coordination between network controllers,

compromise the security and privacy of the information being exchanged, or inject malicious commands or data into the system. To address these security concerns, it is important to analyze and identify potential vulnerabilities and threats associated with the communication interface. This includes evaluating the robustness of the protocols, encryption mechanisms, authentication, and access control measures employed in the interface. By understanding these vulnerabilities, researchers, and practitioners can propose appropriate mitigation strategies and security enhancements to protect the communication interface and ensure the comprehensive security of the O-RAN architecture.

Establishing standards [6] is crucial for both the advancement and effective deployment of O-RAN architectures. It involves the establishment of common protocols, interfaces, and specifications that facilitate interoperability and compatibility among various components and vendors within the O-RAN ecosystem. Standardization efforts aim to ensure that O-RAN implementations from different vendors can seamlessly work together, fostering a more open and competitive market. By adhering to common standards, O-RAN deployments become more flexible, allowing network operators to combine components from different vendors, promoting innovation, and minimizing vendor lock-in.

From a security perspective, standardization efforts also address security requirements, guidelines, and best practices specific to O-RAN. These security standards aim to define security mechanisms, protocols, and procedures to protect the O-RAN infrastructure from various threats and vulnerabilities. Standardization organizations like the O-RAN Alliance [7], work collaboratively to create and refine these protocols for security. They engage industry stakeholders, including network operators, equipment vendors, researchers, and regulators, to ensure a comprehensive and effective approach to security.

By adhering to security standards, O-RAN deployments can benefit from proven security measures, as well as ongoing advancements and updates. Standardization also facilitates the development of security certification processes, enabling network operators to

assess and verify the security posture of O-RAN solutions and make informed procurement decisions. Overall, standardization efforts in the context of O-RAN are essential for establishing a common framework that ensures interoperability, flexibility, and security. They provide a foundation for secure and harmonious collaboration among different components, vendors, and stakeholders within the O-RAN ecosystem.

Security risks in O-RAN [8] refer to the potential vulnerabilities and threats that arise from the adoption and implementation of this innovative mobile network infrastructure. O-RAN introduces a software-defined and virtualized network architecture, which offers benefits such as increased flexibility and interoperability. However, it also brings along specific cybersecurity risks that organizations need to address to protect and secure their network systems.

One significant security risk in O-RAN is related to network vulnerabilities. The use of open interfaces and APIs in O-RAN's design can expose the network to a wider range of cyber threats [9]. Malicious actors can target vulnerabilities in software components or exploit insecure APIs to gain unauthorized network access. These attacks can lead to various consequences, including breaches of data security, compromising data integrity, or launching many other cyberattacks.

While O-RAN presents several advantages, including increased flexibility and cost-effectiveness in mobile network infrastructure, it also brings along specific cybersecurity risks that organizations need to address. The open nature of O-RAN, with its software-defined and virtualized network architecture, creates a more interconnected and accessible environment, which can potentially expose the network to a broader spectrum of cyber threats.

The open and interoperable design of O-RAN facilitates innovation and flexibility but also introduces significant security vulnerabilities. The separation of network components and the dependency on software-driven functionalities can create multiple potential attack points if not adequately protected. Integrating security measures into the Software Development

Lifecycle (SDLC) from the initial design phase is essential for mitigating these risks. Implementing a Secure Software Development Lifecycle (SSDLC) ensures that security best practices are embedded throughout the development process, from design to deployment and maintenance. This strategy not only aids in the early detection and resolution of potential security issues but also embeds security as a fundamental component of the O-RAN architecture, thereby strengthening the network against potential threats and attacks.

The aim of this paper is to investigate the security leakage of O-RAN and explore the effectiveness of robust security testing tools in mitigating such leaks. By examining the current state of O-RAN security and evaluating the capabilities of advanced security testing tools, we can develop an all-encompassing framework to strengthen the security posture of O-RAN deployments. This framework will not only protect against potential leaks but also strengthen the overall resilience of the telecommunications infrastructure, instilling confidence in the viability and security of O-RAN networks.

Through this research, our objective is to enrich the existing knowledge base on O-RAN security while offering actionable recommendations tailored for operators, vendors, and regulatory bodies to strengthen the security of their O-RAN deployments. By proactively addressing security leakage through the application of robust security testing tools, we can enable the widespread adoption of O-RAN while maintaining the integrity and confidentiality of essential network infrastructure in the face of evolving security threats.

Chapter 2 Background

2.1 O-RAN ALLIANCE

The O-RAN Alliance [7], also known as the Open Radio Access Network Alliance, is an international industry consortium established in 2018. It aims to promote open and interoperable standards for the Radio Access Network (RAN) within mobile telecommunications systems. By developing open interfaces and specifications, the alliance enables multi-vendor interoperability, fosters competition, and avoids vendor lock-in. The O-RAN Alliance concentrates on multiple facets of RAN, including virtualization, network intelligence, and the application of artificial intelligence and machine learning. Its goal is to create a flexible, cost-effective, and efficient RAN architecture that supports the evolving requirements of 5G and future mobile networks. The alliance collaborates with other standardization bodies to ensure alignment and widespread the implementation of open RAN standards within the industry.

The O-RAN Alliance consists of 11 work groups (WGs) and 3 focus groups (FGs). The Technical Steering Committee (TSC) oversees the WGs responsible for O-RAN specification work, each covering a distinct segment of the O-RAN framework. Security is of utmost importance in the O-RAN architecture, leading to the creation of the Security Focus Group (SFG) to manage security elements throughout the open RAN ecosystem. While focus groups typically don't engage in O-RAN specifications, the SFG operates differently. It has released multiple technical specifications and operates similarly to a work group (WG), resulting in its transformation into WG11 (Security) Figure 1 [10].

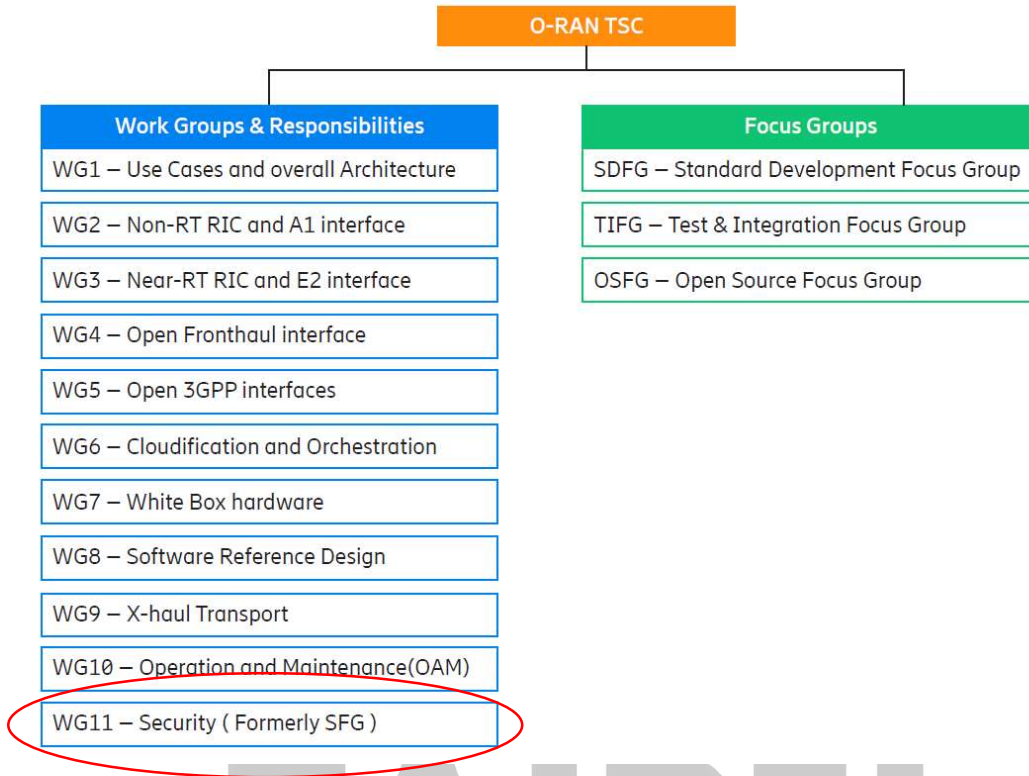


Figure 1: The structure of the O-RAN Alliance Technical Steering Committee

The O-RAN ALLIANCE Security Work Group (WG11) [11] is actively working on developing specifications to create a secure and interoperable open RAN system for mobile network operators. In 2022, they elevated the Security Focus Group (SFG) to a Work Group, underscoring their commitment to security in the design of their systems.

WG11 regularly updates the public about their progress through announcements. These announcements outline their activities, focus areas, security controls, and timelines. Their work is structured around four key security specifications:

1. O-RAN Security Threat Modeling and Remediation Analysis 4.0: This entails a risk-based methodology to identify and mitigate potential threats, facilitating the creation of a secure O-RAN framework.
2. O-RAN Security Requirements Specifications 4.0: These specifications detail the security requirements for all components of O-RAN, addressing areas such as confidentiality, integrity, and availability protection. They include critical security

controls such as authentication, authorization, and the principle of least privilege access control.

3. O-RAN Security Protocols Specifications 4.0: This specification outlines the implementation requirements for security protocols utilized within O-RAN, including SSH, IPSec, DTLS, TLS 1.2, and TLS 1.3.
4. O-RAN Security Tests Specifications 3.0: This documentation details the security testing procedures necessary to validate and verify the implementation of security functions, configurations, and protocol requirements in O-RAN. It's an essential step toward ensuring the verifiability of O-RAN security requirements.

These security specifications are meant to ensure that O-RAN are secure, reliable, and compliant with industry standards. WG11 updates the specifications regularly, and these updates can be found on the O-RAN ALLIANCE website.

2.2 O-RAN Software Community (OSC)

The O-RAN Software Community (OSC) [12] is a joint effort between the O-RAN Alliance and the Linux Foundation, with the goal of creating software for the Radio Access Network (RAN) in the context of the telecom industry's transformation and the emergence of 5G technology. The community intends to utilize existing LF network projects, while tackling challenges related to performance, scalability, and 3GPP alignment. Open source is seen as a crucial means to speed up product development collaboratively and economically.

The focus of the OSC is to align with the O-RAN Alliance's open framework and specifications, enabling industry deployment. As an emerging open-source community within the Linux Foundation, it will develop open source software for disaggregated radio access networks that are modular, open, smart, efficient, and flexible.

These following are the O-RAN software version released by OSC [13]:

- Amber (A) Release : (Nov 2019)
- Bronze (B) Release : (Jun 2020)
- Cherry (C) Release : (Dec 2020)
- D release : (Jul 2021)
- E Release : (Dec 2021)
- F Release : (Jun 2022)
- G Release : (Dec 2022)
- H Release : (Jun 2023)
- I Release : (Dec 2023)
- J Release : (Jun 2024)
- K Release : (Dec 2024)

The O-RAN Software Community (OSC) plays a crucial role in strengthening the security posture of the O-RAN ecosystem. As a collaborative open-source community, OSC faces various security threats commonly found in the software development landscape [14]. These threats encompass cybersecurity risks, supply chain vulnerabilities, interoperability challenges, and the potential for malicious contributions. OSC actively addresses these threats by adopting industry best practices, including regular code audits, vulnerability management, and secure development guidelines [15]. They emphasize supply chain security by validating code contributions and scrutinizing third-party dependencies to minimize the risk of compromised software components. Furthermore, OSC promotes awareness and education among its community members to enhance their ability to identify and mitigate security vulnerabilities.

The OSC's commitment to security extends to interoperability and integration concerns within the O-RAN ecosystem. While interoperability is a key objective, it can introduce risks if not managed effectively. OSC conducts thorough testing and validation to guarantee the compatibility and security of software integrations, thus reducing the risk of vulnerabilities arising from misconfigurations or incompatibilities [16]. By addressing these security challenges comprehensively, OSC strives to create a robust and secure O-RAN ecosystem, making it a safer environment for telecommunications network deployment and operation. To stay current with OSC's evolving security practices and efforts in addressing security threats, it is advisable to consult their latest resources and official documentation.

2.3 O-RAN Architecture

The architecture of O-RAN represents a revolutionary transformation of traditional Radio Access Networks by embracing key principles such as openness, virtualization, intelligence, interoperability, flexibility, cost-effectiveness, and innovation. By emphasizing openness and promoting vendor-neutral interfaces, O-RAN enables network operators to diversify their RAN components, leading to healthy competition and driving innovation in the industry. The introduction of virtualization further enhances the architecture, decoupling software functions from hardware and providing the flexibility to deploy network functions on commodity hardware or in the cloud. This virtualized approach ensures efficient resource utilization and scalability, enabling operators to adapt their networks quickly to meet evolving demands.

From Figure 2[17] the main part of O-RAN framework is the RAN Intelligent Controller (RIC) [18], which adds intelligence and real-time optimization capabilities to the network. The RIC dynamically manages RAN functions, optimizing resource allocation, and enhancing overall network performance. Standardized interfaces facilitate seamless communication and coordination between different RAN components, streamlining network

integration and simplifying upgrades. The interoperability achieved through these interfaces ensures that diverse components from multiple vendors can work together cohesively.

This level of flexibility and interoperability, combined with O-RAN's emphasis on cost-effectiveness, empowers operators to make optimal infrastructure investments and reduce operational expenses. Moreover, the focus on innovation fosters collaboration and opens the door for third-party developers to contribute to the ecosystem, driving the creation of cutting-edge technologies and services. Ultimately, the O-RAN architecture empowers network operators to deliver high-quality services, adapt swiftly to changing demands, and embrace a new era of wireless network deployments [19].

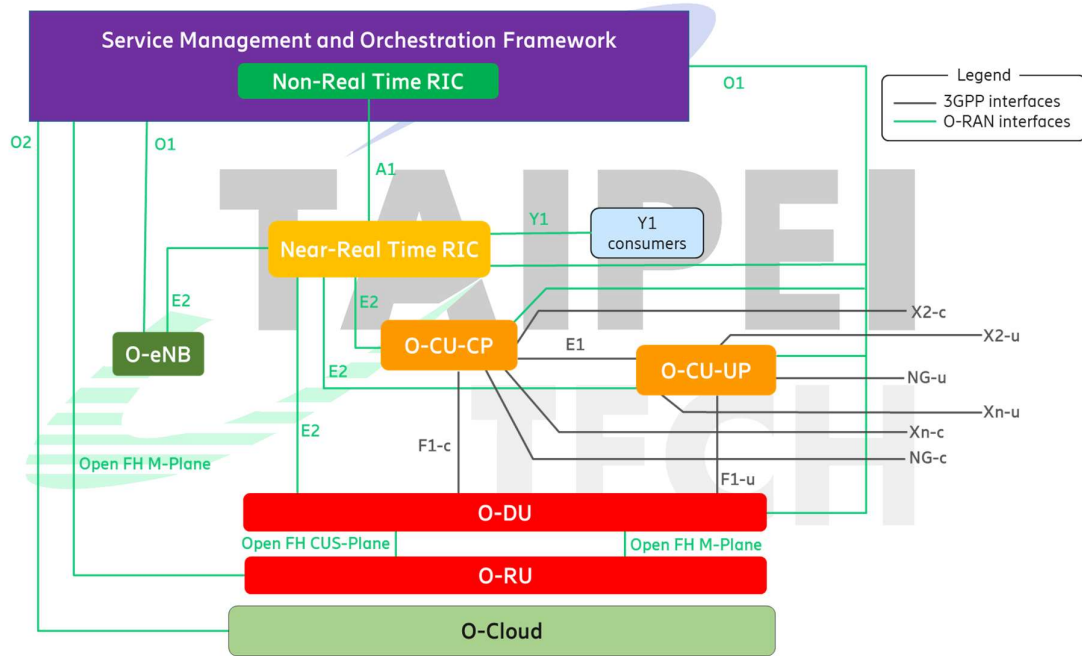


Figure 2: Architecture of O-RAN

O-RAN is also a universal and standardized framework for mobile telecommunications networks, closely aligned with open source and open interface principles [20]. It emphasizes openness, interoperability, and standardization to break down the traditionally closed and proprietary nature of radio access networks. O-RAN promotes open interfaces between network elements, like radio units (RUs), distributed units (DUs), and centralized units (CUs). These open interfaces encourage collaboration between different

vendors, fostering interoperability and competition. Moreover, O-RAN leverages open-source software principles, including initiatives like the O-RAN Software Community (OSC) [16], which offers software building blocks, reference designs, and tools for developing radio access network components. This approach not only enhances flexibility and innovation but also contributes to lower costs, ultimately benefiting both network operators and end-users in the mobile telecommunications industry [21].

Security threats and attacks in the realm of open source and open interface technologies [22] resemble those encountered in any software or networked environment. Common risks include malware and ransomware threats that can compromise systems, zero-day exploits exploiting undiscovered vulnerabilities, DoS attacks overwhelming systems, supply chain attacks compromising trusted software sources, man-in-the-middle attacks intercepting and altering data, data breaches revealing sensitive information, authentication and authorization bypass vulnerabilities, code injection attacks potentially leading to system compromise, web application vulnerabilities like XSS and CSRF, and brute force attacks exploiting weak configurations. Vigilance, patching, strong security measures, and proactive monitoring are essential to mitigate these risks [23].

2.4 O-RAN Components

2.4.1 SMO (Service Management and Orchestration Framework)

Service Management and Orchestration (SMO) is a crucial element within O-RAN framework [24]. It is responsible for orchestrating and managing various services and resources in the radio access network. SMO's primary functions include service orchestration, resource management, automation of network tasks, service assurance, and ensuring interoperability between different network components and vendors. By performing these tasks, SMO helps optimize resource utilization, reduce operational costs, and enhance service

quality, making it an essential component in the development of open and efficient telecommunications networks, particularly in the 5G and beyond 5G era. The essential capabilities of the SMO that provide RAN support in O-RAN include:

- FCAPS* interface to O-RAN Network Functions
- Non-RT RIC for RAN optimization
- O-Cloud Management, Orchestration and Workflow Management

(*FCAPS = Fault, Configuration, Accounting, Performance, Security)

The SMO delivers these services via 4 primary interfaces to the O-RAN elements [17].

- - A1 Interface: Connects the Non-RT RIC in the SMO to the Near-RT RIC for RAN optimization.
- - O1 Interface: Links the SMO to the O-RAN network functions for FCAPS support.
- - In the hybrid model, Open Fronthaul M-plane interface: Connects the SMO to the O-RU for FCAPS support.
- - O2 Interface: Connects the SMO to the O-Cloud to provide platform resources and manage workloads.

Ensuring the security of the Service Management and Orchestration (SMO) component within the O-RAN architecture is crucial for creating a self-regulating network environment. This is crucial for maintaining the overall performance of O-RAN and protecting subscriber data and privacy. Insufficient authentication and authorization protocols for both external and internal SMO connections can lead to unauthorized access. This could compromise sensitive Open RAN information and enable malicious entities to interfere with network operations. O-RAN's foundation in RAN virtualization brings deployment-specific security challenges related to virtualization and software-defined networking, including issues with VM migration, instantiation, hypervisor, orchestration, and SDN controller security [25].

Furthermore, in O-RAN's cloud-native deployment, the shared BBU pool introduces privacy and data access risks. While O-RAN offers flexible services, it's crucial to consider these security challenges in the context of its open and virtualized approach.

DoS attacks or a surge in traffic can lead to overloads, impacting the accessibility of SMO data and functions. Vulnerabilities in orchestrator configurations, access controls, and isolation measures can be exploited by attackers. In scenarios where a single orchestrator oversees multiple virtual machines and containers that are taken care of by many teams and possess varying levels of sensitivity, improper user and group access permissions could allow an attacker or negligent user to disrupt the operation of other virtual machines or containers under the orchestrator's management. Additionally, there is a risk of harmful network traffic emanating from various virtual machines or containers that share the same virtual networks, especially when virtual machines or containers with varying sensitivity levels utilize the same virtual network, leading to insufficient isolation and potentially compromising network security. Proper security measures and access control are essential to mitigate these risks which are described in Table 1 [8].

Table 1: SMO risks in O-RAN

Threat	Description
Lack of or incorrect authentication	Improper or missing authentication on SMO functions can be exploited to gain unauthorized access to the SMO and its functionalities. [26].
Denial-of-Service attacks	Executes excessive load or Inundating DoS attacks on SMO [26].
Security concerns related to orchestration	Takes advantage of weak orchestrator configuration, insufficient access control, and poor isolation [26].

2.4.2 Non-RT RIC (Non Real Time RAN Intelligent Controller)

Non Real Time RAN Intelligent Controller (Non-RT RIC) [27] is a vital component within the O-RAN framework, situated in SMO layer, facilitating intelligent optimization of RAN. Its primary role is to offer guidance based on established policies, manage ML models, and enrich data for the Near-RT RIC through A1 interface. Additionally, Non-RT RIC can carry out intelligent management of radio resources at non-real-time periods, typically exceeding 1 second, and leverage data analysis and artificial intelligence and machine learning methods to identify improvement measures. It interfaces with SMO services for example data acquisition and provisioning services, as well as the O1 and O2 interfaces, to access and exchange necessary data for RAN enhancement.

Non-RT RIC consists of 2 sub-functions [17]:

- **Non-RT RIC Framework:** This functionality is internal to the SMO Framework, logically terminating the A1 interface and exposing the necessary services to rApps via its R1 interface.
- **Non-RT RIC Applications (rApps):** These are modular applications that utilize the functionality provided by the Non-RT RIC Framework to carry out RAN optimization and other functions. The services exposed to rApps through the R1 interface allow them to gather information and initiate actions (e.g., policies, re-configuration) via the A1, O1, O2, and Open FH M-Plane related services.

The Non-RT RIC Framework is responsible for exposing all necessary functionalities to the rApps, whether these functionalities originate from the Non-RT RIC Framework itself or the SMO Framework.

From Figure 3 [27] explains the reference architecture of the Non-RT RIC a component of the SMO framework. There are three categories of logical functionalities within the Non-RT RIC framework and SMO framework.

- **Functions Anchored Inside the Non-RT RIC Framework:**

These functions are integral to the Non-RT RIC framework, closely tied to its core operations, and responsible for core RAN optimization tasks. This is indicated in solid blue box.

- **Functions Anchored Outside the Non-RT RIC Framework:**

These functions exist externally but interact with the Non-RT RIC through defined interfaces, providing complementary services like data collection from O-RAN nodes. This is indicated in solid orange box.

- **Non-Anchored Functions:**

These functions operate independently, often without direct integration with the Non-RT RIC, but can still influence or be influenced by RAN behavior within the O-RAN architecture. This is indicated in dashed line box.

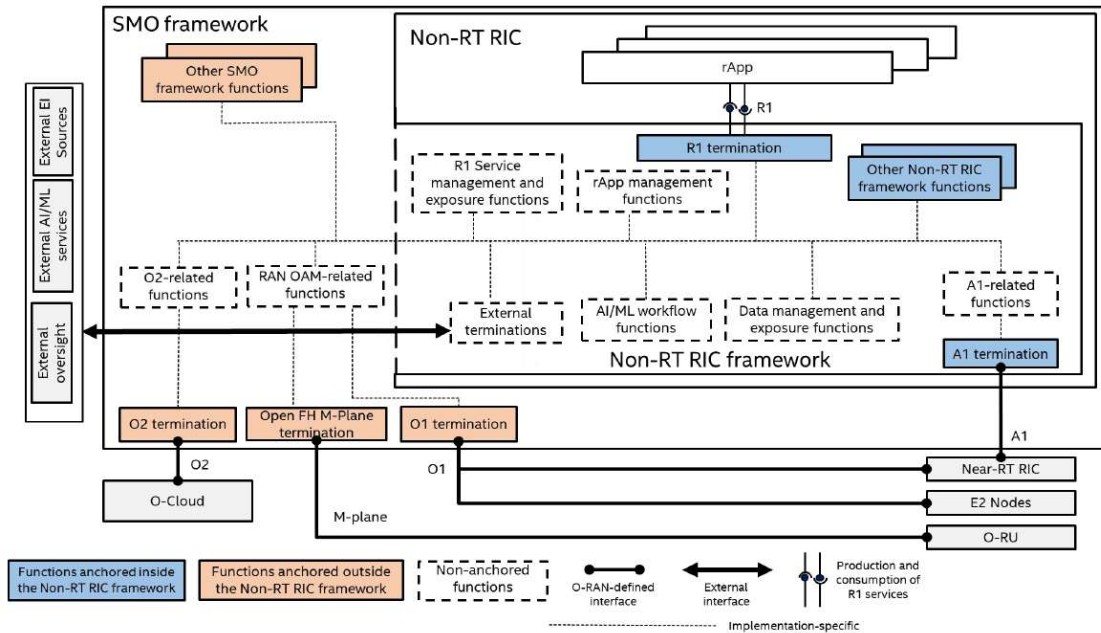


Figure 3: Non-RT RIC Reference Architecture

rApps within the Non-RT RIC framework [28] can impact critical functions like AI/ML model creation, A1 policy administration, data improvement, and network configuration optimization, which can be exploited for purposes like degrading network

performance, initiating DoS attacks, and intercepting enrichment data such as UE site, track, directions, and GPS information. These rApps share similarities with xApps and can influence the behavior of specific cells, groups of UEs, or individual UEs, leading to attacks similar to xApps. These attacks can arise from malicious rApps, vulnerable rApps, misconfigured rApps, compromised rApps, or conflicting rApps that are summarized in Table 2 [8]. In addition to these familiar risks, two additional vulnerabilities specific to Non-RT RIC are identified.

Table 2: Non-RT-RIC risks

Threat	Description
DDoS attack	An attacker breaches the component to initiate attacks or reduce efficiency [26].
Sniffing attacks through the A1 interface for UE identification	An attacker conducts UE sniffing in the Non-RT RIC through the A1 interface or via the R1 interface using rApps to identify UE. For instance, a rApp could potentially be used as a "sniffer" for UE identification [26].
Vulnerabilities and misconfiguration in rApps	Vulnerabilities can potentially exist in any rApp if it is sourced from an untrusted or unmaintained origin. An attacker exploits these vulnerabilities and misconfigurations in such rApps to disrupt the provided network service and potentially take over another rApp or the entire Non-RT RIC [2], [26], [29].
Weak authentication and authorization in rApps	If software vulnerabilities are present in web front-end or REST API interfaces or lack proper authentication and authorization mechanisms, an attacker could exploit these weaknesses to bypass controls and access the rApp,

	impersonating a tenant. This would allow the attacker to manipulate configurations, access logs, and implement backdoors [26], [29], [30].
Compromising isolation in rApps	Attackers can breach rApp separation and escape restrictions. This allows them to conduct an auxiliary route attack, extracting data from co-hosted within a common asset [26].
Conflicts in rApps	Conflicting directly, indirectly, and implicitly either inadvertently or with malicious intent impacts non-realtime Open RAN system functions like managing carrier license schedules, optimizing energy usage, and handling subscriptions. This can result in reduced efficiency or even Denial of Service attacks. [26].

Untrusted or poorly maintained sources can introduce vulnerabilities [31] into any rApp, potentially leading to disruptions in network services and even compromising the entire Non-RT RIC. By taking advantage of these weaknesses, attackers might manipulate information communicated over the A1 interface and extracted crucial data, or take control of other rApps. Furthermore, attackers can exploit rApp isolation to escape from confinement and access data from co-hosted rApps. Unpermitted entry opens avenues for exploiting vulnerabilities [32] in additional rApps or components of Open RAN, facilitating actions like intercepting and spoofing network traffic and launching DoS attacks. Attackers may also infiltrate the Non-RT RIC via A1 interface or O1 interface or outside entities via SMO to initiate attacks or reduce efficiency.

rApps can create conflicts due to their launch by different vendors with varying objectives like carrier license management or energy efficiency measures. These conflicts can manifest as direct, indirect, or implicit, according to the specific factors and their effects.

Direct conflicts involve multiple rApps requesting the same parameter change, indirect ones occur when different parameter changes yield opposing effects, and implicit conflicts arise when parameter changes impact network states. These conflicts can result in network performance degradation and instabilities, and they are challenging to mitigate due to hidden dependencies. Additionally, there's a vulnerability risk if rApp management interfaces are exposed to web front-ends [33] or REST APIs with software interface vulnerabilities [34] or inadequate authentication and authorization, potentially allowing attackers to gain unauthorized access, pose as tenants, alter configurations, access logs, or establish backdoors.

2.4.3 Near-RT RIC (Near Real Time RAN Intelligent Controller)

Near Real Time RAN Intelligent Controller (Near-RT RIC) [35] plays a crucial role in near-real-time management and optimization of E2 Nodes' functions and resources, utilizing accurate data collection and actions with control loops functioning within 10 milliseconds to 1 second. It contains one or more xApps utilizing the E2 interface for gathering near real-time data, including individual user or cell-based information, and provide value-added services. The Near-RT RIC's control over E2 Nodes is guided by policies and enrichment data from the Non-RT RIC via the A1 interface. This allows the Near-RT RIC to produce RAN analytics information, which can be accessed through the Y1 interface. The distribution of Radio Resource Management (RRM) functions between the Near-RT RIC and the E2 Node is determined by the E2 Service Model. This model specifies the capabilities of the E2 Node and outlines the specific RRM responsibilities for each function. In the event of a Near-RT RIC failure, basic services will continue to operate; however, value-added services that depend on the Near-RT RIC may experience interruptions.

xApps are essential to the Near-RT RIC infrastructure in telecommunications networks [36]. These software applications collect and process fine-grained, near real-time data from network elements, like user equipment and base stations, enabling real-time network control

and optimization. xApps operate based on predefined policies, providing value-added services like dynamic network management, load balancing, and traffic steering. They respond dynamically to changing network conditions, ensuring optimal performance. xApps are customizable, allowing network operators to develop specialized solutions, and they work in conjunction with the RIC architecture to enhance network efficiency, quality of service, and user experience.

There are many functions in Near RT RIC. The details are shown below and these functions architecture are shown in Figure 4 [35]:

- **Database, and related SDL (Shared Data Layer) services**, which enables reading and writing of RAN/UE information and other data necessary to accommodate particular use cases;
- **xApp subscription management**, this function consolidates subscriptions from multiple xApps and facilitates the unified distribution of data to these xApps;
- **Conflict mitigation**, which addresses potentially intersecting or conflicting demands from multiple xApps;
- **Messaging infrastructure**, this function enables message interaction among internal functions within Near-RT RIC;
- **Security**, this function sets up the security framework for xApps.;
- **Management Function:**
 - Provision of fault management, configuration management, and performance management services to the SMO.
 - Implementation of logging, tracing, and metrics collection to capture, monitor, and collect the status of Near-RT RIC internals, with the ability to transfer this data to an external system for further evaluation.

- **Interface Termination:**
 - E2 interface termination from an E2 Node.
 - A1 interface termination from the Non-RT RIC.
 - O1 interface termination from the SMO.
 - Y1 interface termination from a Y1 consumer.
- **Functions hosted by xApps**, which enable services to be executed at the Near-RT RIC, with the results being sent to E2 Nodes through the E2 interface;
- **API Enablement** A function that facilitates operations pertaining to the Near-RT RIC API, encompassing tasks such as managing the API repository and registry, handling authentication, enabling discovery, and supporting generic event subscriptions;
- **AI/ML support:**
 - Pipeline management, training processes, and performance monitoring for xApps.
- **xApp Repository Function:**
 - Selecting xApps for A1 message routing based on A1 policy types and operator policies;
 - Managing access control of A1-EI types for xApps according to operator policies.

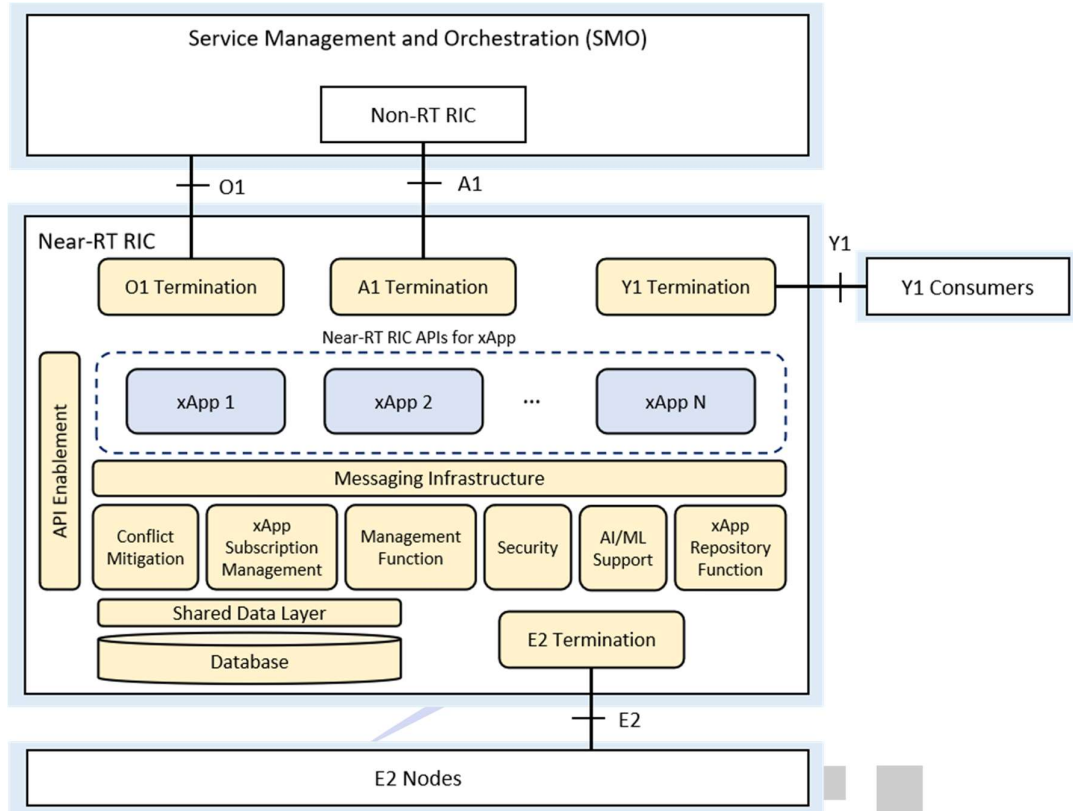


Figure 4: Near-RT RIC Internal Architecture

xApps possess the ability to influence the behavior of specific cellular elements, groups of user equipment (UEs), and individual UEs. Security concerns arise from various sources, including harmful xApps, xApps with security weaknesses, improperly configured xApps, breached xApps, and xApps that have conflicts [26]. These threats are significant because xApps are engineered for executing intelligent operations. related to radio resource management for cellular and device entities. A compromised xApp can potentially take control of cells or devices, enabling tracking of specific consumers within the network. Moreover, malicious xApps can access priority information via the A1 interface, compromising location privacy and service prioritization, ultimately leading to compromised RAN (Radio Access Network) performance and privacy violations that are summarized in Table 3 [8].

Table 3: Near-RT-RIC risks

Threat	Description
Tracking the location of User Equipment (UE) and altering UE priority caused by malicious xApps	XApps have the capacity to impact the behavior of individual cells, clusters of User Equipments (UEs), and particular UEs for purposes such as subscriber tracking or adjusting UE priority levels [26], [29].
Identification of User Equipment (UE) facilitated by malicious xApps	Malicious xApps can manipulate the identification of User Equipments (UEs) and monitor their locations. For instance, an xApp might serve as a 'sniffer' to identify UEs [26], [29], [30].
Vulnerabilities and misconfigurations within xApps	Potential vulnerabilities may be present in any xApp sourced from an untrusted or unmaintained origin. Attackers can exploit these vulnerabilities and misconfigurations in xApps with the intention of disrupting the current network service and possibly taking over another xApp or all of near-RT RIC [2], [26], [29].
Conflicts in xApps	Contradictory xApps, whether deployed inadvertently or with malicious intent, can impact the operations of the O-RAN, including management of mobility, admission controls, bandwidth allocation, and load distribution, leading to reduced performance. Additionally, a malicious entity could exploit a harmful xApp to deliberately activate RRM actions that contradict the internal decisions of O-gNB, with the aim of causing a Denial-of-Service (DoS) situation. [26], [29].

Undermining the isolation of xApps	By undermining xApp isolation, an attacker can escape xApp confinement, enabling the execution of a side-channel attack that could facilitate the extraction of data from co-located xApps within a common resource pool [26].
------------------------------------	--

Malicious xApps can serve as tools for unauthorized User Equipment (UE) identification [37], potentially leading to adverse impacts on Radio Access Network (RAN) performance and subscriber privacy violations. This risk arises because the A1 interface can pinpoint specific UEs in the network via their unique identifiers, creating correlations among anonymized UE identities between RAN nodes. Consequently, malicious actors could track UE locations and alter UE priorities, posing a significant threat, especially when identifying and tracking important subscribers like Very Important Persons (VIPs). E2 signaling channels are more prone to exposing UE identifiers compared to A1, primarily because of the Near-RT conditions inherent in E2. Furthermore, these malicious xApps might tamper with Service Level Agreement (SLA) specifications and priority levels, potentially conflicting with Near-RT-RIC decision processes, leading to breaches of specified execution boundaries and SLAs.

Vulnerabilities in xApps pose significant risks, as they can originate from untrusted or poorly maintained sources [38]. Exploiting these vulnerabilities can lead to compromising other xApps or even the entire Near-RT RIC, typically with the intention of impairing performance, such as through Denial of Service (DoS) attacks. Attackers might also tamper with information exchanged across A1 or E2 interfaces, enabling the extraction of sensitive information. Additionally, The mechanisms for segregation for xApps may possibly be altered to escape confinement and access data from concurrently hosted xApps. Unauthorized access chances to take advantage of weaknesses in other xApps or Open RAN components., enabling network traffic interception, spoofing, and service degradation (DoS attacks). The fact that xApps are open-source makes their weaknesses more apparent to potential adversaries,

whereas misconfigurations and incompatibilities represent inherent risks in the O-RAN ecosystem.

Lack of distinct functional separation between Near-RT RIC with Open RAN Next Gen Node B (O-gNB) can result in disputes, both unintentional and malicious, including conflicts within xApps. These conflicts can affect decisions related to radio resource management, impacting critical Open RAN functions like management of mobility, admission controls, bandwidth regulation, and load distribution may lead to potential performance deterioration. Preserving isolation for xApps is crucial to ensure the independent functioning of O-RAN services and the precise decision-making of Near-RT-RIC. However, this isolation can be undermined by weaknesses in the system, deduction of access information through shared resources, or deceptive authentication attempts [39], potentially allowing attackers to subdue xApp operations.

2.4.4 O-CU (Open Central Unit)

O-CU, or Open Central Unit, is a critical component that is crucial in the radio access network [40]. Traditionally, the Central Unit is responsible for functions like radio resource management and coordination. However, in O-RAN, the O-CU is an open and standardized version of the Central Unit, designed to be flexible, software-defined, and vendor-neutral. It acts as a bridge between the Radio Unit (RU) and the Distributed Unit (DU), promoting interoperability and openness while allowing for network virtualization and efficient, cost-effective network management. O-CU's software-defined nature and adherence to O-RAN Alliance specifications encourage interoperability, reduce vendor lock-in, and facilitate the deployment of agile, 5G-ready networks.

O-CU-CP (Open Central Unit - Control Plane) and O-CU-UP (Open Central Unit - User Plane) [41] are specific subcomponents within O-CU in O-RAN framework, each with distinct functions:

2.4.4.1 O-CU-CP (Open Central Unit - Control Plane)

O-CU-CP is responsible for managing the control plane functions in the radio access network. The control plane is primarily concerned with signaling, network management, and control operations, such as call setup and mobility management. O-CU-CP handles these control functions, including radio resource management, connection establishment, and handovers. It interfaces with other network elements like the Distributed Unit (DU) and the core network to ensure the efficient governance and administration of radio resources and network services.

2.4.4.2 O-CU-UP (Open Central Unit - User Plane)

O-CU-UP focuses on the user plane functions within the radio access network. Handling actual data traffic and user data packets falls under the responsibility of the user plane, such as internet content, voice calls, or video streams. O-CU-UP manages the processing and forwarding of user data, ensuring low-latency and high-throughput delivery of data to and from the Radio Unit (RU). It is designed to efficiently process and transport user data packets while minimizing delays and ensuring a high-quality user experience.

Both O-CU-CP and O-CU-UP are integral parts of the O-CU in O-RAN, working together to manage CP and UP elements of the radio access network. These subcomponents adhere to open standards and interfaces defined by the O-RAN Alliance, contributing to the network's flexibility, interoperability, and cost-effectiveness while enabling innovative solutions and promoting vendor diversity in the network ecosystem.

In an Open RAN cloud-native setup, the shared unit pool could lack adequate isolation, potentially endangering user privacy and compromising the security of confidential information [42]. The transparency and visibility of CU components in Open RAN, especially with the use of eCPRI for fronthaul, make them susceptible to cyber intrusions and hacking attempts, posing a greater risk compared to traditional fronthauls and C-RAN [43]. Although

uncommon, intrusions can happen through the F interface in the Mid-haul, linking the CU to DU. These intrusions may exploit threat vectors such as service migration, offloading, or transfer mechanisms in edge computing infrastructure hosting CU [44]. If a CU is compromised, it can potentially impact both the fronthaul and backhaul directions using the open interfaces of O-RAN.

O-CU (Open Central Unit) in the O-RAN architecture is susceptible to a diverse array of security threats, including Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks that can disrupt network services, Man-in-the-Middle attacks that compromise data integrity, unauthorized access by intruders, malware infections, insider threats, and zero-day vulnerabilities. Supply chain attacks, interoperability risks, and data leakage can further endanger O-CU's security. Regulatory violations and social engineering attacks may result in legal consequences. To safeguard O-CU, robust security measures like access controls, encryption, intrusion detection systems, regular updates, and security awareness training are crucial, alongside adherence to security standards and collaboration within the O-RAN community.

Mitigating these threats is essential to protect O-CU from potential disruptions, data breaches, and regulatory repercussions. Implementing these security measures is necessary not only to ensure network integrity and user privacy but also to preserve trust and reliability within O-RAN context where connections are open and interoperability is an essential objective.

2.4.5 O-DU (Open Distributed Unit)

O-DU (Open Distributed Unit) plays a pivotal role in the transformation of traditional radio access networks. It is designed to disaggregate and virtualize critical functions within the network, bringing about increased flexibility and interoperability [45]. O-DU is responsible for tasks such as protocol termination, radio resource management, and baseband

processing, functioning as a key bridge between the O-RU (Open Radio Unit) and O-CU (Open Central Unit) in the network's data plane. Network operators can leverage this separation to implement best-in-class solutions from multiple vendors, reducing vendor lock-in and promoting innovation.

One of the key features of O-DU is its adherence to open standards and interfaces defined by the O-RAN Alliance. These open interfaces enable seamless communication and interoperability between O-DU and other O-RAN components, fostering collaboration and innovation within the telecommunications industry. Additionally, O-DU supports virtualization, allowing it to run as software on standard off-the-shelf hardware. This not only reduces capital expenses but also enables dynamic scaling and resource allocation based on the changing demands of the network, which is essential in the 5G era and future network deployments.

O-DU's flexibility and adaptability are further highlighted by its ability to be deployed in various network scenarios. It can be utilized in macrocells, small cells, and even at the network edge, making it a versatile component suitable for diverse 5G and future network deployment strategies. Overall, O-DU in O-RAN is a critical element in the quest for open, intelligent, and agile radio access networks. It empowers network operators to build more flexible, interoperable, and innovative RAN solutions, which can lead to cost savings, enhanced network performance, and a more competitive and dynamic telecommunications landscape.

Security risks encompass the potential consequences of security threats. They can manifest in various forms, including data breaches where sensitive information is exposed, leading to a loss of confidentiality. Data integrity compromises involve unauthorized alterations or destruction of data, eroding trust in data accuracy [46]. Service disruptions, often caused by DDoS attacks or cyber incidents, can disrupt operations, leading to downtime and financial losses. Organizations also face financial risks associated with the costs of

investigating, mitigating, and recovering from security incidents, including fines, legal fees, and compensation to affected parties. Reputation damage is a significant risk, as loss of trust and credibility among customers, partners, and stakeholders can result in declining customer bases and partnerships. Furthermore, non-compliance with data protection and privacy regulations can have serious legal consequences, potentially leading to regulatory fines.

The O-DU (Open Distributed Unit) is susceptible to a variety of cyberattacks. Distributed Denial of Service (DDoS) attacks can overwhelm the O-DU with an avalanche of traffic, causing service disruptions and downtime. Malware attacks, such as viruses and Trojans, can compromise the O-DU's integrity and spread across the network. Phishing attacks target network administrators, attempting to deceive them into revealing sensitive credentials, thereby gaining unauthorized access. Man-in-the-Middle (MitM) attacks can intercept and manipulate communication, compromising data integrity and confidentiality. Insider threats, stemming from individuals with access to the O-DU, pose significant risks, potentially leading to unauthorized access or data breaches. Software exploits can take advantage of vulnerabilities in the O-DU's software, granting attackers unauthorized control. Zero-day attacks are particularly concerning as they exploit previously unknown flaws before patches are available. Brute force attacks involve systematic attempts to gain O-DU access by trying numerous username and password combinations. Data integrity and security are at risk from interception and injection attacks. These attacks have the potential to impact network operations and threaten security, and lead to unauthorized access, data breaches, or service outages, emphasizing the critical need for robust cybersecurity measures to safeguard the O-DU and the broader O-RAN network.

The impact of security threats can be wide-ranging and significant, affecting both individuals and organizations. Financial impacts include the costs associated with investigating, mitigating, and recovering from security incidents, along with possible legal expenses, regulatory penalties, and reparations to affected parties. Operational impacts are

common and result from disruptions to normal business operations, including downtime, loss of data, and system unavailability. Reputation damage can be severe, leading to a loss of trust and confidence in the organization, which, in turn, can result in a reduced customer base and fewer partnerships. In cases of intellectual property theft, the loss of sensitive information, trade secrets, or proprietary data can have long-term consequences for an organization's competitive advantage. The multifaceted nature of these impacts underscores the importance of robust cybersecurity measures, proactive policies, and training to safeguard against security threats and minimize their potential repercussions.

2.4.6 O-RU (Open Radio Unit)

The O-RU, or Open Radio Unit, serves as a cornerstone component of the network's architecture, ushering in a new era of openness, flexibility, and innovation in the telecommunications industry. O-RUs are hardware entities typically deployed at cell sites and are charged with managing the network's radio aspects. This includes housing radio transceivers, antennas, and necessary processing functions. What makes O-RUs especially significant is their open nature, featuring standardized interfaces that promote interoperability among various components within the network. This open architecture empowers network operators to transcend vendor lock-in and select components from different suppliers, fostering a more competitive and dynamic telecommunications landscape.

One of the key functions is to connect O-DU and O-CU by a fronthaul connection, which is typically based on established standards like Common Public Radio Interface (CPRI) or Ethernet. This separation of the radio unit (O-RU) from the distributed unit (O-DU) offers a high degree of flexibility in network design, allowing operators to tailor their networks to specific requirements and use cases. O-RUs also have the capability to support virtualization, enabling their integration into virtualized network environments. This not only enhances resource utilization but also streamlines network management, making it more efficient and

adaptable to changing demands.

Furthermore, O-RUs are designed to be versatile, accommodating various frequency bands and radio access technologies, making them well-suited for the dynamic landscape of 5G and future generations of wireless technology. By facilitating the integration of multi-band and multi-RAT (Radio Access Technology) support, O-RUs are instrumental in ensuring networks remain adaptable to evolving technologies and user needs. Overall, O-RUs are pivotal in the O-RAN framework, as they provide a standardized, interoperable, and flexible interface that not only separates and connects radio equipment but also underpins the broader goals of O-RAN to drive innovation, reduce operational costs, and increase the effectiveness of wireless networks.

Security threats in O-RUs within O-RAN (Open Radio Access Network) systems pose significant risks to network integrity, confidentiality, and availability. These threats encompass unauthorized access, potentially leading to network disruptions and eavesdropping on sensitive data. Denial of Service (DoS) attacks are aimed at overwhelming O-RUs with excessive traffic, resulting in network downtime and a reduction in service quality. Man-in-the-Middle (MitM) attacks can intercept and manipulate data, compromising the integrity and confidentiality of communications. Firmware and software exploitation seek to exploit vulnerabilities, allowing attackers to gain unauthorized control over O-RUs, enabling them to make unauthorized configuration changes and causing network disruptions. Spoofing and impersonation may lead to unauthorized access and data manipulation. Eavesdropping on radio signals can compromise data confidentiality, while physical attacks and malware/viruses have the potential to disrupt operations and compromise network security.

Protecting O-RUs against these threats necessitates the implementation of strong security protocols, such as cryptographic techniques, robust authentication, authorization controls, systems for detecting intrusions, regular security updates, and adherence to security policies. The open architecture of O-RAN can enhance security through the promotion of

multi-vendor solutions and transparency but ensuring security in O-RU deployments requires close collaboration among network operators, equipment vendors, and regulatory bodies to establish and enforce security standards and best practices.

2.4.7 O-Cloud

The O-Cloud Platform is a comprehensive system comprising both hardware and software components designed to offer cloud computing capabilities for executing Radio Access Network (RAN) functions [47]. The hardware includes computing, networking, and storage elements, with the potential incorporation of acceleration technologies to meet performance goals. The software component provides open and well-defined APIs, facilitating the coordination and administration of the NF Deployment's lifecycle and the O-Cloud itself. Notably, the software is independent of the hardware, allowing flexibility in sourcing from different vendors.

The management of cloudified Radio Access Network functions brings about new considerations, as the mapping between network functionality and physical hardware can vary based on the chosen scenario. This variability requires flexibility in the design of management aspects related to physical elements rather than logical ones. Examples include logging of physical functions, scale-out actions, and considerations related to survivability, all of which are influenced by the chosen mapping between network functionality and physical hardware.

Relocating Open-RAN elements to the cloud creates a unique threat environment, especially regarding the possible actions of a compromised cloud provider [48]. Recent risk assessments accurately emphasize that a cloud provider managing the O-Cloud has capabilities similar to those of the RAN operator. Given the current scarcity of required safety protocols in O-RAN standards, 2 particular suggestions emerge to address these concerns: firstly, integrating security measures, such as Secure Execution Environments secondly, embedding incorporating compulsory access control and security protocols into the O-RAN

framework. While a compromised cloud provider could threaten RAN security, operators typically plan to establish and manage proprietary data centers instead of depending on external cloud services. Consequently, assigning the same level of trust to the O-Cloud operator as to the RAN-Operator effectively mitigates the scenario involving a malicious cloud provider.

Furthermore, it is strongly advised to exclusively utilize trusted data centers and cloud services for the O-Cloud. Protecting against compromised cloud providers [49], particularly by utilizing confidential computing and secure execution environments, presents a significant challenge due to numerous attack vectors arising from a powerful attacker model. The recommendation emphasizes that if the O-Cloud is trusted and follows standard security best practices in both configuration and design, the expected security risk associated with a cloud-based RAN should be minimal.

2.5 O-RAN Interfaces

The O-RAN interfaces serve as the primary enablers of the O-RAN vision, striving to establish a more open, intelligent, and flexible Radio Access Network. The O-RAN interfaces connect different RAN components from different vendors and allow them to exchange data and control signals. The O-RAN interfaces also provide access to the RAN Intelligent Controllers, which can optimize and manage the network using AI and ML techniques. The O-RAN interfaces are built upon the foundation of 3GPP standards, but with some extensions and modifications to support the O-RAN features and functions [4].

2.5.1 3GPP interfaces

3GPP-defined interfaces are the ones that follow the standards and specifications of the 3GPP organization [50], which is responsible for developing and maintaining mobile communication technologies, such as 2G, 3G, 4G, and 5G. 3GPP-defined interfaces are

designed to ensure the compatibility and interoperability of the RAN components across different vendors and operators. 3GPP-defined interfaces also support the evolution and enhancement of the RAN functionalities and features, such as network slicing, multi-connectivity, and massive MIMO.

The following interfaces are defined and maintained by 3GPP, but seen also as part of the O-RAN architecture [17]:

- E1 interface: To enable the communication and coordination between the O-CU-CP and the O-CU-UP Roles [51].
- F1-c interface: To enable the control plane communication and coordination among the O-CU-CP and the O-DU Roles [52].
- F1-u interface: To enable the control plane communication and coordination among the O-CU-UP and the O-DU Roles [52].
- NG-c interface: To enable the control plane communication and coordination among the O-CU-CP and the 5GC Roles [53].
- NG-u interface: To enable the control plane communication and coordination among the O-CU-UP and the 5GC Roles [53].
- X2-c interface: To transmit the O-CU-CP information for the definition of interoperability profile specifications [54].
- X2-u interface: To transmit the O-CU-UP information for the definition of interoperability profile specifications [54].
- Xn-c interface: To transmit the O-CU-CP information for the definition of interoperability profile specifications [55].
- Xn-u interface: To transmit the O-CU-UP information for the definition of interoperability profile specifications [55].
- Uu interface: To transmit information between the User Equipment (UE) and the O-RAN components [56].

2.5.2 O-RAN interfaces

O-RAN-defined interfaces are the ones that follow the principles and guidelines of the O-RAN Alliance, which is an industry initiative that aims to promote open and intelligent RAN solutions. O-RAN-defined interfaces are designed to enable the innovation and customization of the RAN components by using open and modular architectures, software-defined networking, and artificial intelligence [6]. O-RAN-defined interfaces also support the integration and orchestration of the O-RAN network elements with the existing 3GPP network elements, such as the core network and the OSS.

The following interfaces are defined and maintained by O-RAN [17]:

- A1 interface: For support services between the Non-RT RIC function in SMO and the Near-RT RIC function [57].
- O1 interface: Providing administration and coordination functions to O-RAN components [58], [59].
- O2 interface: For managing the O-Cloud infrastructure and the network functions that run on it [47].
- E2 interface: To get events, control, and policy information between the Near-RT RIC function and the O-RAN network function [60].
- Y1 interface: To connect RAN services between the Near-RT RIC and the other systems.
- O-Cloud Notification interface: For notifies O-RAN workloads of O-Cloud events.
- Open Fronthaul interface: To transmit data and control signals between the O-DU and the O-RU [61], [62].

O-RAN security also aims to be consistent with 3GPP security specifications, which are the global standards for mobile networks. However, O-RAN security also addresses the specific challenges and requirements of the open and modular architecture, such as the

security of the new interfaces between different vendors [63] and the security of the virtualized and cloud-based components [64].

The statement underscores the imperative of ensuring robust security for communication interfaces within a network. It stresses the need for rigorous authentication and authorization processes for devices and entities, encompassing radio units, distributed units, centralized units, and orchestration/management systems [65]. Security measures should address the confidentiality, integrity, and availability of data [66] and control messages exchanged through these interfaces, spanning user plane data, control plane signaling, and management configuration commands. Additionally, the statement emphasizes the necessity of protection against a spectrum of malicious attacks [67], including denial-of-service, replay, spoofing, tampering, eavesdropping, and man-in-the-middle attacks. Finally, compliance with regulatory and legal requirements, such as privacy laws, lawful interception obligations, and network security standards, is deemed essential to establish a comprehensive and resilient security framework for network interfaces [68].

2.6 SSDLC and Security Testing Methods

The traditional Software Development Lifecycle (SDLC) is a well-established method in software engineering for designing, developing, testing, and deploying software. However, in this conventional approach, security considerations were frequently an afterthought or addressed late in the process, resulting in vulnerabilities and costly security issues that were difficult to fix. In contrast, the Secure Software Development Lifecycle (SSDLC) integrates security best practices into every stage of the software development process [69]. This approach marks a significant shift from traditional methods, embedding security as a core component from the beginning.

In the requirements analysis phase, security needs are identified and documented along with functional requirements. This involves threat modeling to pinpoint potential threats and

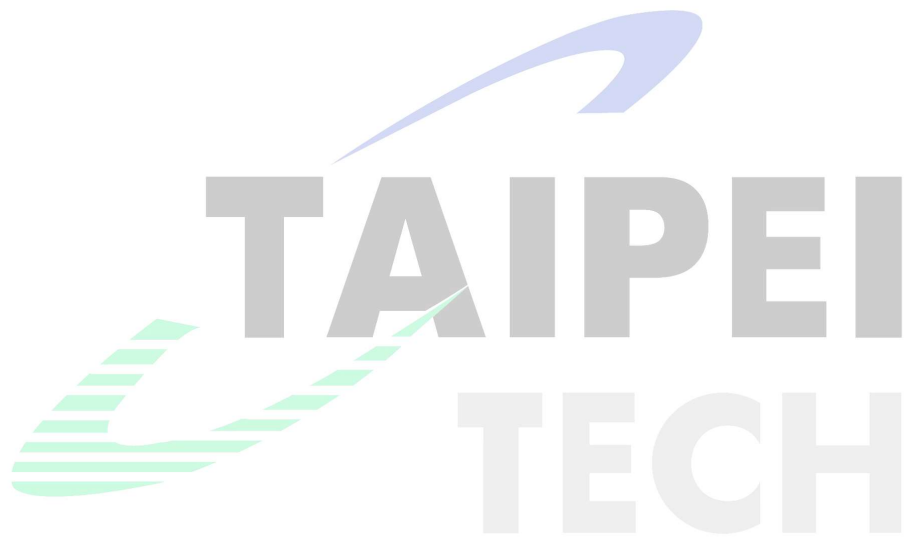
vulnerabilities and defining security controls to mitigate these risks. During the design phase, security is woven into the system's architecture and detailed design. Key tasks include developing a strong security architecture, conducting security design reviews, and applying secure design principles such as least privilege and defense in depth.

The implementation phase focuses on applying secure coding practices to prevent vulnerabilities. This includes adopting coding standards that mitigate common vulnerabilities and performing peer code reviews to catch security issues early. The testing phase involves thorough security testing to identify and address vulnerabilities before the software is deployed. Activities include penetration testing to find vulnerabilities that might not be detected through automated testing and security regression testing to ensure new code changes do not introduce new vulnerabilities.

During deployment, security measures are put in place to ensure a secure transition to the production environment. This includes configuring the application and environment securely, implementing secure release practices to prevent unauthorized changes, and setting up monitoring and logging systems to detect and respond to security incidents. Post-deployment, the maintenance phase focuses on maintaining the security posture of the application. This includes regular patch management to address vulnerabilities, developing and executing incident response plans to handle security incidents effectively, and conducting periodic security audits to identify and mitigate new risks.

Within the SSDLC framework, several security methods are employed to ensure robust protection against vulnerabilities. Software Composition Analysis (SCA) is used to identify and manage open-source components, detecting known vulnerabilities and compliance risks. Static Application Security Testing (SAST) analyzes source code for vulnerabilities early in the development cycle, while Dynamic Application Security Testing (DAST) evaluates the application in its running state to uncover runtime vulnerabilities [70]. Interactive Application Security Testing (IAST) combines elements of both SAST and DAST by analyzing code and

runtime behavior simultaneously, offering real-time feedback on security issues. Penetration Testing, or ethical hacking, involves simulating real-world cyberattacks to identify and mitigate security weaknesses before the application goes live. By integrating these security methods, the SSDLC ensures that security is embedded throughout the development process, leading to more secure and resilient software products.



Chapter 3 Implementation

3.1 Environment Setup

In this section, from Figure 5 we meticulously delineate the subject into two principal divisions to facilitate an exhaustive investigation into the intricacies of RAN intelligent controllers. The initial segment concentrates on Non-RT RIC. Following this, our investigation transitions to Near-RT RIC.

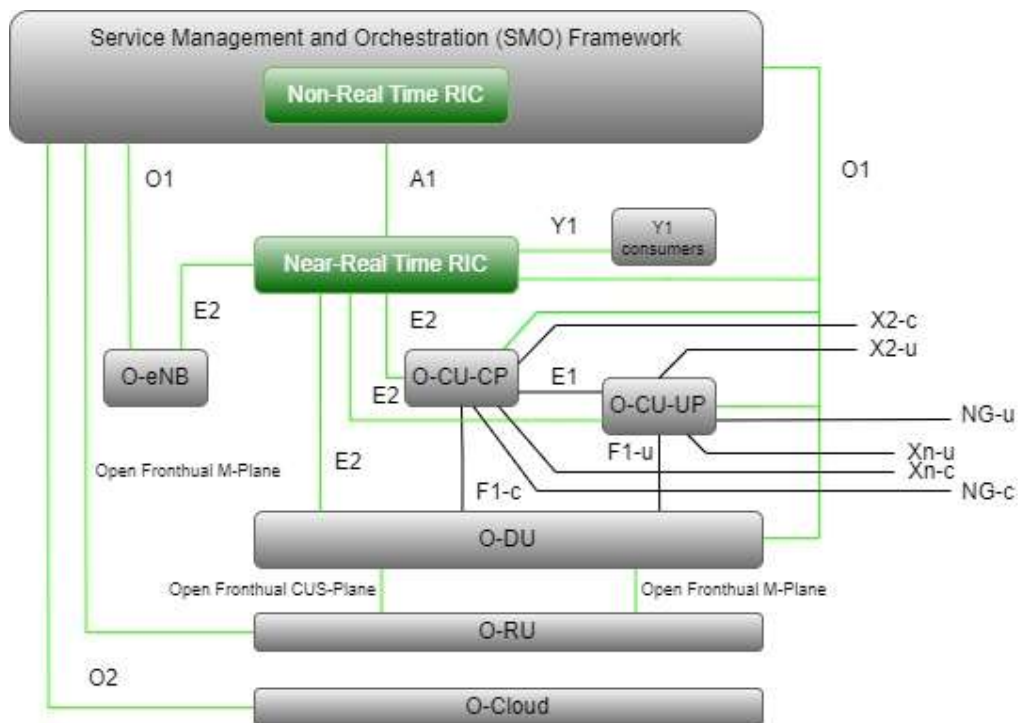


Figure 5: Testing Focus in O-RAN Architecture

Source: <https://docs.o-ran-sc.org/>

In the experimental framework's design and implementation, a rigorous selection of technological resources was crucial for ensuring the integrity and applicability of the findings. Table 4 presents an elaborate list of software requirements and operating systems, including precise versions of each software and OS utilized. This table underscores the importance of selecting tools that not only offer the latest features and security patches but also ensure compatibility and efficiency throughout the experimental processes.

Table 4: Software Requirements

Name	Description	Version
Ubuntu [71]	Ubuntu is a Linux distribution based on Debian and composed mostly of free and open-source software.	20.04
Docker [72]	Docker is an open platform for developing, shipping, and running applications.	24.0.7 (For Non-RT RIC) 24.0.5 (For Near-RT RIC)
Kubernetes (K8s) [73]	A system that is open-source, facilitating the automation of deployment, scaling, and management of containerized applications.	1.22.10 (For Non-RT RIC) 1.16.00 (For Near-RT RIC)
ChartMuseum [74]	Written in Go (Golang), ChartMuseum is an open-source Helm Chart Repository that supports cloud storage backends.	0.13.1 (For Non-RT RIC) 0.15.0 (For Near-RT RIC)
Helm [75]	Helm serves as a tool for handling Charts.	3.5.4 (For Non-RT RIC) 3.14.3 (For Near-RT RIC)
Non-RT RIC [27]	The functionality internal to the SMO in O-RAN architecture that provides the A1 interface to the Near-RT RIC	Release F
Near-RT RIC	A logical function via fine-grained data collection	Release F

[35]	and actions over the E2 interface and control over the E2 Nodes is steered via the policies and the enrichment data provided via A1 from the Non-RT RIC.	
------	--	--

Table 5 details the hardware requirements, illustrating the necessary computational and technical specifications designed to meet the demanding nature of the experiment. These requirements were established to eliminate potential bottlenecks, thus facilitating a smooth and efficient data processing environment.

Table 5: Hardware Requirements

Type	Specifications
CPU	6vCPU
Memory	64GB
Hard disk	200GB

Pre-condition setup process:

- Install Docker
- Install Kubernetes
- Setup ChartMuseum
- Setup Helm

3.1.1 Non-RT RIC Setup Process

To install Non-RT RIC, the initial step involves downloading the SMO Package. Subsequently, executing the O-RAN SMO Package on the Linux command line is necessary, with a detailed depiction provided in [76]. Once this process is completed, the installation results can be observed, as illustrated in Figure 6. This sequential procedure ensures the successful deployment of Non-RT RIC.

```

root@ubuntu:~# kubectl get pods -n nonrtric
NAME                                READY   STATUS    RESTARTS   AGE
a1-sim-osc-0-b5d7d6dff-rl9bg        1/1     Running   0           118s
a1-sim-osc-1-5f5448956b-hqqql       1/1     Running   0           118s
a1-sim-std-0-855b849b58-9q6mf       1/1     Running   0           118s
a1-sim-std-1-dd59fb685-9862s        1/1     Running   0           118s
a1-sim-std2-0-6dd988dfd7-cfqbn      1/1     Running   0           118s
a1-sim-std2-1-7dd668b674-zb59v     1/1     Running   0           118s
controlpanel-7b6bf6c56d-m5qzd       1/1     Running   0           118s
dmaapadapterservice-0               1/1     Running   0           118s
dmaapmediatorservice-0              1/1     Running   0           118s
helmanager-0                         1/1     Running   0           118s
informationservice-0                 1/1     Running   0           118s
nonrtricgateway-5c7b9c597-dd48s     1/1     Running   0           117s
oran-nonrtric-kong-594db9cb8b-ggn9d  2/2     Running   2 (113s ago) 117s
oran-nonrtric-odu-app-6459699f7-tf6vs 1/1     Running   0           117s
oran-nonrtric-odu-app-lcs-version-67c49b4bdc-cpczr 1/1     Running   0           117s
oru-app-56c594849b-wl44x            1/1     Running   0           117s
rappcatalogueservice-765444455b-g6gck 1/1     Running   0           117s
topology-6c5cd99d6d-5lrxt           1/1     Running   0           116s
root@ubuntu:~#

```

Figure 6: The Installation Results of Non-RT RIC

3.1.2 Near-RT RIC Setup Process

To initiate the installation process of Near-RT RIC, the first step entails acquiring the Near-RT RIC Package. Once obtained, the package is executed on the Linux command line, as illustrated in [77] for detailed guidance. Subsequently, upon completion of the installation process, users can observe the outcome depicted in Figure 7. This sequential procedure ensures the successful installation and deployment of Near-RT RIC.

```

root@ubuntu:~# kubectl get pods -n ricplt
NAME                                READY   STATUS    RESTARTS   AGE
deployment-ricplt-a1mediator-74f45b6bc6-q2b8x 1/1     Running   0           37d
deployment-ricplt-alarmmanager-7f7986fd57-z9gln 1/1     Running   0           37d
deployment-ricplt-appmgr-c47b999bc-j49wt        1/1     Running   0           37d
deployment-ricplt-e2mgr-855fdb9777-r778p        1/1     Running   0           37d
deployment-ricplt-e2term-alpha-6f97c4896d-qlmpk 1/1     Running   0           37d
deployment-ricplt-o1mediator-6f7d8998cf-9dscf    1/1     Running   0           37d
deployment-ricplt-rtmgr-5b7965bc8f-2sw6z        1/1     Running   4           37d
deployment-ricplt-submgr-f8fdfdb54-fm64b        1/1     Running   0           37d
deployment-ricplt-vespamgr-84f7d87dfb-b5vbk     1/1     Running   0           37d
r4-infrastructure-kong-646b68bd88-9mdhh         2/2     Running   1           37d
r4-infrastructure-prometheus-alertmanager-75dff54776-sgr4c 2/2     Running   0           37d
r4-infrastructure-prometheus-server-5fd7695-fvlfk 1/1     Running   0           37d
statefulset-ricplt-dbaas-server-0              1/1     Running   0           37d
root@ubuntu:~#

```

Figure 7: The Installation Results of Near-RT RIC

3.2 Testing Process

Central to our analysis in both divisions is the application of the Secure Software Development Lifecycle (SSDLC) framework as delineated in NIST 800-160 [78]. This comprehensive approach ensures that our examination of both components is underpinned by rigorous testing methodologies aligned with the SSDLC's best practices. By referencing the SSDLC in Figure 8, we systematically establish testing tasks designed to evaluate and enhance the security postures of both types of RAN intelligent controllers. This methodology not only provides a structured approach to assessing the controllers' resilience against threats but also aligns with industry-standard practices for secure software development. The detailed overview of these testing tasks, mapped against the various stages of the SSDLC, thereby offering a clear and actionable framework for our analysis can be catalogized as Table 6

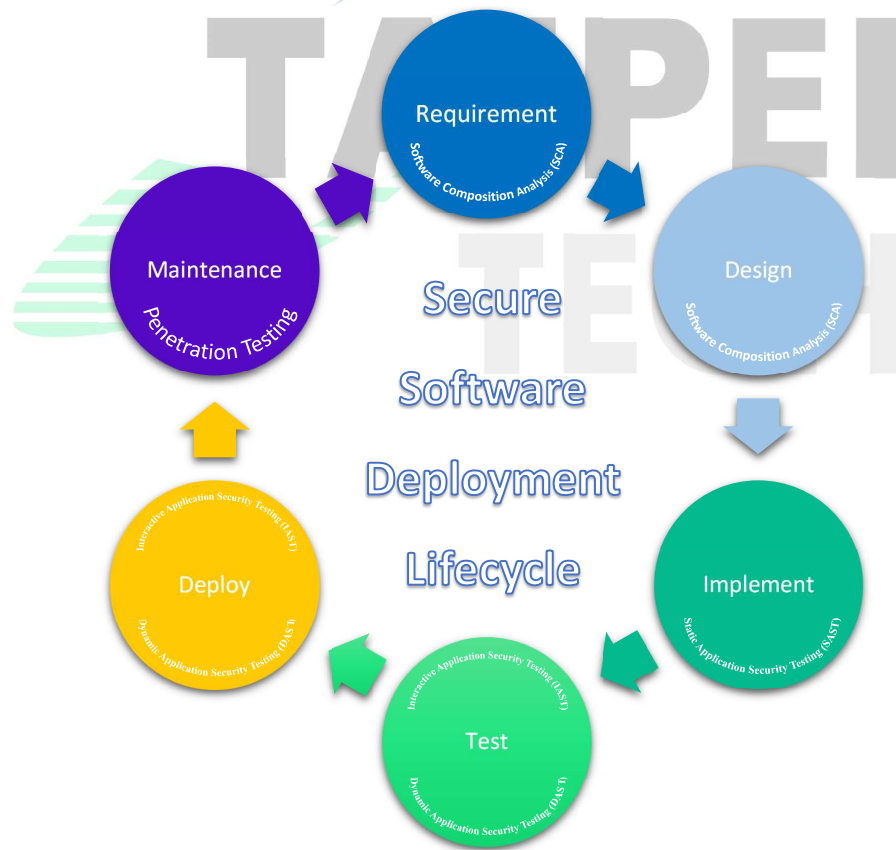


Figure 8: Secure Software Development Lifecycle (SSDLC)

Table 6: Security Testing Methods Across SSDLC

SSDLC Phase	Security Testing Methods
Requirements	Software Composition Analysis (SCA)
Design	Software Composition Analysis (SCA)
Implement	Static Application Security Testing (SAST)
Test	Interactive Application Security Testing (IAST)
	Dynamic Application Security Testing (DAST)
Deploy	Interactive Application Security Testing (IAST)
	Dynamic Application Security Testing (DAST)
Maintenance	Penetration Testing

Furthermore, Table 7 organizes the security analysis tools employed during the research. This categorization by tool type emphasizes the diverse approaches and methodologies adopted to scrutinize the data, ensuring a comprehensive security assessment from multiple perspectives.

Table 7: Security Analysis Tools

Method	Name	Description
SCA	OWASP Dependency Check	An SCA tool that aims to identify and uncover vulnerabilities in the dependencies of a project. https://owasp.org/www-project-dependency-check/
SCA	Mend.io (Bolt)	A complimentary tool that scans projects to find open-source components and their licenses, identifies known vulnerabilities and suggests fixes. https://www.mend.io/free-developer-tools/bolt/
SAST	Codacy	An automated tool for code analysis and quality assurance, helping developers release superior software in less time.

		https://www.codacy.com/
SAST	Aikido	<p>A developer-first software security platform. We scan your source code & cloud to show you which vulnerabilities are important to solve.</p> <p>https://www.aikido.dev/</p>
SAST	Embold	<p>Embold checks your code for design issues, code issues, security issues, and duplication & metric violations, and then assigns each method/class an overall as well as an individual rating for each of these issues.</p> <p>https://embold.io/</p>
SAST	SonarQube	<p>An automatic code review tool that is self-managed and systematically aids in producing Clean Code.</p> <p>https://www.sonarsource.com/products/sonarqube/</p>
DAST	Nessus	<p>A tool for remote security scanning checks computers for vulnerabilities and raises an alert if it discovers any that malicious hackers could exploit to access network-connected computers.</p> <p>https://www.tenable.com/products/nessus</p>
DAST	Trivy	<p>An intuitive and all-encompassing vulnerability scanner for containers and various artifacts.</p> <p>https://trivy.dev/</p>
IAST	Nikto	<p>A free command-line vulnerability scanner that examines web servers for dangerous files/CGIs, outdated server software, and other issues.</p> <p>https://cirt.net/Nikto2</p>

IAST	OpenVAS	The vulnerability scanner which is a software framework that includes several services and tools for scanning and managing vulnerabilities. https://www.openvas.org/
Pentest	Nmap (Network Mapper)	An open-source utility, used for network discovery and security audits. https://nmap.org/
Pentest	Metasploit	A cybersecurity tool that offers information on security vulnerabilities and assists with penetration testing and IDS signature development. https://www.metasploit.com/
Pentest	Kube-hunter	A free tool designed to detect security weaknesses in Kubernetes clusters, developed to enhance awareness and visibility of security issues in Kubernetes environments. https://github.com/aquasecurity/kube-hunter

Together, these tables elucidate the foundational elements of our experimental setup. They highlight the thoughtful integration of software, hardware, and analytical tools, which collectively enabled a robust exploration of the study's objectives, ensuring that the results are both credible and replicable.

The Common Vulnerabilities and Exposures (CVE) [92] system serves as a cornerstone in the cybersecurity domain, offering a comprehensive catalog of publicly disclosed cybersecurity vulnerabilities and exposures. Each entry within this catalog is assigned a unique CVE identifier, facilitating a standardized reference for specific vulnerabilities. This system is instrumental in enabling the seamless exchange of information regarding security vulnerabilities across various platforms and services within the cybersecurity community. By providing a unified reference point for vulnerabilities, the CVE

system enhances the ability of organizations to access and share critical information, thereby significantly contributing to the improvement of system security through the rapid dissemination of vulnerability details. This framework is essential for organizations seeking to safeguard their systems against cyber threats by staying informed about potential vulnerabilities and implementing timely protective measures.

The Common Weakness Enumeration (CWE) [93] system complements the CVE by providing a standardized catalog of software and hardware weaknesses that may result in vulnerabilities. CVE emphasizes particular vulnerability cases, while CWE organizes and describes the underlying issues that cause them. By addressing the root causes of security flaws, CWE enables developers and security practitioners to identify, prioritize, and mitigate weaknesses in their software and systems before they can be exploited. The synergy between CVE and CWE enhances the overall effectiveness of cybersecurity efforts by ensuring both vulnerabilities and their underlying weaknesses are systematically identified and addressed.

In parallel, the Common Vulnerability Scoring System (CVSS) [94] provides a robust framework for evaluating the severity of software vulnerabilities. Through a structured scoring system, CVSS assigns numerical values to vulnerabilities based on an array of metrics that assess the potential impact and exploitability of the vulnerability. Ranging from 0 to 10, these scores offer a quantitative measure of the vulnerability's severity, with higher scores indicating greater severity. The CVSS framework is divided into three scoring dimensions: Base, Temporal, and Environmental Scores. The Base Score reflects the inherent attributes of a vulnerability, the Temporal Score considers time-dependent factors, and the Environmental Score accounts for the specific impact on an individual organization's environment. This comprehensive scoring system enables organizations to prioritize their vulnerability management efforts effectively, focusing on mitigating the most severe threats to maintain system integrity and confidentiality. Table 8 provides a detailed explanation of the score ranges for each severity level.

Table 8: CVSS Severity Score Ranges

Severity	CVSS Score Range
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

The National Vulnerability Database (NVD) [95] complements the CVE system by serving as the U.S. government's repository of standards-based vulnerability management data. This includes annotations of the Common Vulnerabilities and Exposures (CVE) entries with additional impact metrics and detailed analysis. The NVD enhances each CVE entry with its own severity rankings and metadata, utilizing the Common Vulnerability Scoring System (CVSS) for a more detailed impact assessment. It integrates with various other security-related tools and resources, providing a deeper, more comprehensive view of each vulnerability. This enriched data not only includes technical specifics but also practical recommendations for mitigation and patching, making the NVD an invaluable tool for security professionals and IT administrators. By offering real-time updates and historical data, the NVD helps ensure that organizations have access to the most current and relevant information needed to address vulnerabilities effectively and maintain robust security protocols.

In conclusion, the integration of CVE, CVSS, CWE, and NVD creates a robust and cohesive approach to managing cybersecurity risks. CVE provides a standardized identification system for vulnerabilities, facilitating global communication and cooperation among cybersecurity experts. CVSS enhances this system by offering a detailed scoring method to evaluate the severity of vulnerabilities, helping organizations prioritize their security efforts effectively. CWE contributes by listing common software weaknesses, offering insights into the underlying causes of vulnerabilities and guiding developers to avoid

frequent security errors. NVD complements these frameworks by offering comprehensive analyses, impact assessments, and remediation strategies for CVE entries, making it an essential tool for thorough vulnerability management. Together, these frameworks underpin a proactive security infrastructure, enabling organizations to respond to known threats and anticipate and mitigate potential vulnerabilities. This combined use of resources is crucial for building strong defenses, maintaining operational continuity, and protecting sensitive information in an increasingly connected and threat-laden digital environment.



Chapter 4 Results Analysis and Demonstration

4.1 Non-RT RIC

In the security testing results of Non-RT RIC as followed from SSDLC in Table 6, we implemented a series of specialized testing techniques to ensure comprehensive protection. SCA initiated the process by scrutinizing system settings and configurations against industry best practices to prevent vulnerabilities due to misconfiguration. SAST, integral to the early stages of development, analyzed the source code for security flaws without executing the code, thereby identifying potential vulnerabilities such as improper input validation and insecure dependencies. IAST offered a real-time analysis by combining the methodologies of SAST and DAST to detect complex security issues during the application's operation. Following this, DAST was employed during later stages, testing the application in its running state to uncover runtime-related security flaws. Finally, Penetration Testing was conducted to simulate external cyber-attacks, rigorously testing the system's defenses to identify any exploitable security weaknesses. Together, these methods form a robust SSDLC framework, enhancing the security posture of the Non-RT RIC throughout its development and operational phases.

4.1.1 SCA

Software Composition Analysis (SCA) plays a crucial role in the Requirements and Design phases of the Secure Software Development Lifecycle (SSDLC). In the Requirements phase, SCA ensures that security requirements are defined with consideration for the use of third-party components by establishing security policies, ensuring compliance with industry standards, and guiding the selection of secure and reliable components. In the Design phase, SCA aids in managing dependencies, supports threat modeling by identifying potential vulnerabilities, and validates that the design adheres to security requirements and policies. By

integrating SCA into these early phases, security is considered from the outset, enabling early detection and mitigation of risks associated with vulnerable or non-compliant components. This proactive approach reduces the cost and effort required to fix vulnerabilities later and establishes a continuous feedback loop to refine security practices. Embedding SCA in the Requirements and Design phases significantly enhances the security and reliability of software products, ensuring that security concerns are an integral part from the beginning of the process.

In the context of cybersecurity, it is crucial to categorize the severity of potential threats to prioritize response strategies effectively. The severity of security vulnerabilities is categorized into 4 levels: Critical, High, Medium, and Low. For details see Table 9

Table 9: Severity Description

Severity	Description
Critical	Extremely high risk; potential for remote system control or data access; requires immediate remediation.
High	Significant risk; might allow substantial control over systems or access to sensitive information; requires quick prioritization.
Medium	Moderate risk; may need specific conditions to exploit; scheduled for regular update cycles.
Low	Minimal risk; limited impact on operations; routinely addressed in scheduled updates.

From Figure 9 the overview results of OWASP Dependency Check are pivotal in the identification and remediation of publicly disclosed vulnerabilities within project dependencies. The tool's efficacy is evidenced by the detailed report generated using version 9.0.10. This scan meticulously evaluated 75 unique dependencies, revealing a concerning figure of 62 vulnerable dependencies. Alarming, these vulnerabilities culminate in a total of 163 potential security risks. The absence of any suppressed vulnerabilities underscores the

report's unfiltered transparency and suggests an immediate call to action. Furthermore, the scan's reliance on the National Vulnerability Database (NVD), with the API last checked and modified in early April 2024, ensures that the most recent and relevant vulnerability data is incorporated into the assessment. This comprehensive approach not only identifies critical security flaws but also catalyzes the adoption of practice for security software, underscoring the indispensable nature of such tools in cybersecurity and the imperative to address the vulnerabilities identified with due diligence.

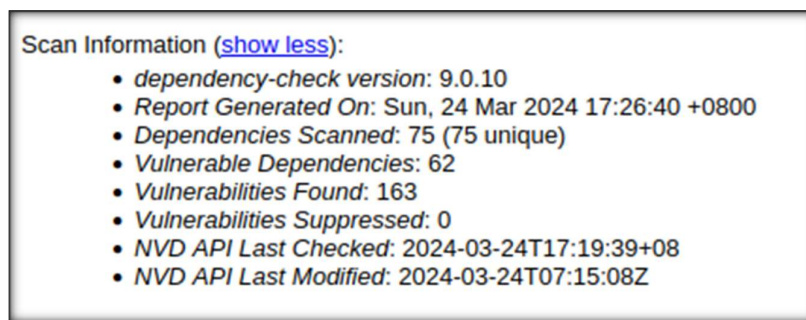


Figure 9: OWASP Dependency Check Total Results from Non-RT RIC

Figure 10 presents the pie chart of the results from OWASP Dependency Check, categorizing CVE by severity. The analysis reveals that most of the vulnerabilities are classified as high severity, comprising 61% of the total, equating to 99 instances. This is followed by 23% classified as critical severity, accounting for 38 instances. Medium severity vulnerabilities constitute 15% of the total with 24 instances, while low severity vulnerabilities are minimal, representing only 1% with a single instance. The distribution highlights the predominance of high and critical severity vulnerabilities, underscoring the need for prioritizing these issues in security remediation efforts.

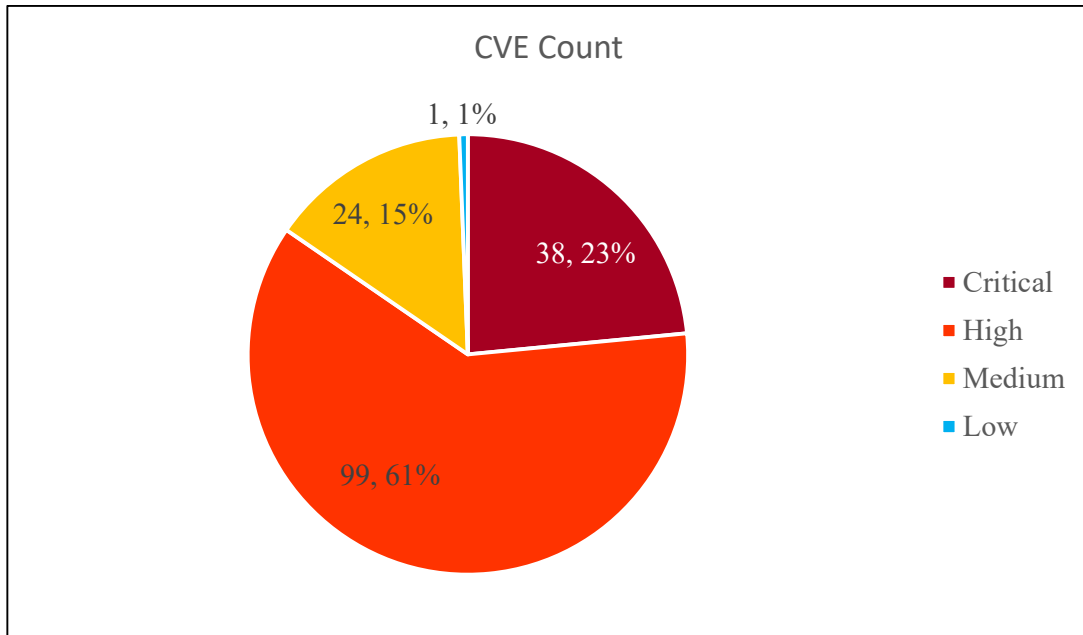


Figure 10: OWASP Dependency Check CVE Results Categorize by Severity

Table 10: OWASP Dependency Check Results and Solutions

No	Severity	Dependency Name (CVE)	Solutions
1	Critical	url-parse:1.4.7 (12)	Update url-parse version >= 1.5.9
2	Critical	loader-utils:1.4.0 (6)	Update loader-utils version >= 2.0.3
3	Critical	immer:1.10.0 (4)	Update immer version >= 9.0.6
4	Critical	ip:1.1.5 (2)	Update ip version >= 1.1.9
5	Critical	json-schema:0.2.3 (2)	Update json-schema version >= 0.4.0
6	Critical	minimist:1.2.5 (2)	Update minimist version >= 1.2.6

7	Critical	property-expr:2.0.2 (2)	Update property-expr version \geq 2.0.3
8	Critical	shell-quote:1.7.2 (2)	Update shell-quote version \geq 1.7.3
9	Critical	tough-cookie:2.5.0 (2)	Update tough-cookie version \geq 4.1.3
10	Critical	eventsourcing:1.0.7 (2)	Update eventsourcing version \geq 1.1.1
11	Critical	merge-deep:3.0.2 (1)	Update merge-deep version \geq 3.0.3
12	Critical	@babel/traverse:7.11.5 (1)	Update @babel/traverse version \geq 7.23.2
13	High	node-forge:0.9.0 (13)	Update node-forge version \geq 0.9.0
14	High	follow-redirects:1.5.10 (8)	Update follow-redirects version \geq 1.5.10
15	High	axios:0.19.2 (6)	Update axios version \geq 0.19.2
16	High	postcss:7.0.32 (6)	Update postcss version \geq 8.4.31
17	High	lodash:4.17.19 (4)	Update lodash version \geq 4.17.21
18	High	object-path:0.11.4 (4)	Update object-path version \geq 0.11.8
19	High	is-svg:3.0.0	Update is-svg version \geq 4.3.0

		(3)	
20	High	lodash-es:4.17.15 (3)	Update lodash version >= 4.17.21
21	High	ansi-html:0.0.7 (2)	Update ansi-html version >= 0.0.8
22	High	ansi-regex:3.0.0 (2)	Update ansi-regex version >= 3.0.1
23	High	ansi-regex:4.1.0 (2)	Update ansi-regex version >= 4.1.1
24	High	ansi-regex:5.0.0 (2)	Update ansi-regex version >= 5.0.1
25	High	async:2.6.3 (2)	Update async version >= 2.6.4
26	High	browserify-sign:4.2.1 (2)	Update browserify-sign >= 4.2.2
27	High	decode-uri- component:0.2.0 (2)	Update decode-uri-component version >= 0.2.1
28	High	dns-packet:1.3.1 (2)	Update dns-packet version >= 1.3.2
29	High	glob-parent:3.1.0 (2)	Update glob-parent version >= 5.1.2
30	High	json5:1.0.1 (2)	Update json5 version >= 1.0.2
31	High	json5:2.1.3	Update json5 version >= 2.2.2

		(2)	
32	High	nth-check:1.0.2 (2)	Update nth-check version >= 2.0.1
33	High	qs:6.5.2 (2)	Update qs version >= 6.5.3
34	High	qs:6.7.0 (2)	Update qs version >= 6.7.3
35	High	react-dev-utils:10.2.1 (2)	Update react-dev-utils version >= 11.0.4
36	High	semver:5.7.1 (2)	Update semver version >= 7.5.2
37	High	semver:6.3.0 (2)	Update semver version >= 7.5.2
38	High	semver:7.3.2 (2)	Update semver version >= 7.5.2
39	High	ssri:6.0.1 (2)	Update ssri version >= 8.0.1
40	High	ssri:7.1.0 (2)	Update ssri version >= 8.0.1
41	High	terser:4.8.0 (2)	Update terser version >= 4.8.1
42	High	tmpl:1.0.4 (2)	Update tmpl version >= 1.0.5
43	High	webpack-dev- middleware:3.7.2	Update webpack-dev-middleware version >= 7.1.0

		(2)	
44	High	word-wrap:1.2.3 (2)	Update word-wrap version >= 1.2.4
45	High	y18n:4.0.0 (2)	Update y18n version >=4.0.1
46	High	ini:1.3.5 (1)	Update ini version >= 1.3.6
47	High	minimatch:3.0.4 (1)	Update minimatch version >= 3.0.5
48	Medium	ajv:6.12.2 (2)	Update ajv version >= 6.12.3
49	Medium	browserslist:4.10.0 (2)	Update browserslist version >= 4.16.5
50	Medium	color-string:1.5.3 (2)	Update color-string version >= 1.5.5
51	Medium	elliptic:6.5.3 (2)	Update elliptic version >= 6.5.4
52	Medium	hosted-git-info:2.8.8 (2)	Update hosted-git-info >= 2.8.9
53	Medium	node-notifier:5.4.3 (2)	Update node-notifier version >= 8.0.1
54	Medium	path-parse:1.0.6 (2)	Update path-parse version >= 1.0.7
55	Medium	request:2.88.2 (2)	Update request version >= 2.88.3

56	Medium	ws:5.2.2 (2)	Update ws version >= 5.2.3
57	Medium	ws:6.2.1 (2)	Update ws version >= 6.2.2
58	Medium	debug:3.2.6 (1)	Update debug version >= 3.2.7
59	Medium	debug:4.1.1 (1)	Update debug version >= 4.3.1
60	Medium	jsdom:14.1.0 (1)	Update jsdomversion >= 16.4.0
61	Medium	jsonpointer:4.0.1 (1)	Update jsonpointer version >= 5.0.0
62	Low	es5-ext:0.10.53 (2)	Update es5-ext version >= 0.10.63

From Table 10 the OWASP Dependency Check results identify several vulnerabilities within the dependencies utilized in the project, categorized by severity: Critical, High, Medium, and Low. Critical vulnerabilities include dependencies such as `jsonwebtoken` (2.2.0), `lodash` (4.17.15), `marked` (0.6.2), `react` (16.0.0), and `axios` (0.18.0), necessitating immediate updates to their latest secure versions. High severity vulnerabilities present in `express` (4.16.4), `debug` (2.6.8), `angular` (1.7.8), and `jquery` (3.3.1) should be promptly addressed to mitigate associated risks. Medium severity vulnerabilities, such as those in `underscore` (1.9.1), `handlebars` (4.1.2), `moment` (2.22.2), and `node-fetch` (2.3.0), should be scheduled for updates during regular maintenance cycles. To effectively manage these updates, it is recommended to automate dependency management using tools like Dependabot or Renovate, integrating them into the CI/CD pipeline for continuous monitoring and mitigation. Post-update, rigorous testing in a staging environment is crucial to

prevent disruptions in the production environment. Systematically addressing these vulnerabilities will significantly enhance the project's security posture and mitigate potential risks.

Table 11: Mend.io Results from Non-RT RIC

CVE	Severity	CVSS	Vulnerable Library	Suggested Fix
CVE-2023-43804	High	8.1	urllib3-1.25.11-py2.py3-none-any.whl	Upgrade to version: urllib3 - 1.26.17,2.0.6
CVE-2021-33503	High	7.5	urllib3-1.25.11-py2.py3-none-any.whl	Upgrade to version: urllib3 - 1.26.5
CVE-2023-32681	Medium	6.1	requests-2.24.0-py2.py3-none-any.whl	Upgrade to version: requests -2.31.0
CVE-2023-45803	Medium	4.2	urllib3-1.25.11-py2.py3-none-any.whl	Upgrade to version: urllib3 - 1.26.18,2.0.7

Table 11 derived the results from Mend.io illustrate a detailed vulnerability assessment, providing a systematic breakdown of identified vulnerabilities within software libraries. The data includes each vulnerability's CVE identifier, Severity, CVSS score, the specific library affected, and the recommended remedial action. For instance, the table lists CVE-2023-43804 with a high severity rating and a CVSS score of 8.1, highlighting its critical nature and the urgent need for patching the `urllib3` library to versions 1.26.17 or 2.0.6. Similarly, CVE-2021-33503, also of high severity with a CVSS score of 7.5, underscores the necessity to upgrade the same library to version 1.26.5. This structured approach not only aids in prioritizing responses based on the severity and impact of the vulnerabilities but also delineates clear pathways for mitigating these vulnerabilities through specific updates. Such comprehensive vulnerability management is pivotal, as it underpins the thesis that meticulous and proactive software maintenance is crucial for enhancing system security and thwarting potential exploits that could lead to significant data breaches or system disruptions.

When discussing the relevance of security tools to O-RAN, it is crucial to consider both OWASP Dependency Check and Mend.io for their distinct yet complementary functionalities. OWASP Dependency Check provides a targeted focus on identifying known vulnerabilities within software dependencies. It utilizes specific identifiers and comprehensive databases such as NVD to pinpoint these vulnerabilities. This targeted focus is crucial for O-RAN, which rely heavily on a variety of software libraries and dependencies. Any known vulnerabilities in these components could pose significant risks, potentially compromising the entire network's security. By identifying and mitigating these vulnerabilities, OWASP Dependency Check helps to fortify the O-RAN system against specific, well-documented threats.

On the other hand, Mend.io is particularly valuable in O-RAN environments due to its comprehensive approach, which addresses both security and compliance issues by leveraging multiple databases and proprietary research. This holistic perspective is essential for O-RAN, which often involve a diverse mix of software components from various vendors. These components must adhere to rigorous industry standards and compatibility requirements, making a broad focus on security and compliance indispensable. Mend.io's ability to identify potential risks from a wide array of sources ensures that any compliance violations or security threats are promptly detected and addressed, thereby maintaining the integrity and reliability of the O-RAN system.

Therefore, the choice between OWASP Dependency Check and Mend.io should be guided by the specific security needs of the O-RAN system. OWASP Dependency Check excels at pinpointing and addressing known security vulnerabilities in software dependencies, offering a focused approach to threat mitigation. Meanwhile, Mend.io is better suited for ensuring broad security and compliance across various components, providing a comprehensive overview of potential risks. Employing both tools in tandem can significantly enhance the security posture of O-RAN. This dual approach ensures that both general compliance issues and specific dependency vulnerabilities are effectively covered, thereby

providing a robust and secure network infrastructure.

4.1.2 SAST

Static Application Security Testing (SAST) is crucial in the Implementation phase of the Secure Software Development Lifecycle (SSDLC), where developers write and integrate code. SAST tools analyze source code, bytecode, or binary code for security vulnerabilities without executing the program, allowing for the early detection and resolution of issues such as SQL injection, cross-site scripting, and buffer overflows. By automating code reviews and providing immediate feedback, SAST ensures consistent and thorough analysis, reducing the burden on manual reviews and fostering a culture of security within the development team. Seamlessly integrating with Integrated Development Environments (IDEs), build systems, and version control systems, SAST makes security checks a natural part of the development workflow, supporting continuous monitoring and immediate remediation of vulnerabilities. Additionally, SAST helps ensure compliance with internal security policies, industry standards, and regulatory requirements by identifying non-compliant code. This "shift-left" approach to security, where considerations are integrated early in the development process, reduces the overall cost and effort of fixing vulnerabilities and improves overall code quality and robustness. By incorporating SAST, developers enhance their secure coding skills, contributing to a more security-aware development team and ensuring the application is more secure upon release. Integrating SAST in the Implementation phase is essential for developing secure software, offering cost-effective, high-quality code, and reducing the risk of security breaches.

Table 12: Codacy Total Results from Non-RT RIC

Category	Severity		
	Critical	Medium	Low
Code Style	-	120	140
Security	50	19	1
Error prone	3	1	-

Codacy is a comprehensive automated code review and quality analysis tool that plays a vital role in maintaining code standards, particularly in the realm of security. Table 12 provides a detailed breakdown of coding issues detected by Codacy, categorized by severity and type. Focusing on the Security category, the table reveals a significant number of critical security issues, with 50 instances identified. Additionally, it highlights 19 medium severity issues and 1 low issue. This distribution emphasizes the importance of addressing security vulnerabilities in the codebase. The high count of critical security issues underscores the necessity for developers to prioritize security measures, ensuring robust and secure software development. By identifying and categorizing security issues, Codacy aids in mitigating potential risks and enhancing the overall security posture of applications. This proactive approach to security is essential in preventing breaches and maintaining the integrity of the software.

Table 13: Codacy Results Focus Security Category from Non-RT RIC

Severity	Details (Total Affects)(Amount of Files)	Solutions
Critical	<p>Command Injection:</p> <p>Subprocess call with `shell=True` seems safe but may be changed in the future, consider rewriting without shell</p> <p>(31)(6)</p>	<ul style="list-style-type: none"> - Use shlex.quote() function - Use environment variables

Critical	<p>Input Validation:</p> <p>Found `subprocess` function `check_output` with `shell=True`. This is dangerous because this call will spawn the command using a shell process.</p> <p>(8)(3)</p>	Use `shell=False` instead
Critical	<p>Input Validation:</p> <p>Found `subprocess` function `run` with `shell=True`. This is dangerous because this call will spawn the command using a shell process.</p> <p>(8)(3)</p>	Use `shell=False` instead
Critical	<p>Authentication:</p> <p>The application was found using the `requests` module without configuring a timeout value for connections.</p> <p>(1)(1)</p>	Remove `verify=False` argument or set `verify=True` to each `requests` call
Critical	<p>SSL:</p> <p>Call to requests with `verify=False` disabling SSL certificate checks, security issue.</p> <p>(1)(1)</p>	Re-enable certification validation by change `verify=True`
Critical	<p>Visibility:</p> <p>Certificate verification has been explicitly disabled.</p> <p>(1)(1)</p>	Re-enable certification validation by change `verify=True`
Medium	<p>Insecure Modules Libraries:</p>	Use shlex.quote() function

	Consider possible security implications associated with the subprocess module. (14)(6)	in subprocess module
Medium	SQL Injection: Possible SQL injection vector through string-based query construction. (3)(1)	Use Object Relational Mapping (ORM) [96] tools
Medium	Input Validation: Detected direct use of jinja2. If not done properly, this may bypass HTML escaping which opens up the application to cross-site scripting (XSS) vulnerabilities. (1)(1)	Prefer using the Flask method 'render_template()' and templates with a '.html' extension in order to prevent XSS.
Medium	Requests call without timeout (1)(1)	Input timeout for request
Low	Other: The application was found using `assert` in non-test code. (1)(1)	Replace them with either `if` conditions or `try/except` blocks.

Table 13 provides a detailed analysis of various security issues identified by Codacy, categorized by their severity and the total number of instances and files affected. Critical security issues are prominently highlighted, including command injection vulnerabilities from subprocess calls with `shell=True`, which impact 31 instances across 6 files. Input validation issues with the subprocess functions `check_output` and `run` using `shell=True` affect 8 instances each, across 3 files. Additionally, critical concerns are raised about the use of the `requests` module without configured timeouts and calls with `verify=False`, each impacting

1 instance in 1 file. Medium severity issues encompass security implications with the subprocess module, affecting 14 instances in 6 files, and potential SQL injection risks from string-based query construction, affecting 3 instances in 1 file. Moreover, direct use of `jinja2` without proper handling, posing cross-site scripting (XSS) vulnerabilities, and requests calls without timeout configurations, each affect 1 instance in 1 file. Finally, a low severity issue is noted with the use of `assert` in non-test code, impacting 1 instance in 1 file. This comprehensive analysis underscores the importance of addressing these security vulnerabilities to ensure robust and secure software development practices.

Table 14: Aikido Total Results from Non-RT RIC

Severity	Details (Amount of files)	Solutions
High	Container running as root can allow attacker to escalate attacks (1)	On your Pod, set <code>runAsNonRoot: true</code> and make sure <code>runAsUser:</code> is not set to 0, which is root
High	Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (6)	Move the secret out and use a tool to inject the secrets at run-time.
High	Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (4)	
High	Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (1)	
High	3 exposed secrets (3)	

Low	Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (1)	
Low	Detected a Generic API Key, potentially exposing access to various services and sensitive operations. (1)	
Medium	Filesystem for docker container should not be writeable (1)	On your Pod, set securityContext: readOnlyRootFilesystem: true
Medium	Potential file including attack via reading file (1)	Whitelisted or sanitized the file before input going into this function
Medium	Container processes can gain more privileges than its parent (1)	Set AllowPrivilegeEscalation to False
Medium	Default Kubernetes settings allow containers to eavesdrop on traffic. (1)	Define at least one PodSecurityPolicy (PSP) to prevent containers with NET_RAW capability from launching.

Aikido is a security tool designed to identify vulnerabilities and misconfigurations in containerized environments. Table 14 the results with Aikido revealed several security issues. High-severity findings included a container running as root, which could allow attackers to escalate attacks, and the presence of generic API keys in multiple files, potentially exposing sensitive operations. To mitigate these, it is recommended to configure the Pod with

`runAsNonRoot: true` and ensure `runAsUser` is not set to 0, as well as moving secrets out of git repositories and using tools like AWS Secrets Manager. Additionally, three exposed secrets were detected. There are 4 issues for Medium-severity: writable Docker container filesystems, potential file inclusion attacks, container processes gaining more privileges than their parent processes, and the default Kubernetes settings allowing containers to eavesdrop on network traffic. Mitigation strategies include setting filesystems to read-only, sanitizing input files, configuring `AllowPrivilegeEscalation` to false, and implementing PodSecurityPolicies (PSPs) to prevent the launch of containers with NET_RAW capabilities. These findings highlight the importance of implementing stringent security measures in containerized environments.

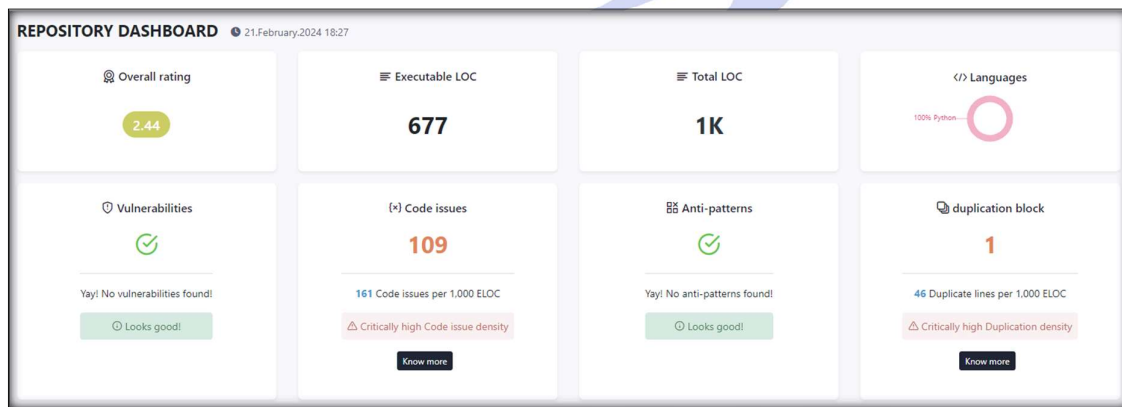
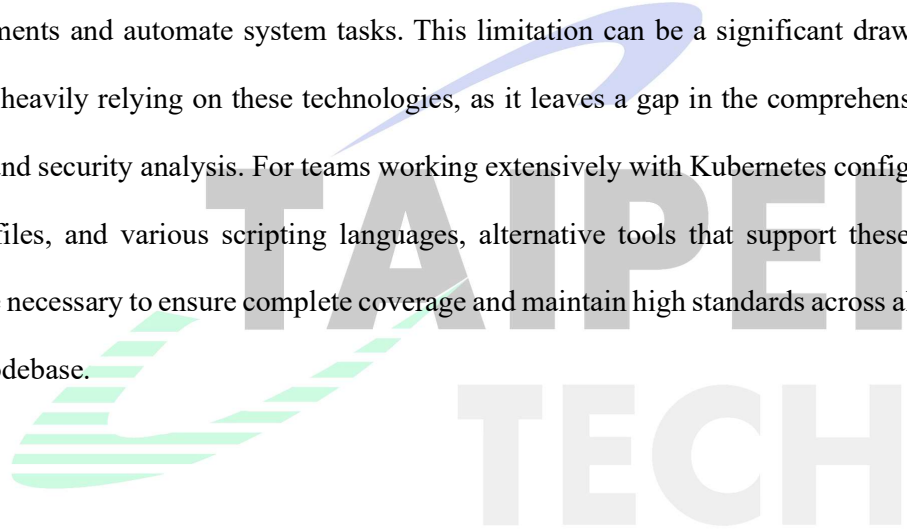


Figure 11: Embold Total Results from Non-RT RIC

The Embold static code analysis tool offers a comprehensive approach to enhancing software quality by analyzing source code across multiple dimensions such as code issues, design flaws, metrics, and duplication. Figure 11 Embold identified several key metrics for a Python-based project, including an overall rating of 2.44, indicating moderate code quality. The project contained 1,000 total lines of code (LOC) and 677 executable lines of code (ELOC). The tool detected 109 code issues, which translates to 161 issues per 1,000 ELOC, highlighting areas that require significant attention. Additionally, it found one duplication block, equating to 46 duplicate lines per 1,000 ELOC. Notably, no vulnerabilities or anti-patterns were found, suggesting robust security practices and adherence to design best

practices. Embold accommodates a broad spectrum of programming languages totaling 18, including C, C++, C#, TypeScript, Java, JavaScript, Python, PHP, and more. The latest update to Embold has improved CWE coverage for Java, introduced new checks for C++, and included various bug fixes and performance enhancements, ensuring it remains an effective tool for maintaining high code quality and security standards.

However, Embold has notable limitations that make it less suitable for projects involving Kubernetes (K8s), YAML files, shell scripts, and batch scripts. These components are critical for O-RAN system. The lack of support for these file types means that Embold cannot analyze or provide insights into the configurations and scripts that orchestrate containerized environments and automate system tasks. This limitation can be a significant drawback for projects heavily relying on these technologies, as it leaves a gap in the comprehensive code quality and security analysis. For teams working extensively with Kubernetes configurations, YAML files, and various scripting languages, alternative tools that support these formats might be necessary to ensure complete coverage and maintain high standards across all aspects of the codebase.



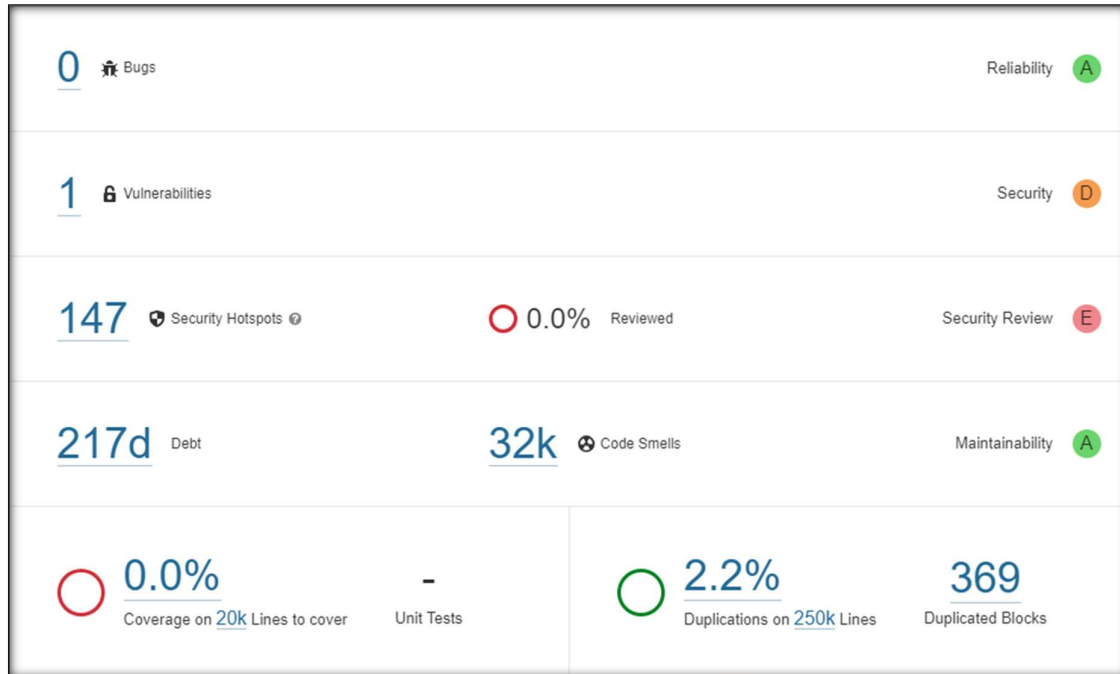


Figure 12: SonarQube Total Results from Non-RT RIC

The total results from SonarQube in Figure 12 reveal critical areas requiring attention, particularly concerning vulnerabilities and security hotspots. Despite the project's high reliability and maintainability ratings, the security dimension is notably lacking, with a low security rating (D) due to the presence of one significant vulnerability and 147 identified security hotspots. Alarming,ly, none of these security hotspots have undergone review, resulting in an abysmal security review rating (E). This oversight suggests a considerable risk, as unresolved vulnerabilities and unreviewed security hotspots could be potential entry points for malicious attacks, compromising the integrity and confidentiality of the system. Addressing these security issues is imperative. It necessitates a structured approach involving regular and thorough security reviews, prompt remediation of vulnerabilities, and continuous monitoring to prevent future risks. Enhancing security measures will ensure the robustness of the codebase, safeguarding it against potential threats and aligning it with best practices for secure software development.

Table 15: SonarQube Security Results from Non-RT RIC

Severity	Category	Details (Amount of Files)(CWE)	Solutions
Critical	Vulnerabilities	Enable server certificate validation on this SSL/TLS connection (1)(1)	Re-enable certification validation by change <code>`verify=True`</code>
High	Security Hotspots	Authentication: “password” detected here, make sure this is not a hard-coded credential. (15)(2)	Store the credentials in a file, database or cloud
Medium	Security Hotspots	Permission: Make sure setting capabilities is safe here. (59)(2)	Capabilities are high privileges, traditionally associated with superuser (root), thus make sure that the most restrictive and necessary capabilities are assigned.
Low	Security Hotspots	Encryption of Sensitive Data: Using http protocol is insecure. Use https instead. (24)(2)	Use https instead of http
Low	Security Hotspots	Log Injection: Make sure that this logger’s	Check the permissions, limits of size, format,

		configuration is safe. (20)(2)	configuration, and location are safe
Low	Security Hotspots	Others: Make sure publicly writable directories are used safely here. (29)(2)	Use a dedicated sub- folder with tightly controlled permissions or Use secure-by-design APIs to create temporary files

The results from SonarQube in Table 15 reveal several critical areas requiring attention to improve the project's overall security posture. The highest severity issue involves a critical vulnerability in one file, where the lack of server certificate validation may cause man-in-the-middle attacks, identified by one CWE. High severity concerns include hard-coded credentials found in 15 files, identified by 2 CWEs, posing significant security risks if the source code is compromised. Medium severity issues relate to capabilities and permissions settings in 59 files, identified by 2 CWEs, where improperly set permissions could grant unnecessary elevated privileges. Low severity issues include the use of HTTP instead of HTTPS for transmitting sensitive data in 24 files, identified by 2 CWEs, log injection vulnerabilities in 20 files due to unsafe logger configurations, identified by 2 CWEs, and publicly writable directories affecting 29 files, identified by 2 CWEs, which can be exploited if not properly controlled. Addressing these issues by adopting best practices such as enabling server certificate validation, securely storing credentials, managing permissions appropriately, using HTTPS, securing logger configurations, and controlling access to writable directories is crucial for mitigating these vulnerabilities and enhancing the security of the project.

To ensure the integrity and security of Open Radio Access Network (O-RAN) systems, it is essential to utilize effective code analysis and security tools. Codacy, Aikido, Embold, and SonarQube each offer unique features that contribute to this goal.

Codacy provides automated code review, supporting multiple programming languages and integrating seamlessly with CI/CD pipelines. It offers customizable rules and coding standards, along with detailed reports and metrics, making it an excellent tool for maintaining the code quality and security of O-RAN software components throughout the development process.

Aikido specializes in automated vulnerability scanning and patching, with strong integration capabilities for CI/CD tools such as Jenkins, GitHub Actions, and GitLab CI. It delivers real-time alerts and comprehensive security reports, which are crucial for preventing vulnerabilities in the disaggregated architecture of O-RAN, ensuring secure code from the outset.

Embold enhances code reliability and maintainability through AI-powered code analysis that detects design flaws, code smells, and bugs. With support for multiple languages and integration with various IDEs and CI/CD tools, Embold provides deep insights into code quality and architecture, benefiting the overall robustness of O-RAN software.

SonarQube is an open-source platform focused on continuous code quality inspection. It supports a wide range of programming languages and integrates well with CI/CD pipelines and development workflows. SonarQube's extensive plugins and strong community support make it a robust solution for maintaining code quality and security in O-RAN through continuous integration and delivery processes.

Each of these tools brings specific advantages to the development and maintenance of secure and reliable O-RAN, making them invaluable for ensuring the highest standards of software quality and security.

4.1.3 IAST

Interactive Application Security Testing (IAST) is essential in the Test and Deploy phases of the Secure Software Development Lifecycle (SSDLC), providing real-time analysis of an

application as it runs and combining elements of static and dynamic application security testing. In the Test phase, IAST tools monitor the application during functional and security testing, identifying vulnerabilities such as injection flaws and cross-site scripting by analyzing the application's behavior and data flow. IAST offers contextual insights, covering both server-side and client-side code, which traditional testing methods might miss. Detailed reports generated by IAST tools include the severity and location of vulnerabilities, along with remediation guidance, aiding developers and testers in addressing security issues effectively. The integration of IAST with existing testing frameworks and CI/CD pipelines ensures continuous security validation throughout the development lifecycle. During the Deploy phase, IAST performs pre-deployment security checks to ensure the application meets security standards, and post-deployment, it continues to monitor the application in the production environment, identifying new vulnerabilities that may arise. This continuous monitoring and feedback loop between development, testing, and operations teams foster ongoing security improvement. By identifying and addressing vulnerabilities before and after deployment, IAST mitigates security risks and enhances the overall security posture of the application, ensuring more secure software releases and reducing the likelihood of security breaches in production.

```

+ Nikto v2.5.0
+-----+
+ Target IP:      192.168.40.128
+ Target Hostname: 192.168.40.128
+ Target Port:    30091
+ Start Time:     2024-05-13 18:57:41 (GMT8)
+-----+
+ Server: nginx/1.25.1
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
+ See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /archive.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /168.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /168.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /site.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /40.jks: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192168.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.40.128.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /19216840128.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /128.tar.bz2: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.40.128.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /database.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /site.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /40.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.40.128.tar.lzma: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.40.128.war: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.cer: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /192.168.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /19216840.tgz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /19216840.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /archive.alz: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /dump.tar: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ /128.pem: Potentially interesting backup/cert file found. . See: https://cwe.mitre.org/data/definitions/530.html
+ 7948 requests: 0 error(s) and 162 item(s) reported on remote host
+ End Time:      2024-05-13 18:57:56 (GMT8) (15 seconds)
+-----+
+ 1 host(s) tested

```

Figure 13: Nikto Total Results from Non-RT RIC

Nikto is an open-source web server scanner that tests for various vulnerabilities, including outdated software and potential misconfigurations. Figure 13, several security issues were identified, including the absence of the X-Frame-Options and X-Content-Type-Options headers. The X-Frame-Options header prevents clickjacking by controlling whether a browser can render a page in a frame, while the X-Content-Type-Options header prevents MIME-type sniffing by ensuring the browser adheres to the declared content type. Additionally, the scan highlighted numerous backup files accessible on the server, categorized under CWE-530, indicating a risk of information exposure through unsecured backups. The scan reported 162 items, suggesting a significant exposure risk. To mitigate these issues, it is recommended to configure the X-Frame-Options and X-Content-Type-Options headers properly and secure or remove unnecessary backup files. Implementing these measures will enhance the web server's security posture by reducing the risk of exploitation through known vulnerabilities.

Table 16: OpenVAS Total Results from Non-RT RIC

Severity	Details (CVSS)(CVE)	Solutions
High	SSL/TLS: Report Vulnerable Cipher Suites for	The configuration of these

	<p>HTTP</p> <p>(7.5)</p> <p>(CVE-2016-2183,CVE-2016-6329, CVE-2020-12872)</p>	<p>services should be changed so that it does not accept the listed cipher suites anymore.</p>
Low	<p>TCP Timestamps Information Disclosure</p> <p>(2.6)(None)</p>	<p>To disable TCP timestamps</p>
Low	<p>Weak MAC Algorithm(s) Supported (SSH)</p> <p>(2.6)(None)</p>	<p>Disable the reported weak MAC algorithm(s).</p>
Low	<p>ICMP Timestamp Reply Information Disclosure</p> <p>(2.1)(CVE-1999-0524)</p>	<p>Block ICMP Timestamp request and reply</p>

OpenVAS stands as an open-source solution tailored for thorough vulnerability scanning and management purposes, used to identify security issues within networked systems by performing various checks on network protocols, operating systems, and applications. In Table 16, several vulnerabilities of varying severities were identified. A high-severity issue with SSL/TLS configurations reported the presence of vulnerable cipher suites for HTTP (CVSS 7.5), including CVE-2016-2183, CVE-2016-6329, and CVE-2020-12872, which can be mitigated by updating the configuration to reject these insecure cipher suites. Additionally, low-severity vulnerabilities were found: TCP Timestamps Information Disclosure (CVSS 2.6), resolvable by disabling TCP timestamps; weak MAC algorithms in SSH (CVSS 2.6), addressable by disabling the weak algorithms; and ICMP Timestamp Reply Information Disclosure (CVSS 2.1, CVE-1999-0524), which can be mitigated by protecting the remote host with a firewall. Addressing these issues will significantly enhance the security of the system by closing potential avenues for exploitation.

Nikto and OpenVAS are distinct tools with different scopes and functionalities, making direct comparisons challenging. Nikto primarily focuses on web server scanning, identifying vulnerabilities related to outdated software, dangerous files, and misconfigurations specific to

HTTP/S services. In contrast, OpenVAS offers comprehensive vulnerability assessments across entire networks, covering a broad range of vulnerabilities affecting different protocols, operating systems, and applications. Nikto generates straightforward reports focused on web server issues, whereas OpenVAS provides detailed reports with vulnerability descriptions, CVSS scores, and remediation recommendations, supporting integration with security management systems for comprehensive vulnerability management. In the context of O-RAN, Nikto is valuable for assessing the security of web interfaces used for management or configuration, quickly identifying and mitigating web-related vulnerabilities. OpenVAS, on the other hand, is essential for thorough and ongoing vulnerability management across the entire network of interconnected components typical of O-RAN environments. Its detailed reporting and broad coverage make it an ideal tool for maintaining the security of complex, multi-component O-RAN. Thus, while both tools offer unique advantages, OpenVAS is better suited for comprehensive security assessments and management in O-RAN, whereas Nikto excels in targeted web server vulnerability scanning.

4.1.4 DAST

Dynamic Application Security Testing (DAST) is critical in the Test and Deploy phases of the Secure Software Development Lifecycle (SSDLC). Unlike static analysis, DAST involves testing the application by simulating real-world attacks on a running application and identifying vulnerabilities during runtime. In the Test phase, DAST conducts black-box testing, probing for vulnerabilities such as SQL injection, cross-site scripting (XSS), and authentication flaws without prior knowledge of the codebase. By simulating real-world attacks, DAST ensures that vulnerabilities missed by static analysis are uncovered, providing comprehensive coverage through automated scans integrated into the CI/CD pipeline. This dynamic analysis reveals issues that manifest only during runtime, such as session management and data handling flaws, and generates detailed reports with remediation steps

for developers and testers. In the Deploy phase, DAST performs pre-deployment security validation to ensure all vulnerabilities have been mitigated and continues post-deployment monitoring to identify new vulnerabilities that may emerge. This continuous security assessment maintains the application's integrity, mitigating risks and ensuring robustness against potential attacks. By integrating DAST into the SSDLC, organizations can enhance their application's security posture, ensuring secure software releases and reducing the likelihood of security breaches in production.

Table 17: Nessus Total Results from Non-RT RIC

Severity	Details (CVSS)(CVE)	Solutions
High	SSL Medium Strength Cipher Suites Supported: SWEET32 (7.5)(CVE-2016-2183)	Reconfigure the affected application if possible to avoid use of medium strength ciphers.
Medium	SSL Certificate Cannot Be Trusted (6.5)(None)	Purchase or generate a proper SSL certificate for this service.
Low	ICMP Timestamp Request Remote Date Disclosure (2.1)(CVE-1999-0524)	Filter out the ICMP timestamp requests, and the outgoing ICMP timestamp replies.

Nessus, a widely used vulnerability assessment tool, identifies and helps remediate security vulnerabilities in networked systems by performing comprehensive scans to detect misconfigurations, missing patches, and potential exploits. The results in Table 17 highlight vulnerabilities categorized by severity: High, Medium, and Low, each with detailed descriptions, CVSS scores, and remediation suggestions. The High severity vulnerability

involves the support of SSL medium strength cipher suites (SWEET32), with a CVSS score of 7.5 (CVE-2016-2183). The recommended solution is to reconfigure the application to avoid using medium-strength ciphers. The Medium severity vulnerability pertains to an untrusted SSL certificate, with a CVSS score of 6.5, and the suggested fix is to obtain a proper SSL certificate from a trusted certificate authority. The Low severity vulnerability involves ICMP timestamp request remote date disclosure, with a CVSS score of 2.1 (CVE-1999-0524), and can be mitigated by filtering out ICMP timestamp requests and replies. Resolving these vulnerabilities will strengthen the system's security stance, providing better protection against potential threats.

Workload Assessment						
Namespace	Resource	Vulnerabilities				
		C	H	M	L	U
nonrtic	Deployment/topology	15	53	26	6	
nonrtic	Deployment/a1-sim-std-0		11	27	2	
nonrtic	StatefulSet/dmaapmediatorservice	2	21	29	13	
nonrtic	StatefulSet/informationsservice	3	54	58	76	
nonrtic	Deployment/oran-nonrtic-kong	2	23	28	6	
nonrtic	Deployment/oru-app	17	112	125	133	8
nonrtic	Deployment/a1-sim-std2-0		11	27	2	
nonrtic	Deployment/controlpanel	2	11	22	4	
nonrtic	Deployment/oran-nonrtic-odu-app	6	31	41	15	6
nonrtic	Deployment/nonrticgateway	2	47	50	76	
nonrtic	StatefulSet/helmmanager	18	181	162	93	
nonrtic	Deployment/a1-sim-osc-0		11	27	2	
nonrtic	Deployment/oran-nonrtic-odu-app-ics-version	6	31	41	15	6
nonrtic	Deployment/a1-sim-std2-1		11	27	2	
nonrtic	Deployment/rappcatalogueservice	2	47	55	76	
nonrtic	StatefulSet/dmaapadapterservice	3	55	62	77	
nonrtic	Deployment/a1-sim-osc-1		11	27	2	
nonrtic	Deployment/a1-sim-std-1		11	27	2	

Severities: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

Figure 14: Trivy Total Results from Non-RT RIC

Trivy is a tool for scanning security flaws in containers, Kubernetes clusters, and other artifacts, integrating seamlessly into CI/CD pipelines for continuous security monitoring. The results presented in Figure 14 identify a range of vulnerabilities across various deployments and stateful sets, categorized by severity levels: Critical, High, Medium, Low, and Unknown. This scan highlights significant security risks, particularly due to the presence of multiple

critical and high-severity vulnerabilities, necessitating immediate action to prevent exploitation. Additionally, the detection process has identified duplicate vulnerabilities within the same libraries, complicating the analysis and prioritization of these issues. The high number of medium and low-severity vulnerabilities further underscores the need for ongoing monitoring and timely updates to maintain a strong security posture. Despite the added complexity from duplicate vulnerabilities, it is essential to address each one systematically to enhance the system's security and stability, thereby reducing the risk of exploitation and improving overall resilience against potential threats.

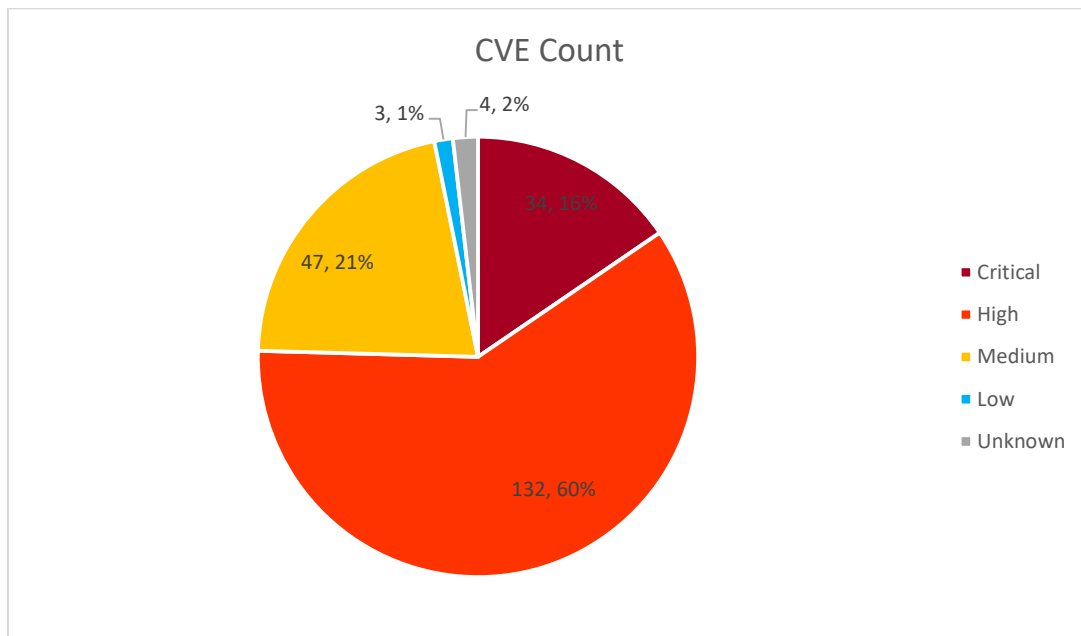


Figure 15: Trivy CVE Results Categorize by Severity from Non-RT RIC

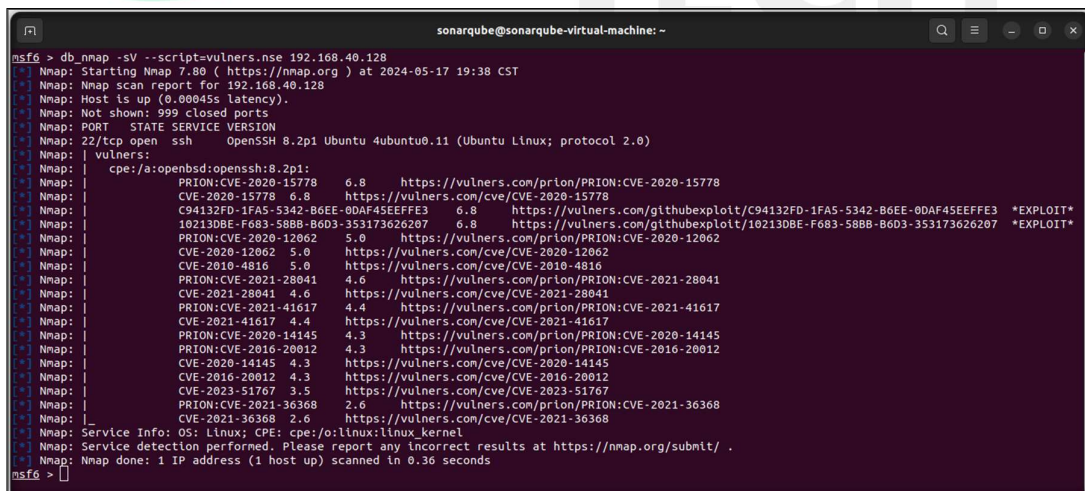
Figure 15 highlights a high concentration of vulnerabilities after duplicate entries were removed, with 60% categorized as High severity and 16% as Critical. This indicates an urgent need for remediation to maintain system security and stability. Medium severity CVEs account for 21%, requiring planned mitigation, while Low and Unknown severities make up only 3%, necessitating regular monitoring. A significant number of Critical vulnerabilities are linked to buffer overflow issues, especially in the libc6 library, including CVE-2021-33574, CVE-2021-35942, CVE-2022-23218, and CVE-2022-23219. These buffer overflow vulnerabilities

pose a serious risk, potentially leading to arbitrary code execution and system compromise. To improve the accuracy of vulnerability assessments, it is essential to use third-party tools to remove duplicate entries from raw data. These tools ensure the dataset is accurate and reliable by identifying and eliminating redundant data, thus enhancing data integrity. This results in more precise vulnerability analyses and more effective risk management strategies. Moreover, numerous vulnerabilities still require analysis, highlighting the necessity for continuous vigilance and assessment. By consistently updating and refining the dataset, organizations can better prioritize remediation efforts and effectively address potential security gaps.

Nessus and Trivy are both valuable tools for securing O-RAN, each with distinct strengths and some limitations. Nessus offers comprehensive vulnerability scanning across network devices, operating systems, and applications, providing detailed reports with CVSS scores, descriptions, and remediation recommendations. Its ability to perform configuration audits and detect malware makes it essential for ensuring network security and compliance in O-RAN. Trivy, on the other hand, specializes in container security, focusing on scanning container images and application dependencies for vulnerabilities. It integrates seamlessly with Kubernetes, making it highly relevant for O-RAN that use containerized microservices and Kubernetes clusters. Trivy's real-time scanning capabilities support continuous integration and deployment workflows, crucial for the dynamic nature of O-RAN environments. However, Trivy's tendency to produce duplicate results can complicate analysis, making it more challenging to prioritize and address vulnerabilities effectively. While Nessus offers broad coverage and detailed analysis suitable for thorough security audits, Trivy provides quick and efficient scanning for modern, containerized deployments but requires careful management of duplicate findings. Using both tools together can provide a comprehensive security solution, leveraging Nessus for wide-ranging vulnerability assessments and Trivy for specialized container and Kubernetes security.

4.1.5 Pentest

Penetration testing, or ethical hacking, is crucial in the Maintenance phase of the Secure Software Development Lifecycle (SSDLC), ensuring the ongoing security and integrity of software post-deployment. This phase focuses on tasks such as applying patches, updating components, and responding to emerging threats. Penetration testing simulates real-world attacks to identify and mitigate vulnerabilities, thereby preventing exploitation by malicious actors. It identifies new vulnerabilities introduced through updates, verifies the effectiveness of patches, assesses existing security controls, and detects configuration issues. By simulating advanced threats, penetration testing helps organizations understand their defenses' resilience against modern attacks, informing security strategy adjustments. Additionally, regular penetration testing ensures compliance with regulatory frameworks and industry standards, avoiding penalties. The proactive approach of penetration testing enables organizations to manage risks effectively, continuously improve security measures, and enhance incident response strategies. Thus, penetration testing in the Maintenance phase is essential for maintaining a robust security posture, ensuring software reliability, and safeguarding against evolving threats.



```
nsf6 > db_nmap -sv --script=vulners.nse 192.168.40.128
[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2024-05-17 19:38 CST
[*] Nmap: Nmap scan report for 192.168.40.128
[*] Nmap: Host is up (0.000455 latency).
[*] Nmap: Not shown: 999 closed ports
[*] Nmap: PORT      STATE SERVICE VERSION
[*] Nmap: 22/tcp open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
[*] Nmap: | vulners:
[*] Nmap: | cpe:/a:openssh:openssh:8.2p1:
[*] Nmap: | PRION:CVE-2020-15778 6.8 https://vulners.com/prion/PRION:CVE-2020-15778
[*] Nmap: | CVE-2020-15778 6.8 https://vulners.com/cve/CVE-2020-15778
[*] Nmap: | C94132FD-1FA5-5342-86EE-0DAF45EEFFE3 6.8 https://vulners.com/githubexploit/C94132FD-1FA5-5342-86EE-0DAF45EEFFE3 *EXPLOIT*
[*] Nmap: | 102130BE-F683-58BB-B6D3-353173626207 6.8 https://vulners.com/githubexploit/102130BE-F683-58BB-B6D3-353173626207 *EXPLOIT*
[*] Nmap: | PRION:CVE-2020-12062 5.0 https://vulners.com/prion/PRION:CVE-2020-12062
[*] Nmap: | CVE-2020-12062 5.0 https://vulners.com/cve/CVE-2020-12062
[*] Nmap: | CVE-2010-4816 5.0 https://vulners.com/cve/CVE-2010-4816
[*] Nmap: | PRION:CVE-2021-28041 4.6 https://vulners.com/prion/PRION:CVE-2021-28041
[*] Nmap: | CVE-2021-28041 4.6 https://vulners.com/cve/CVE-2021-28041
[*] Nmap: | PRION:CVE-2021-41617 4.4 https://vulners.com/prion/PRION:CVE-2021-41617
[*] Nmap: | CVE-2021-41617 4.4 https://vulners.com/cve/CVE-2021-41617
[*] Nmap: | PRION:CVE-2020-14145 4.3 https://vulners.com/prion/PRION:CVE-2020-14145
[*] Nmap: | PRION:CVE-2016-20012 4.3 https://vulners.com/prion/PRION:CVE-2016-20012
[*] Nmap: | CVE-2020-14145 4.3 https://vulners.com/cve/CVE-2020-14145
[*] Nmap: | CVE-2016-20012 4.3 https://vulners.com/cve/CVE-2016-20012
[*] Nmap: | CVE-2023-51767 3.5 https://vulners.com/cve/CVE-2023-51767
[*] Nmap: | PRION:CVE-2021-36368 2.6 https://vulners.com/prion/PRION:CVE-2021-36368
[*] Nmap: | CVE-2021-36368 2.6 https://vulners.com/cve/CVE-2021-36368
[*] Nmap: Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 0.36 seconds
nsf6 >
```

Figure 16: Metasploit with Nmap Integration Total Results from Non-RT RIC

Metasploit, an open-source penetration testing framework, integrates with Nmap to

enhance vulnerability assessment by simulating real-world attacks and identifying security weaknesses. Nmap is utilized to discover hosts, open ports, and services on a network, providing detailed information about each service. The results in Figure 16 identified one open port, TCP port 22, which is running the SSH service OpenSSH 8.2p1 on an Ubuntu Linux system. Several vulnerabilities were detected in the SSH service, categorized with varying levels of severity. These vulnerabilities indicate potential security weaknesses that need to be addressed to ensure the system's security. The integration of Nmap's scanning capabilities within the Metasploit framework allows for a comprehensive analysis of the network, identifying critical security issues that can be mitigated to enhance the overall security posture.

Table 18: Metasploit with Nmap Integration Results Details and Solutions

CVE (CVSS)	Details	Solutions
CVE-2020-15778 (6.8)	A vulnerability in OpenSSH that allows an attacker to potentially execute arbitrary code.	Upgrade to the latest version of OpenSSH. Apply security patches provided by your OS vendor.
CVE-2020-12062 (5.0)	A vulnerability in OpenSSH that could allow an attacker to cause a denial of service.	
CVE-2010-4816 (5.0)	A vulnerability in OpenSSH related to improper handling of network connections.	
CVE-2021-28041 (4.6)	A vulnerability in OpenSSH that allows unauthorized users to bypass certain access controls.	
CVE-2021-41617	A vulnerability in OpenSSH that	

(4.4)	could lead to unintended information disclosure.	
CVE-2020-14145 (4.3)	A vulnerability in OpenSSH affecting the handling of certain network packets.	
CVE-2016-20012 (4.3)	A vulnerability in OpenSSH that could lead to a potential denial of service.	
CVE-2023-51767 (3.5)	A vulnerability in OpenSSH that affects certain encryption protocols.	
CVE-2021-36368 (2.6)	A minor vulnerability in OpenSSH that could lead to information leakage under specific conditions.	

Table 18 provides an overview of vulnerabilities detected in OpenSSH 8.2p1, listing CVE identifiers, severity scores, descriptions, and mitigation solutions. Notable vulnerabilities include CVE-2020-15778 (score 6.8), which allows arbitrary code execution, and CVE-2020-12062 (score 5.0), which can cause a denial of service. Other issues involve unauthorized access control bypass and unintended information disclosure. The primary solution across all vulnerabilities is to upgrade to the latest OpenSSH version and apply relevant security patches. This proactive approach is crucial for minimizing security risks and maintaining system integrity.

```

Nodes
-----
| TYPE | LOCATION |
-----
| Node/Master | 192.168.40.128 |
-----

Detected Services
-----
| SERVICE | LOCATION | DESCRIPTION |
-----
| Kubelet API | 192.168.40.128:10250 | The Kubelet is the main component in every node, all pod operations goes through the kubelet |
-----
| Etcd | 192.168.40.128:2379 | Etcd is a DB that stores cluster's data, it contains configuration and current state information, and might contain secrets |
-----
| API Server | 192.168.40.128:6443 | The API server is in charge of all operations on the cluster. |
-----

Vulnerabilities
For further information about a vulnerability, search its ID in:
https://avd.aquasec.com/
-----
| ID | LOCATION | MITRE CATEGORY | VULNERABILITY | DESCRIPTION | EVIDENCE |
-----
| KHV002 | 192.168.40.128:6443 | Initial Access // Exposed sensitive interfaces | K8s Version Disclosure | The kubernetes version could be obtained from the /version endpoint | v1.22.17 |
-----

root@ubuntu:~/kube-hunter#

```

Figure 17: Kube-hunter Total Results from Non-RT RIC

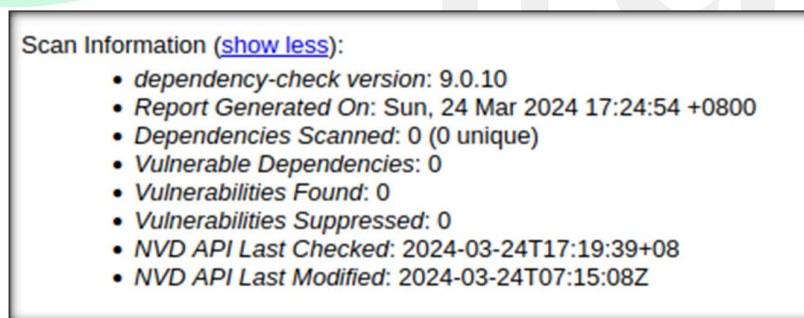
Kube-Hunter is a security tool designed to scan Kubernetes clusters, identifying potential vulnerabilities in their configurations and components. Figure 17 identified a node operating as both node and master at IP address 192.168.40.128. Key services detected include the Kubelet API on port 10250, responsible for managing pod operations; Etcd on port 2379, which stores cluster data, configurations, and secrets; and the API Server on port 6443, which manages all cluster operations. A significant vulnerability identified, labeled KHV002, involves the Kubernetes version disclosure via the /version endpoint on the API server, categorized under Initial Access. This vulnerability, evidenced by the version v1.22.17, could be exploited by attackers to target specific weaknesses in that version. To mitigate these risks, it is recommended to restrict API server access, hide version information, regularly update Kubernetes components, and continuously monitor and audit the cluster for suspicious activities. These measures are essential to enhancing the security posture of Kubernetes deployments.

The comparison of Metasploit with Nmap integration and Kube-Hunter highlights their key security features and relevance to O-RAN. Metasploit combined with Nmap offers extensive vulnerability scanning by leveraging Nmap's network discovery and service

identification with Metasploit’s exploit framework, enabling automated exploitation and detailed vulnerability reporting. This tool combination is particularly relevant for O-RAN due to the need for both broad network scans and targeted penetration testing. It provides a holistic security assessment, addressing network-level vulnerabilities and application-level security. In contrast, Kube-Hunter, designed specifically for Kubernetes environments, performs targeted scans to identify misconfigurations and vulnerabilities in Kubernetes components. Its relevance to O-RAN lies in the adoption of containerized architectures within O-RAN, where Kubernetes often orchestrates network functions. Kube-Hunter's focused approach ensures the security of the orchestration layer in cloud-native deployments. Together, these tools offer a comprehensive security assessment for O-RAN, with Metasploit and Nmap addressing broad and deep security evaluations, and Kube-Hunter ensuring the integrity of the container orchestration layer.

4.2 Near-RT RIC

4.2.1 SCA



```
Scan Information (show less):

- dependency-check version: 9.0.10
- Report Generated On: Sun, 24 Mar 2024 17:24:54 +0800
- Dependencies Scanned: 0 (0 unique)
- Vulnerable Dependencies: 0
- Vulnerabilities Found: 0
- Vulnerabilities Suppressed: 0
- NVD API Last Checked: 2024-03-24T17:19:39+08
- NVD API Last Modified: 2024-03-24T07:15:08Z

```

Figure 18: OWASP Dependency Check Total Results from Near-RT RIC

Figure 18 reported zero dependencies scanned and no vulnerabilities detected, as the directory lacked the configuration files needed for the tool to identify and analyze dependencies. This outcome is primarily due to the absence of necessary configuration files,

such as `package.json` or `pom.xml`, which define dependencies for the project. Similarly, Mend.io found no vulnerabilities, as it also relies on these configuration files to perform its analysis. This absence highlights a significant difference between the Near-RT RIC and Non-RT RIC directories, with the latter containing various tools and simulators that include their dependencies.

The substantial difference in the results between the Non-RT Ric and Near-RT Ric scans is due to the presence of various tools and a simulator in the Non-RT Ric directory, which are absent in the Near-RT Ric directory. The Non-RT Ric directory includes essential configuration files, such as `package.json` for JavaScript projects, which define the dependencies necessary for a thorough scan. These tools and simulators inherently possess their own dependencies, enabling the scan to detect and analyze potential vulnerabilities. In contrast, the Near-RT Ric directory lacks these tools and configuration files, resulting in no dependencies being identified or assessed, leading to a report of zero vulnerabilities. This highlights the importance of ensuring that all necessary configuration files and tools that define dependencies are present in the directory being scanned. Without these components, the dependency-check tool cannot perform an effective analysis, leading to incomplete or inaccurate scan results.

4.2.2 SAST

The Codacy scan of the Near-RT RIC identified only one low-severity issue concerning the use of insecure modules or libraries. Specifically, the Dockerfile in the `ci` directory was flagged for not using the `--no-install-recommends` option during package installation. This omission can lead to the inclusion of additional, potentially unnecessary packages that were not explicitly wanted, which can introduce security vulnerabilities, increase the Docker image size, and complicate maintenance. To mitigate this risk, it is recommended to modify the Dockerfile to include the `--no-install-recommends` option, ensuring that only the specified

packages are installed. For example, the command `apt-get install -y some-package` should be changed to `apt-get install -y --no-install-recommends some-package`. This adjustment aligns with best practices for creating leaner, more secure Docker images and enhances overall system efficiency and security by reducing unnecessary dependencies.

Table 19: Aikido Total Results from Near-RT RIC

Severity	Details (Amount of files)	Solutions
High	Container running as root can allow attacker to escalate attacks (3)	On your Pod, set runAsNonRoot: true and make sure runAsUser: is not set to 0, which is root
High	Automatic upgrades or base Docker images can lead to supply chain attacks. (1)	It's recommended to pin the version of base images inside of Docker containers.
High	1 exposed secrets (1)	Move the secret out and use a tool to inject the secrets at run-time.
Medium	Docker container runs as default root user (2)	Add 'USER username' to the end of your file.
Medium	Container processes can gain more privileges than its parent (8)	set AllowPrivilegeEscalation to False
Medium	Filesystem for docker container should not be writeable (8)	On your Pod, set securityContext: readOnlyRootFilesystem:

		true
Medium	Default security context allows pods to access host system. (7)	Define a Security Context based that gives the minimum amount of access required for this workload.
Medium	Default Kubernetes settings allow containers to eavesdrop on traffic. (8)	Define at least one PodSecurityPolicy (PSP) to prevent containers with NET_RAW capability from launching.

The Aikido results in Table 19 reveal significant security vulnerabilities primarily associated with privilege escalation, improper user permissions, and insecure default configurations in Docker and Kubernetes environments. High severity issues include containers running as root, automatic upgrades leading to supply chain attacks, and exposed secrets, each requiring specific configurations such as setting `runAsNonRoot: true`, pinning base image versions, and managing secrets at runtime. Medium severity issues encompass the use of default root users, containers gaining excessive privileges, writable filesystems, and default security contexts allowing undue access to host systems and network traffic. Addressing these vulnerabilities involves implementing security best practices such as defining non-root users, setting `AllowPrivilegeEscalation` to `false`, making filesystems read-only, and establishing stringent `PodSecurityPolicies`. These measures are crucial for enhancing the security posture of containerized applications, thereby mitigating potential risks and vulnerabilities.

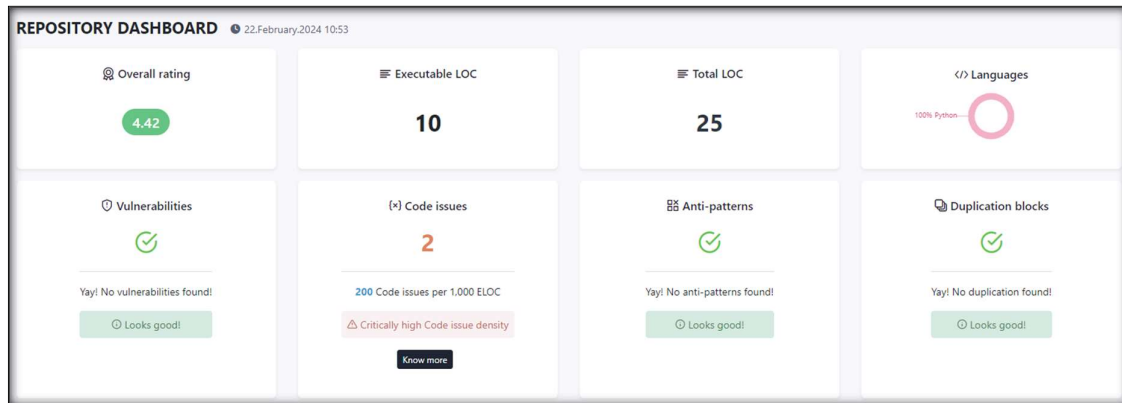


Figure 19: Embold Total Results from Near-RT RIC

The Embold analysis of the Near-RT RIC repository indicates strong overall code quality. Figure 19 is reflected in a high rating of 4.42. The repository consists of 25 lines of Python code, with 10 executable lines, making it relatively small and potentially easier to manage. Importantly, the scan found no security vulnerabilities, anti-patterns, or code duplication, all of which are positive indicators of the code's robustness and maintainability. However, the analysis did identify two code issues, resulting in a high issue density due to the small size of the codebase. Addressing these issues could further improve the code's quality.

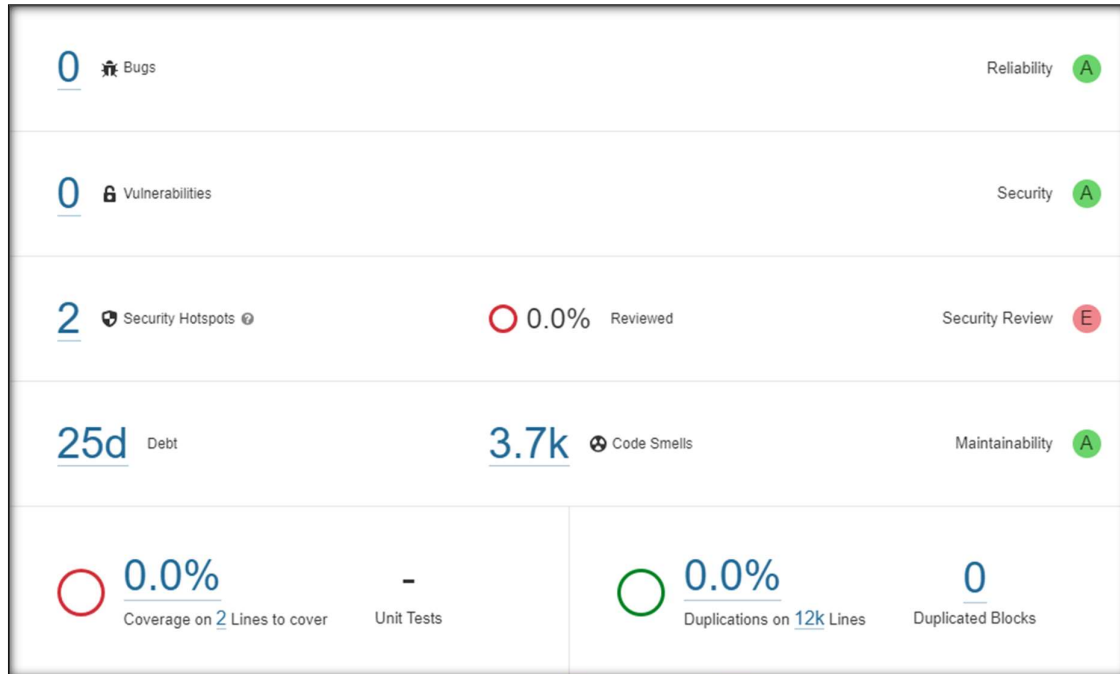


Figure 20: SonarQube Total Results from Near-RT RIC

The SonarQube in Figure 20 reveals high reliability and security, with no bugs or vulnerabilities detected, earning the code an A rating in both categories. However, the presence of two security hotspots highlights the need for further review, specifically addressing the insecure use of the HTTP protocol for sensitive data by switching to HTTPS. This concern aligns with the specifications outlined by WG11 [11], which mandates stringent security measures to ensure the confidentiality, integrity, and availability of data transmitted within O-RAN. Therefore, before implementing the switch to HTTPS, it is crucial to confirm compliance with WG11's specifications, ensuring that the transition adheres to the established security protocols and does not introduce new vulnerabilities. The codebase is considered highly maintainable, despite 3,700 code smells. The analysis also shows no code duplication, which is a positive indicator of maintainability. Nonetheless, the lack of unit tests and 0% code coverage indicate a need for improved testing practices to ensure the robustness and reliability of the code.

4.2.3 IAST

The absence of results from the Nikto scan can be attributed to the lack of web services within this environment. Nikto is a web server scanner designed to identify vulnerabilities and misconfigurations in web servers by checking for outdated software, insecure files, and other common issues. However, since the Near-RT RIC does not host any web servers or web services, there were no HTTP(S) endpoints for Nikto to scan. Consequently, this resulted in no findings. This underscores the importance of selecting appropriate tools based on the specific context and components present in the environment being analyzed. In this case, using Nikto was ineffective because there were no web services present.

The results from OpenVAS in Table 20 reveal several vulnerabilities, consistent with similar results in Non-RT RIC, that need addressing to enhance system security. A high severity issue involves the use of vulnerable cipher suites for HTTP, with specific vulnerabilities identified as CVE-2016-2183, CVE-2016-6329, and CVE-2020-12872, necessitating the reconfiguration of services to use secure cipher suites. Low severity issues include TCP timestamps information disclosure, weak MAC algorithms in SSH, and ICMP timestamp reply information disclosure. These vulnerabilities can be mitigated by disabling TCP timestamps, weak MAC algorithms, and protecting the system against ICMP timestamp requests and replies through firewall configurations. Addressing these issues is essential to strengthen the system's security and protect against potential exploits.

Table 20: OpenVAS Total Results from Near-RT RIC

Severity	Details (CVSS)(CVE)	Solutions
High	SSL/TLS: Report Vulnerable Cipher Suites for HTTP (7.5)	The configuration of this services should be changed so that it does not accept the

	(CVE-2016-2183,CVE-2016-6329, CVE-2020-12872)	listed cipher suites anymore.
Low	TCP Timestamps Information Disclosure (2.6)(None)	To disable TCP timestamps
Low	Weak MAC Algorithm(s) Supported (SSH) (2.6)(None)	Disable the reported weak MAC algorithm(s).
Low	ICMP Timestamp Reply Information Disclosure (2.1)(CVE-1999-0524)	Protect the remote host by a firewall

4.2.4 DAST

The Nessus scan results from Table 21 reveal several security vulnerabilities that are consistent with those found in the Non-RT RIC. A high severity issue identified is the support of medium strength cipher suites, specifically SWEET32, which poses a significant risk and requires reconfiguration to use stronger ciphers. Additionally, a medium severity issue is the presence of an untrusted SSL certificate, which can facilitate man-in-the-middle attacks, necessitating the acquisition of a proper SSL certificate from a trusted authority. A low severity issue involves ICMP timestamp request remote date disclosure, which can expose the system's time and potentially aid attackers. Mitigating this issue involves filtering out ICMP timestamp requests and replies. Addressing these vulnerabilities is essential to improving the security posture and protecting against potential exploits.

Table 21: Nessus Total Results from Near-RT RIC

Severity	Details (CVSS)(CVE)	Solutions
High	SSL Medium Strength Cipher Suites Supported: SWEET32	Reconfigure the affected application if possible to

	(7.5)(CVE-2016-2183)	avoid use of medium strength ciphers.
Medium	SSL Certificate Cannot Be Trusted (6.5)(None)	Purchase or generate a proper SSL certificate for this service.
Low	ICMP Timestamp Request Remote Date Disclosure (2.1)(CVE-1999-0524)	Filter out the ICMP timestamp requests (13), and the outgoing ICMP timestamp replies (14).

Figure 21 highlights a significant number of vulnerabilities across various deployments, with particular concern for the critical and high-severity vulnerabilities found in key deployments such as ‘ricplt-dbaas-server’, ‘prometheus-alertmanager’, ‘ricplt-olmediator’, and ‘appmgr’. These critical and high-severity issues require urgent remediation to mitigate major security risks. Additionally, the high number of medium and low-severity vulnerabilities underscores the need for comprehensive security upgrades throughout the system. Similar findings were observed in the Non-RT RIC, where duplicate vulnerabilities in the same libraries were identified, complicating the analysis and prioritization. Addressing these vulnerabilities is essential to improve the security posture of the Near-RT RIC environment, ensuring strong protection against potential exploits and maintaining the system’s overall integrity.

Workload Assessment						
Namespace	Resource	Vulnerabilities				
		C	H	M	L	U
ricplt	Deployment/deployment-ricplt-e2term-alpha		2	185	93	
ricplt	Deployment/r4-infrastructure-kong	9	152	102	9	
ricplt	StatefulSet/statefulset-ricplt-dbaas-server	4	57	47	1	
ricplt	Deployment/r4-infrastructure-prometheus-alertmanager	6	122	74	2	
ricplt	Deployment/deployment-ricplt-o1mediator	6	146	780	255	
ricplt	Deployment/deployment-ricplt-alarmmanager	6	95	103	51	
ricplt	Deployment/deployment-ricplt-e2mgr	3	45	209	94	
ricplt	Deployment/deployment-ricplt-submgr	2	41	210	94	
ricplt	Deployment/deployment-ricplt-a1mediator	1	59	712	250	
ricplt	Deployment/deployment-ricplt-vespamgr	12	198	193	81	
ricplt	Deployment/r4-infrastructure-prometheus-server	6	116	82	4	
ricplt	Deployment/deployment-ricplt-appmgr	4	61	102	67	
ricplt	Deployment/deployment-ricplt-rtmgr	3	50	214	94	

Severities: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

Figure 21: Trivy Total Results from Near-RT RIC

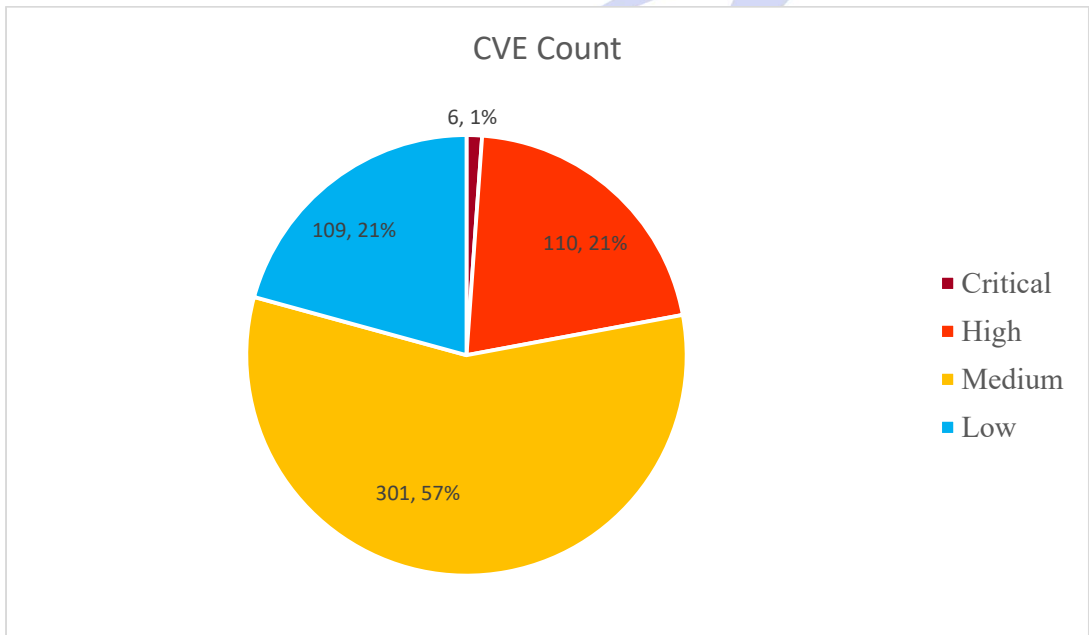


Figure 22: Trivy CVE Results Categorize by Severity from Near-RT RIC

After cleaning some data, Figure 22 reveals that medium severity vulnerabilities dominate, with 301 CVEs, representing 57% of the total. High and low severity vulnerabilities are evenly split, each comprising 21% of the total, with 110 high severity CVEs and 109 low severity CVEs. Critical vulnerabilities are the rarest, with only 6 CVEs, making up 1% of the total. Significantly, these critical vulnerabilities match those identified in the Non-RT RIC,

mainly involving buffer overflow issues. This distribution indicates a pressing need to prioritize remediation efforts on medium and high severity vulnerabilities to bolster system security, while also addressing the critical buffer overflow vulnerabilities due to their severe risk potential.

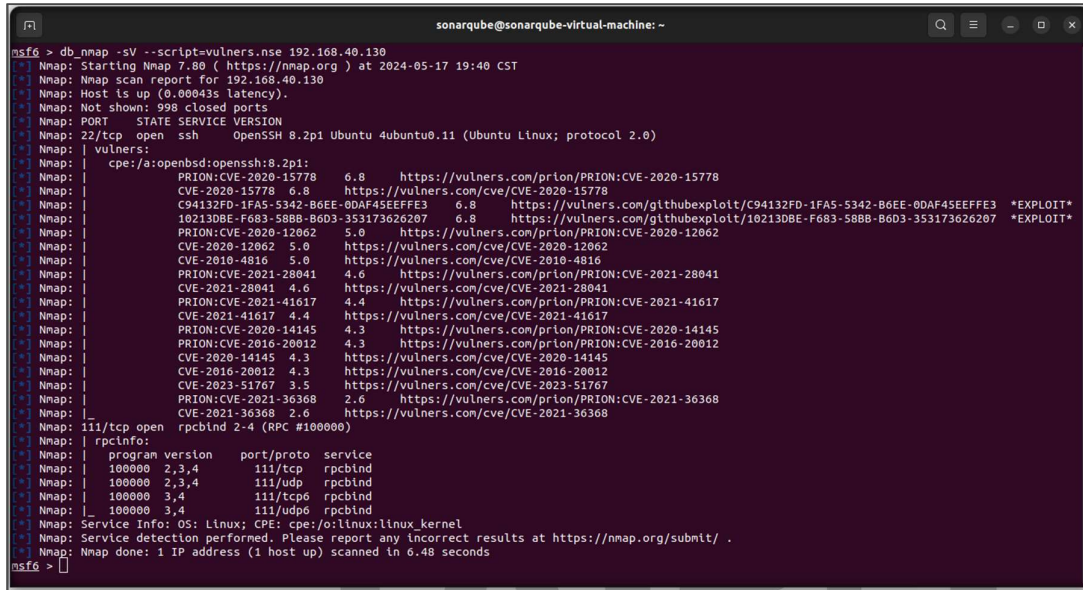
Trivy's results reveal several challenges and limitations in managing vulnerabilities. One issue is the repetition of CVEs across multiple deployments, making analysis more complex as each CVE must be individually reviewed. As a free tool, Trivy lacks the advanced features available in paid solutions, which limits the depth and comprehensiveness of its reports. It also lacks advanced filtering, reporting, and integration capabilities, making thorough analysis in complex environments difficult.

To address these challenges, third-party tools with advanced features can provide better reporting, filtering, and integration capabilities, facilitating easier management and remediation of vulnerabilities. Tools that consolidate and deduplicate vulnerabilities can help prioritize and address critical issues more effectively. Investing in tools that offer detailed and customizable reports can provide insights into critical vulnerabilities and suggest specific remediation steps, thereby enhancing the overall security posture. While Trivy is useful for identifying vulnerabilities, its limitations highlight the need for more advanced third-party solutions to improve the security of the Near-RT RIC environment.

4.2.5 Pentest

The result from Metasploit in Figure 23 identified several critical vulnerabilities similar to those found in the Non-RT RIC environment. Notably, the scan revealed that port 111/tcp, associated with the RPCBind service, is open but did not show specific vulnerabilities linked to it. While no immediate vulnerabilities were detected for RPCBind, the presence of an open port remains a security concern, as it could potentially be exploited if not properly secured. It is crucial to review and harden the configuration of the RPCBind service to minimize any

risks. Regular monitoring and vulnerability scanning should be implemented to ensure that the service remains secure. Addressing these issues and enforcing consistent security policies across all environments are essential steps to bolster the overall security posture of the Near-RT RIC, preventing possible exploits through open ports.



```
msf6 > db_nmap -sV --script=vulners.nse 192.168.40.130
* Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2024-05-17 19:40 CST
* Nmap: Nmap scan report for 192.168.40.130
* Nmap: Host is up (0.000435 latency).
* Nmap: Not shown: 998 closed ports
* Nmap: PORT      STATE SERVICE
* Nmap: 22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
* Nmap: | vulners:
* Nmap: |   cpe:/a:openbsd:openssh:8.2p1:
* Nmap: |   PRION:CVE-2020-15778 6.8 https://vulners.com/prion/PRION:CVE-2020-15778
* Nmap: |   CVE-2020-15778 6.8 https://vulners.com/cve/CVE-2020-15778
* Nmap: |   C94132FD-1FA5-5342-B6EE-0DAF45EEFFE3 6.8 https://vulners.com/githubexploit/C94132FD-1FA5-5342-B6EE-0DAF45EEFFE3 *EXPLOIT*
* Nmap: |   10213DBE-F683-58BB-B6D3-353173626207 6.8 https://vulners.com/githubexploit/10213DBE-F683-58BB-B6D3-353173626207 *EXPLOIT*
* Nmap: |   PRION:CVE-2020-12062 5.0 https://vulners.com/prion/PRION:CVE-2020-12062
* Nmap: |   CVE-2020-12062 5.0 https://vulners.com/cve/CVE-2020-12062
* Nmap: |   CVE-2010-4816 5.0 https://vulners.com/cve/CVE-2010-4816
* Nmap: |   PRION:CVE-2021-28041 4.6 https://vulners.com/prion/PRION:CVE-2021-28041
* Nmap: |   CVE-2021-28041 4.6 https://vulners.com/cve/CVE-2021-28041
* Nmap: |   PRION:CVE-2021-41617 4.4 https://vulners.com/prion/PRION:CVE-2021-41617
* Nmap: |   CVE-2021-41617 4.4 https://vulners.com/cve/CVE-2021-41617
* Nmap: |   PRION:CVE-2020-14145 4.3 https://vulners.com/prion/PRION:CVE-2020-14145
* Nmap: |   PRION:CVE-2016-20012 4.3 https://vulners.com/prion/PRION:CVE-2016-20012
* Nmap: |   CVE-2020-14145 4.3 https://vulners.com/cve/CVE-2020-14145
* Nmap: |   CVE-2016-20012 4.3 https://vulners.com/cve/CVE-2016-20012
* Nmap: |   CVE-2023-51767 3.5 https://vulners.com/cve/CVE-2023-51767
* Nmap: |   PRION:CVE-2021-36368 2.6 https://vulners.com/prion/PRION:CVE-2021-36368
* Nmap: |   CVE-2021-36368 2.6 https://vulners.com/cve/CVE-2021-36368
* Nmap: | 111/tcp open  rpcbind 2-4 (RPC #100000)
* Nmap: | rpcinfo:
* Nmap: |   program version port/proto service
* Nmap: |   100000 2,3,4 111/tcp  rpcbind
* Nmap: |   100000 2,3,4 111/udp  rpcbind
* Nmap: |   100000 3,4 111/tcp6 rpcbind
* Nmap: |   100000 3,4 111/udp6 rpcbind
* Nmap: Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
* Nmap: Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
* Nmap: Nmap done: 1 IP address (1 host up) scanned in 6.48 seconds
msf6 >
```

Figure 23: Metasploit with Nmap Integration Total Results from Near-RT RIC

The Kube-hunter results reveal a significant vulnerability related to Kubernetes version disclosure in Figure 24, which mirrors findings in the Non-RT RIC environment, albeit with different Kubernetes versions (Near-RT is v1.16.0 but v1.22.17 from Non-RT RIC). This vulnerability exposes the Kubernetes version via the API server's /version endpoint, potentially aiding attackers in exploiting known issues specific to these versions. Despite the version differences, the consistent presence of this vulnerability indicates similar security configurations across both environments. To mitigate this risk, it is imperative to restrict access to sensitive endpoints, regularly update Kubernetes to the latest stable versions, and implement stringent role-based access control (RBAC) policies. These measures will enhance the security posture and protect against potential exploits, ensuring robust defense mechanisms are in place for both Near-RT RIC and Non-RT RIC environments.

```

Nodes
-----+-----+
| TYPE | LOCATION |
+-----+-----+
| Node/Master | 192.168.40.130 |
+-----+-----+

Detected Services
-----+-----+-----+
| SERVICE | LOCATION | DESCRIPTION |
+-----+-----+-----+
| Kubelet API | 192.168.40.130:10250 | The Kubelet is the main component in every Node, all pod operations goes through the kubelet |
+-----+-----+-----+
| Etcd | 192.168.40.130:2379 | Etcd is a DB that stores cluster's data, it contains configuration and current state information, and might contain secrets |
+-----+-----+-----+
| API Server | 192.168.40.130:6443 | The API server is in charge of all operations on the cluster. |
+-----+-----+-----+

Vulnerabilities
For further information about a vulnerability, search its ID in:
https://avd.aquasec.com/
-----+-----+-----+-----+-----+-----+
| ID | LOCATION | MITRE CATEGORY | VULNERABILITY | DESCRIPTION | EVIDENCE |
+-----+-----+-----+-----+-----+-----+
| KHV002 | 192.168.40.130:6443 | Initial Access // Exposed sensitive interfaces | K8s Version Disclosure | The Kubernetes version could be obtained from the /version endpoint | v1.16.0 |
+-----+-----+-----+-----+-----+-----+

root@ubuntu:~/kube-hunter#

```

Figure 24: Kube-hunter Total Results from Near-RT RIC

In the context of ensuring robust security for O-RAN with SSDLC, a detailed comparison of these tools is shown in Table 22. Various tools are employed across different phases to enhance security and mitigate vulnerabilities. During the Requirement and Design Phase, OWASP Dependency Check and Mend.io are utilized. OWASP Dependency Check is advantageous due to its capability to detect known vulnerabilities in project dependencies, its support for multiple programming languages, and its regular updates. However, it may present false positives or negatives and is limited to known vulnerabilities. Mend.io offers comprehensive open-source security and compliance management, automated policy enforcement, and real-time alerts, though it can be expensive for small teams and requires complex initial setup and configuration.

In the Implementation Phase, tools like Codacy, Aikido, Embold, and SonarQube are critical. Codacy provides continuous code quality and security analysis with support for various programming languages and CI/CD tools integration, but its free tier has limited customization and it may miss context-specific issues. Aikido, tailored for O-RAN systems, integrates real-time threat intelligence and offers comprehensive security checks but is

relatively new and necessitates thorough training. Embold identifies design issues, vulnerabilities, and code smells, prioritizing issues based on impact, although it may require significant configuration and can be costly. SonarQube, known for detecting bugs, vulnerabilities, and code smells across over 25 programming languages, has a strong community and extensive documentation, but performance issues may arise with large codebases, and some features require a paid version.

For the Test and Deploy Phase, Nessus, Trivy, and Nikto are employed. Nessus provides comprehensive vulnerability scanning with regular updates and detailed reports, though it can be resource-intensive and expensive in its professional version. Trivy is a fast and simple tool that scans containers, filesystems, and Git repositories, but it is limited to known vulnerabilities and may produce false positives. Nikto is an open-source web server scanner that detects various vulnerabilities and configuration issues, but it generates a high number of false positives and is limited to web server vulnerabilities.

Finally, in the Maintenance Phase, OpenVAS, Metasploit with Nmap, and Kube-hunter are instrumental. OpenVAS offers a comprehensive open-source vulnerability scanning solution with regular updates and detailed remediation guidance, but it is complex to set up and resource-intensive. The combination of Metasploit with Nmap is powerful for penetration testing, featuring an extensive database of exploits and strong community support, yet it requires significant expertise and may be overkill for small projects. Kube-hunter, designed specifically for Kubernetes security, detects a wide range of Kubernetes vulnerabilities and is easy to set up, though it is limited to Kubernetes environments and may not cover all types of vulnerabilities.

Table 22 Tool Alignment with SSDLC Phases for O-RAN

Phase	Tool	Pros	Cons
Requirement	OWASP	- Detects known vulnerabilities in project dependencies.	- May have false positives or false negatives.

and Design	Dependency Check	<ul style="list-style-type: none"> - Supports multiple programming languages. - Regular updates with the latest vulnerability data. 	<ul style="list-style-type: none"> - Limited to known vulnerabilities only.
	Mend.io	<ul style="list-style-type: none"> - Comprehensive open-source security and compliance management. - Automated policy enforcement. - Real-time alerts and detailed remediation guidance. 	<ul style="list-style-type: none"> - Can be expensive for small teams. - Initial setup and configuration can be complex.
Implement	Codacy	<ul style="list-style-type: none"> - Continuous code quality and code security analysis. - Supports various programming languages. - Integration with multiple CI/CD tools. 	<ul style="list-style-type: none"> - Limited customization in the free tier. - May miss some context-specific issues.
	Aikido	<ul style="list-style-type: none"> - Specific focus on O-RAN systems. - Comprehensive security and compliance checks. - Real-time threat intelligence integration. 	<ul style="list-style-type: none"> - Still a relatively new tool, so it may have some teething issues. - Requires thorough training for effective use.
	Embold	<ul style="list-style-type: none"> - Identifies design issues, vulnerabilities, and code smells. - Prioritizes issues based on impact. - Supports a wide range of programming languages. 	<ul style="list-style-type: none"> - May require significant configuration for optimal use. - Pricing can be high for extensive features.
	SonarQube	<ul style="list-style-type: none"> - Detects bugs, vulnerabilities, and code smells. - Supports over 25 programming languages. - Strong community and extensive documentation. 	<ul style="list-style-type: none"> - Performance can be an issue with large codebases. - Some advanced features require a paid version.
Test and	Nessus	<ul style="list-style-type: none"> - Comprehensive vulnerability scanning. 	<ul style="list-style-type: none"> - Can be resource-intensive.

Deploy		<ul style="list-style-type: none"> - Regular updates with the latest vulnerabilities. - Easy-to-use interface and detailed reports. 	<ul style="list-style-type: none"> - Higher cost for the professional version.
	Trivy	<ul style="list-style-type: none"> - Fast and simple to use. - Scan containers, filesystems, and Git repositories. - Regular updates with the latest vulnerability data. 	<ul style="list-style-type: none"> - Limited to known vulnerabilities. - May have false positives.
	Nikto	<ul style="list-style-type: none"> - Open-source web server scanner. - Detects various vulnerabilities and configuration issues. - Regularly updated with new vulnerabilities. 	<ul style="list-style-type: none"> - Generates a high number of false positives. - Limited to web server vulnerabilities.
	OpenVAS	<ul style="list-style-type: none"> - Comprehensive open-source vulnerability scanner. - Regular updates with the latest vulnerabilities. - Detailed reporting and remediation guidance. 	<ul style="list-style-type: none"> - Can be complex to set up and configure. - Resource-intensive during scans.
Maintenance	Metasploit with Nmap	<ul style="list-style-type: none"> - Powerful combination for penetration testing. - Extensive database of known exploits. - Strong community and regular updates. 	<ul style="list-style-type: none"> - Requires significant expertise to use effectively. - Can be overkill for small projects.
	Kube-hunter	<ul style="list-style-type: none"> - Specifically designed for Kubernetes security. - Detects a wide range of Kubernetes vulnerabilities. - Easy to set up and run. 	<ul style="list-style-type: none"> - Limited to Kubernetes environments. - May not cover all types of vulnerabilities.

4.3 Demonstration

4.3.1 OpenVAS: ICMP Timestamp Disclosure

From the various results, some vulnerabilities appear in both Non-RT RIC and Near-RT RIC, including the ICMP Timestamp disclosure identified using OpenVAS in Figure 25, as described in Tables 17 and 21. ICMP or Internet Control Message Protocol operates at the network layer and helps network devices diagnose communication issues. One notable message type within ICMP is the Timestamp message, which allows devices to request and respond with the current time in milliseconds since midnight UTC. However, the vulnerability identified as CVE-1999-0524 involves systems responding to these ICMP Timestamp requests, inadvertently revealing their system time. This disclosure can be exploited by attackers, presenting several risks. Attackers can obtain system uptime information, enabling them to discern patterns of activity and maintenance, thus aiding in the planning of targeted attacks. Additionally, it allows for network mapping, enabling attackers to identify the most active devices. Furthermore, timestamp information can be used to synchronize coordinated attacks, leveraging precise timing to maximize their impact.



Vulnerability	Severity
SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	7.5 (High)
TCP Timestamps Information Disclosure	2.6 (Low)
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Low)
ICMP Timestamp Reply Information Disclosure	2.1 (Low)

Figure 25: OpenVAS Vulnerability Scan Results Before Remediation

To address the identified issue, various solutions were evaluated. The solution selected, due to its simplicity and minimal side effects, involved blocking ICMP timestamps using Iptables. This approach was chosen because it effectively mitigates the vulnerability without significantly impacting other network functionalities. As shown in Figure 26, a specific rule was added to the iptables configuration to discard ICMP timestamp requests and replies. The

figure confirms that this rule has been successfully implemented within the system, ensuring that no timestamp information is transmitted.

```
root@ubuntu:~# iptables -A INPUT -p icmp --icmp-type timestamp-request -j DROP
root@ubuntu:~# iptables -A OUTPUT -p icmp --icmp-type timestamp-reply -j DROP
root@ubuntu:~# iptables -L | grep timestamp
DROP      icmp -- anywhere          anywhere          icmp timestamp-request
DROP      icmp -- anywhere          anywhere          icmp timestamp-reply
root@ubuntu:~#
```

Figure 26: iptables Rules for Dropping ICMP Timestamp Requests and Replies

Figures 27 and 28 detail the testing process using hping3, a network tool for packet crafting and analysis. In Figure 27, an ICMP timestamp reply test directed at the destination server resulted in an "operation not permitted" message, clearly indicating that the ICMP timestamp reply cannot be sent. This response confirms the effectiveness of the rule in blocking outgoing ICMP timestamp replies. Figure 28 presents the results of an ICMP timestamp request test initiated from an external server. The test revealed that all five requested packets resulted in 100% packet loss, meaning the ICMP timestamp requests could not reach the server. This outcome further validates the rule's efficacy in blocking incoming ICMP timestamp requests.

```
root@ubuntu:~# hping3 --icmp --icmptype 14 kali-server
HPING kali-server (ens33 192.168.40.133): icmp mode set, 28 headers + 0 data bytes
[send_ip] sendto: Operation not permitted
root@ubuntu:~#
```

Figure 27: ICMP Timestamp Reply Testing Results

```
(kali@kali)-[~]
└─$ sudo hping3 --icmp --icmptype 13 non-rt
HPING non-rt (eth0 192.168.40.128): icmp mode set, 28 headers + 0 data bytes
^C
— non-rt hping statistic —
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 28: ICMP Timestamp Request Testing Results

To ensure comprehensive mitigation of the ICMP timestamp vulnerability, an additional test was conducted using OpenVAS again. Figure 29 illustrates the results of this test,

confirming that the ICMP timestamp vulnerability had been successfully resolved.

Vulnerability	Severity
SSL/TLS: Report Vulnerable Cipher Suites for HTTPS	7.5 (High)
TCP Timestamps Information Disclosure	2.6 (Low)
Weak MAC Algorithm(s) Supported (SSH)	2.6 (Low)

(Applied filter: apply_overrides=0 levels=hml rows=100 min_qod=70 first=1 sort-reverse=severity)

Figure 29: OpenVAS Vulnerability Scan Results After Remediation

While blocking ICMP timestamps is an effective measure to mitigate certain vulnerabilities, it is important to consider the potential side effects on network operations. One major challenge involves troubleshooting difficulties. ICMP, particularly echo requests and replies (commonly known as ping), is a fundamental tool used for network troubleshooting. By blocking or limiting ICMP traffic, network administrators may find it more challenging to diagnose connectivity issues, as it becomes harder to determine if a host is reachable or to identify the source of connectivity problems.

Additionally, network monitoring disruptions can occur. Many network monitoring tools rely on ICMP to assess the availability and latency of devices. Blocking ICMP traffic can interfere with these tools, leading to inaccurate monitoring results. This interference can cause monitoring systems to report false positives or fail to detect actual network issues, thereby reducing the overall effectiveness of network monitoring and management.

Furthermore, remote management challenges arise when ICMP is blocked. Remote management tools often use ICMP to verify connectivity before proceeding with more complex operations. Disrupting ICMP traffic can make remote management less reliable, potentially leading to increased downtime or difficulties in managing remote devices effectively. These challenges underscore the importance of carefully weighing the benefits of blocking ICMP timestamps against the potential impact on network operations and management.

Blocking ICMP timestamps using iptables has been demonstrated as an effective solution

for mitigating the CVE-1999-0524 vulnerability in both Non-RT RIC and Near-RT RIC systems. This method prevents the disclosure of system time, which could otherwise be exploited for network mapping, system uptime analysis, and coordinated attacks. Successful implementation and testing, using tools like hping3 and OpenVAS, confirmed the efficacy of this approach with minimal impact on other network functionalities.

However, the strategy of blocking ICMP traffic presents potential trade-offs. It can complicate network troubleshooting, disrupt network monitoring, and hinder remote management tasks. These challenges necessitate a careful balance between the security benefits and operational impact. Thus, while blocking ICMP timestamps is a straightforward and effective mitigation tactic, ongoing evaluation and adjustments are essential to maintain both network functionality and security at optimal levels.

4.3.2 Kube-hunter: Kubernetes Version Disclosure

In both the Non-RT RIC and Near-RT RIC environments, Figures 19 and 26, respectively, depict a vulnerability associated with Kubernetes version disclosure identified by Kube-hunter. This vulnerability underscores a significant security threat linked to the exposure of the Kubernetes version used in the system infrastructure. Knowledge of specific Kubernetes versions can markedly increase the risk of targeted attacks, as adversaries can exploit known vulnerabilities specific to that version. Critical information sources include the Kubernetes API `/version`` endpoint, which can divulge essential version details. Preventing unauthorized access to such information is crucial for maintaining a secure environment.

When attackers ascertain the specific Kubernetes version in use, they can exploit known vulnerabilities within that version, potentially leading to unauthorized access, privilege escalation, or service disruption. For example, a certain version might have a documented privilege escalation flaw that an attacker could exploit to gain administrative access. Additionally, knowledge of the Kubernetes version enables more precise and effective

targeted attacks compared to generic ones. Attackers can develop exploits tailored to the specific version, making their attacks more successful and difficult to defend against.

The public disclosure of version information also assists attackers in reconnaissance, providing valuable insights into the system's defenses and potential vulnerabilities. This information allows attackers to plan their strategies more effectively, increasing the likelihood of a successful breach. Moreover, exposing such sensitive information can lead to compliance and regulatory issues. Regulatory frameworks like GDPR or HIPAA require strict controls on information disclosure, and non-compliance can result in legal repercussions, fines, and a loss of trust from customers and stakeholders.

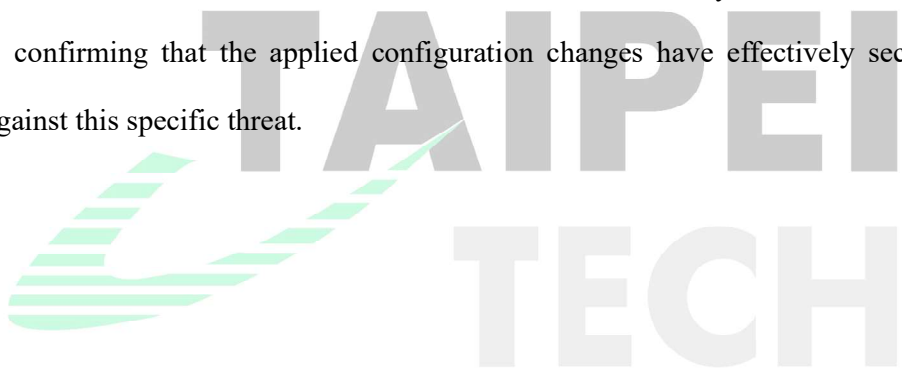
To effectively and minimally disruptively address the Kubernetes version disclosure vulnerability, the recommended solution is to modify the API Server Configuration file by adding the "--enable-debugging-handlers=false" flag to the command section as shown in Figure 30. This file is typically located at "/etc/kubernetes/manifests/kube-apiserver.yaml". By default, Kubernetes enables debugging handlers, which can expose sensitive information, including the Kubernetes version, via endpoints like /version. Disabling debugging handlers ensures that these endpoints are not available, thereby preventing unauthorized users from accessing them.

By implementing this configuration, the security of the Kubernetes environment is significantly enhanced. Only necessary API endpoints will be exposed, and debugging information that includes version details will not be accessible. This greatly reduces the risk of attackers exploiting known vulnerabilities associated with specific Kubernetes versions. The restriction on debugging handlers mitigates potential security breaches by limiting the exposure of version details that could be used for targeted attacks.

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 192.168.40.128:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
  - command:
    - kube-apiserver
    - --enable-debugging-handlers=false
    - --advertise-address=192.168.40.128
```

Figure 30: Kubernetes API Server Configuration with Debugging Handlers Disabled

After making the necessary changes to the API Server Configuration file, it is crucial to verify the effectiveness of the solution. Running the Kube-hunter tool again post-implementation showed that "No vulnerabilities were found" as depicted in Figure 31. This result indicates that the Kubernetes version disclosure vulnerability has been successfully resolved, confirming that the applied configuration changes have effectively secured the system against this specific threat.



```

Nodes
+-----+-----+
| TYPE | LOCATION |
+-----+-----+
| Node/Master | 192.168.40.128 |
+-----+-----+

Detected Services
+-----+-----+-----+
| SERVICE | LOCATION | DESCRIPTION |
+-----+-----+-----+
| Unrecognized K8s API | 192.168.40.128:6443 | A Kubernetes API service |
+-----+-----+-----+
| Kubelet API | 192.168.40.128:10250 | The Kubelet is the main component in every Node, all pod operations goes through the kubelet |
+-----+-----+-----+
| Etcd | 192.168.40.128:2379 | Etcd is a DB that stores cluster's data, it contains configuration and current state information, and might contain secrets |
+-----+-----+-----+

No vulnerabilities were found

```

Figure 31: Kube-hunter Vulnerability Scan Results After Remediation

Implementing the "--enable-debugging-handlers=false" flag to mitigate the Kubernetes version disclosure vulnerability does come with some significant side effects that need to be addressed. One of the primary side effects is limited debugging capabilities. Disabling this flag prevents access to several crucial debugging endpoints such as `/metrics`, `/logs`, `/run`, `/exec`, `/attach`, and `/portforward`. These endpoints are essential for diagnosing issues within pods and the cluster, and without them, troubleshooting issues with pods and nodes may become more challenging, requiring alternative methods to gather necessary information. Another significant side effect is the impact on automated tools. Any automated scripts or tools that rely on these endpoints for monitoring or debugging will need to be updated or replaced, potentially leading to temporary disruptions in operations until the changes are implemented. Additionally, there are potential gaps in monitoring. Disabling these handlers means losing access to some metrics provided by the `/metrics` endpoint, which can lead to

monitoring gaps unless compensated by other monitoring solutions that do not depend on these endpoints.

To address these side effects, several mitigation strategies can be implemented. For alternative monitoring solutions, dedicated monitoring tools like Prometheus can be used to gather metrics without relying on Kubelet's debugging endpoints. Comprehensive logging solutions can also be implemented to capture and analyze logs from all cluster components. Enhanced authentication and authorization policies can be implemented to ensure that only authorized users can access these endpoints, instead of completely disabling debugging handlers. Finally, documenting the changes and training the operations team on alternative debugging and monitoring methods is essential to ensure a smooth transition. By carefully planning and implementing these strategies, the security of the Kubernetes environment can be enhanced while minimizing the operational impact of disabling the "--enable-debugging-handlers" flag.

In conclusion, addressing the Kubernetes version disclosure vulnerability in both Non-RT RIC and Near-RT RIC environments is critical for maintaining a secure infrastructure. This vulnerability, identified by Kube-hunter, poses a significant threat by potentially allowing attackers to exploit known vulnerabilities associated with specific Kubernetes versions. Implementing the "--enable-debugging-handlers=false" flag in the API Server Configuration file ensures that sensitive debugging information is not exposed, thereby preventing unauthorized access to version details. This configuration significantly enhances security by restricting access to necessary endpoints only. Verification through tools like Kube-hunter should show that the vulnerability has been resolved. However, it is essential to address potential side effects such as limited debugging capabilities, impact on automated tools, and potential gaps in monitoring by implementing alternative monitoring solutions, enhancing authentication and authorization, and providing necessary documentation and training.

Chapter 5 Conclusion and Future Work

From Table 22, it is evident that each tool has both advantages and disadvantages for the O-RAN system. To achieve optimal results, it is essential to implement cross-functional testing, which revealed that many vulnerabilities were detected as early as the requirement phase, underscoring the importance of early security integration. Addressing these vulnerabilities at this stage ensures potential issues are identified before they can be exploited. However, attempts to fix these issues post-deployment led to various side effects, the severity of which varied by case, resulting in increased development or fixing costs, consuming additional resources and time, and introducing new risks and unforeseen challenges. The findings emphasize the necessity of incorporating security measures from the initial stages of system design. By addressing vulnerabilities early, organizations can prevent complications that arise later, minimizing costs and reducing the likelihood of introducing new risks during deployment. Integrating security tools and practices early promotes a more robust and resilient O-RAN system, allowing for comprehensive assessments and targeted solutions that are more effective and less disruptive. Therefore, it is advisable to fix vulnerability problems from the beginning of the design phase before the system is deployed, ensuring a more secure, efficient, and cost-effective development process.

Adopting a proactive approach to address vulnerabilities from the beginning of a project is essential. This strategy prevents serious issues post-deployment and enhances ongoing improvement and resilience against new threats. By providing specific steps for remediation, these tools streamline the process of securing open-source systems, reducing the complexity and time required for fixes. Additionally, they facilitate the assessment of potential impacts before updating libraries or dependencies, ensuring that any changes do not introduce new vulnerabilities. This proactive approach helps maintain a secure development environment, fosters continuous improvement, and enhances the overall resilience of the software against

future threats.

The benefits of employing multiple security tools are considerable, as they detect vulnerabilities and offer detailed remediation solutions, simplifying the task of addressing security issues early in the development process. These tools streamline the process of securing open-source systems by providing specific steps for remediation, reducing the complexity and time required for fixes. They also facilitate the assessment of potential impacts before updating libraries or dependencies, ensuring that any changes do not introduce new vulnerabilities. This proactive approach helps maintain a secure development environment, fosters continuous improvement, and enhances the overall resilience of the software against future threats. Additionally, aligning with the security specifications set forth by the WG11 (Security) of the O-RAN Alliance is crucial. WG11 emphasizes the need for robust security measures in the open RAN architecture, addressing concerns such as authentication, authorization, data integrity, and confidentiality. By integrating multiple security tools that adhere to these specifications, organizations can ensure their O-RAN implementations meet the highest security standards, protecting against known threats and anticipating emerging vulnerabilities specific to the O-RAN ecosystem. Consequently, the combined use of these tools and adherence to WG11 guidelines significantly bolsters the security posture of O-RAN systems, fostering trust and reliability in these critical network components.

Integrating CI/CD pipelines to continuously test software for security issues is a priority. This approach catches problems early and provides immediate feedback, fostering a security-first culture. Integrating these tools with CI/CD pipelines will facilitate continuous security testing and immediate feedback for developers, enhancing the security of open-source systems. Utilizing advanced automated tools for real-time vulnerability scanning and remediation throughout the software development lifecycle is essential. These tools should work seamlessly together, providing unified dashboards for streamlined monitoring and management of security issues. Enhancing machine learning capabilities within these tools

can improve the accuracy of vulnerability predictions and prioritization based on potential impact, optimizing resource allocation.

Leveraging machine learning to predict and prioritize vulnerabilities based on impact is critical. This enables efficient resource use and proactive threat management. Additionally, enhancing machine learning capabilities within these tools can improve the accuracy of vulnerability predictions and prioritization based on potential impact, optimizing resource allocation. It is also crucial to extend the scope of security tools to address emerging technologies and architectures, such as serverless computing and microservices, to ensure comprehensive threat mitigation.



References

- [1] S. Parkvall, E. Dahlman, A. Furuskar, and M. Frenne, “NR: The new 5G radio access technology,” *IEEE Communications Standards Magazine*, vol. 1, no. 4, pp. 24–30, Dec. 2017, doi: 10.1109/MCOMSTD.2017.1700042.
- [2] A. S. Abdalla, P. S. Upadhyaya, V. K. Shah, and V. Marojevic, “Toward Next Generation Open Radio Access Networks: What O-RAN Can and Cannot Do!,” *IEEE Netw*, vol. 36, no. 6, pp. 206–213, Nov. 2022, doi: 10.1109/MNET.108.2100659.
- [3] B.-S. P. LinI, “Toward an AI-Enabled O-RAN-based and SDN/NFV-driven 5 G & IoT Network Era,” *Network and Communication Technologies*, vol. 6, no. 1, p. 6, Jun. 2021, doi: 10.5539/nct.v6n1p6.
- [4] 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 and Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023, doi: 10.1109/COMST.2023.3239220.
- [5] H. Lefeuvre, V.-A. B⁺, Y. Chien, F. Huici, N. Dautenhahn, and P. Olivier, “Assessing the Impact of Interface Vulnerabilities in Compartmentalized Software”, doi: 10.14722/ndss.2023.24117.
- [6] M. Alavirad *et al.*, “O-RAN architecture, interfaces, and standardization: Study and application to user intelligent admission control,” *Frontiers in Communications and Networks*, vol. 4, 2023, doi: 10.3389/frcmn.2023.1127039.
- [7] “About O-RAN ALLIANCE.” [Online]. Available: <https://www.o-ran.org/about>
- [8] M. Liyanage, A. Braeken, S. Shahabuddin, and P. Ranaweera, “Open RAN security: Challenges and opportunities,” *Journal of Network and Computer Applications*, vol. 214. Academic Press, May 01, 2023. doi: 10.1016/j.jnca.2023.103621.
- [9] C. T. Shen *et al.*, “Security Threat Analysis and Treatment Strategy for ORAN,” in

- International Conference on Advanced Communication Technology, ICACT*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 417–422. doi: 10.23919/ICACT53585.2022.9728862.
- [10] “The Complete Guide to O-RAN Alliance - Moniem-Tech.” [Online]. Available: <https://moniem-tech.com/2022/09/15/the-complete-guide-to-o-ran-alliance/>
- [11] “The O-RAN ALLIANCE Security Work Group Continues Defining O-RAN Security Solutions.” [Online]. Available: <https://www.o-ran.org/blog/the-o-ran-alliance-security-work-group-continues-defining-o-ran-security-solutions>
- [12] “O-RAN Software Community (O-RAN SC).” [Online]. Available: <https://wiki.o-ran-sc.org/>
- [13] “Releases Version of O-RAN.” [Online]. Available: <https://wiki.o-ran-sc.org/display/REL/Releases>
- [14] A. Boulanger, “Open-source versus proprietary software: Is one more reliable and secure than the other?,” *IBM Systems Journal*, vol. 44, no. 2, pp. 239–248, 2005, doi: 10.1147/SJ.442.0239.
- [15] Z. Pan *et al.*, “Ambush From All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines,” *IEEE Trans Dependable Secure Comput*, 2023, doi: 10.1109/TDSC.2023.3253572.
- [16] F. Asisi Bimo *et al.*, “OSC Community Lab: The Integration Test Bed for O-RAN Software Community,” *ArXiv*, p. arXiv:2208.14885, Aug. 2022, doi: 10.48550/ARXIV.2208.14885.
- [17] “O-RAN Downloads - WG1: Use Cases and Overall Architecture Workgroup (O-RAN Architecture Description 9.0).” [Online]. Available: <https://orandownloadsweb.azurewebsites.net/specifications>
- [18] “What is the RIC in Open RAN ? - 5G Training and 5G Certification.” [Online]. Available: <https://www.5gworldpro.com/blog/2022/09/12/what-is-the-ric-in-open-ran/>

- [19] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 21–27, Oct. 2021, doi: 10.1109/MCOM.101.2001120.
- [20] M. J. Heron, V. L. Hanson, and I. Ricketts, "Open source and accessibility: advantages and limitations," *Journal of Interaction Science 2013 1:1*, vol. 1, no. 1, pp. 1–10, May 2013, doi: 10.1186/2194-0827-1-2.
- [21] M. Almarzouq, L. Zheng, G. Rong, and V. Grover, "Communications of the Association for Information Systems Open Source: Concepts, Benefits, and Challenges Recommended Citation Open Source: Concepts, Benefits, and Challenges," *Communications of the Association for Information Systems*, vol. 16, pp. 756–784, 2005, doi: 10.17705/1CAIS.01637.
- [22] G. Schryen, "What does vulnerability and patch data say? is open source security a myth?", doi: 10.1145/1941487.1941516.
- [23] H. Plate, S. E. Ponta, and A. Sabetta, "Impact assessment for vulnerabilities in open-source software libraries," *2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings*, pp. 411–420, Nov. 2015, doi: 10.1109/ICSM.2015.7332492.
- [24] B. Barritt and W. Eddy, "Service Management & Orchestration of 5G and 6G Non-Terrestrial Networks," *IEEE Aerospace Conference Proceedings*, vol. 2022-March, 2022, doi: 10.1109/AERO53065.2022.9843390.
- [25] M. Daghmehchi Firoozjaei, J. (Paul) Jeong, H. Ko, and H. Kim, "Security challenges with network functions virtualization," *Future Generation Computer Systems*, vol. 67, pp. 315–324, Feb. 2017, doi: 10.1016/J.FUTURE.2016.07.002.
- [26] "O-Ran Policy Coalition, 2021. Open RAN Security in 5G," 2021. [Online]. Available: <https://www.openranpolicy.org/wp-content/uploads/2021/04/Open-RAN-Security-in-5G-4.29.21.pdf>

- [27] “O-RAN Downloads - WG2: Non-real-time RAN Intelligent Controller and A1 Interface Workgroup (O-RAN Non-RT RIC Architecture 3.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [28] Marcin Dryjański, “O-RAN Non-RT RIC: Architecture and rApps.” [Online]. Available: <https://rimedolabs.com/blog/o-ran-non-rt-ric-architecture-and-rapps/>
- [29] “Ericsson, 2020. Security Considerations of Open RAN,” 2020. [Online]. Available: www.ericsson.com/en/security/a-guide-to-5g-network-security
- [30] Amy Zwarico and Sébastien Jeux, “The O-RAN ALLIANCE Security Task Group Tackles Security Challenges on All O-RAN Interfaces and Components.” [Online]. Available: <https://www.o-ran.org/blog/the-o-ran-alliance-security-task-group-tackles-security-challenges-on-all-o-ran-interfaces-and-components>
- [31] Y. Jin, M. Maniatakos, and Y. Makris, “Exposing vulnerabilities of untrusted computing platforms,” *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 131–134, 2012, doi: 10.1109/ICCD.2012.6378629.
- [32] R. H. Sloan and R. Warner, “Unauthorized Access,” p. 401, 2017.
- [33] A. W. Kondoro and J. S. Mtebe, “Investigating Secure Implementation of Government Web based Systems in Tanzania,” pp. 978–979, 2018, Accessed: Nov. 02, 2023. [Online]. Available: www.IST-Africa.org/Conference2018
- [34] M. Helenius and M. Vallius, “REST API Security: Testing and Analysis,” May 2022, Accessed: Nov. 02, 2023. [Online]. Available: <https://trepo.tuni.fi/handle/10024/139682>
- [35] “O-RAN Downloads - WG3: Near-real-time RIC and E2 Interface Workgroup (O-RAN Near-RT RIC Architecture 4.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [36] H.-T. Thieu, V.-Q. Pham, A. Kak, and N. Choi, “Demystifying the Near-real Time RIC: Architecture, Operations, and Benchmarking Insights,” *IEEE INFOCOM 2023 - IEEE*

- Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–8, May 2023, doi: 10.1109/INFOCOMWKSHPS57453.2023.10225852.
- [37] T. O. Atalay, S. Maitra, D. Stojadinovic, A. Stavrou, and H. Wang, “Securing 5G OpenRAN with a Scalable Authorization Framework for xApps,” pp. 1–10, Aug. 2023, doi: 10.1109/INFOCOM53939.2023.10228961.
- [38] S. Soltani, M. Shojafar, A. Brighente, M. Conti, and R. Tafazolli, “Poisoning Bearer Context Migration in O-RAN 5G Network,” *IEEE Wireless Communications Letters*, vol. 12, no. 3, pp. 401–405, Mar. 2023, doi: 10.1109/LWC.2022.3227676.
- [39] X. Han, N. Kheir, and D. Balzarotti, “Deception Techniques in Computer Security,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, Jul. 2018, doi: 10.1145/3214305.
- [40] 3rd Generation Partnership Project (3GPP), “Study on CU-DU lower layer split for NR: Technical Report (TR) 38.816, version 15.0.0.,” 2017.
- [41] A. Đurović, A. Plečić, F. Banković, and G. Marković, “OPEN RAN-POSSIBILITIES AND CHALLENGES”, doi: 10.37528/FTTE/9788673954165/POSTEL.2022.021.
- [42] S. Niknam *et al.*, “Intelligent O-RAN for Beyond 5G and 6G Wireless Networks,” in *2022 IEEE GLOBECOM Workshops, GC Wkshps 2022 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 215–220. doi: 10.1109/GCWkshps56602.2022.10008676.
- [43] N. Kazemifard and V. Shah-Mansouri, “Minimum delay function placement and resource allocation for Open RAN (O-RAN) 5G networks,” *Computer Networks*, vol. 188, p. 107809, Apr. 2021, doi: 10.1016/J.COMNET.2021.107809.
- [44] W. Azariah, F. A. Bimo, C.-W. Lin, R.-G. Cheng, R. Jana, and N. Nikaein, “A Survey on Open Radio Access Networks: Challenges, Research Directions, and Open Source Approaches,” Aug. 2022, Accessed: Oct. 05, 2023. [Online]. Available: <https://arxiv.org/abs/2208.09125v1>
- [45] H. Hojeij, M. Sharara, S. Hoteit, and V. Vèque, “Dynamic Placement of O-CU and O-

- DU Functionalities in Open-RAN Architecture,” Sep. 2023, doi: 10.13039/501100001665.
- [46] L. Goyal and B. Keswani, “Study, analysis and formulation of a new method for integrity protection of digital data from unauthorized access,” *IJSRD-International Journal for Scientific Research & Development*, vol. 1, 2013, Accessed: Nov. 24, 2023. [Online]. Available: www.ijrd.com
- [47] “O-RAN Downloads - WG6: Cloudification and Orchestration Workgroup (O-RAN Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN 5.0).” [Online]. Available: <https://orandownloadsweb.azurewebsites.net/specifications>
- [48] F. Klement *et al.*, “Open or not open: Are conventional radio access networks more secure and trustworthy than Open-RAN?,” Apr. 2022, Accessed: Nov. 01, 2023. [Online]. Available: <https://arxiv.org/abs/2204.12227v3>
- [49] M. Kandias, N. Virvilis, and D. Gritzalis, “The insider threat in cloud computing,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6983 LNCS, pp. 93–103, 2013, doi: 10.1007/978-3-642-41476-3_8/COVER.
- [50] M. A. Habibi, G. YILMA, X. Costa-Perez, and H. D. Schotten, “Unifying 3GPP, ETSI, and O-RAN SMO Interfaces: Enabling Slice Subnets Interoperability,” Oct. 2023, doi: 10.36227/TECHRXIV.24225532.V1.
- [51] “3GPP TS 38.460: NG-RAN; E1 general aspects and principles.” [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3428>
- [52] “3GPP TS 38.470: NG-RAN; F1 general aspects and principles.” [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3257>
- [53] “3GPP TS 38.300: NR; NR and NG-RAN Overall Description; Stage 2.” [Online].

Available:

<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191>

- [54] “3GPP TS 36.420: Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 general aspects and principles.” [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2449>
- [55] “3GPP TS 38.420: NG-RAN; Xn general aspects and principles.” [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3225>
- [56] “3GPP TS 38.401: NG-RAN; Architecture description.” [Online]. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3219>
- [57] “O-RAN Downloads - WG2: Non-real-time RAN Intelligent Controller and A1 Interface Workgroup (O-RAN A1 interface: General Aspects and Principles 3.01).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [58] “O-RAN Downloads - WG10: OAM for O-RAN (O-RAN Operations and Maintenance Interface Specification 11.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [59] “O-RAN Downloads - WG10: OAM for O-RAN (O-RAN Operations and Maintenance Architecture 10.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [60] “O-RAN Downloads - WG3: Near-real-time RIC and E2 Interface Workgroup (O-RAN E2 General Aspects and Principles (E2GAP) 4.01).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [61] “O-RAN Downloads - WG4: Open Fronthaul Interfaces Workgroup (O-RAN Control,

- User and Synchronization Plane Specification 13.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [62] “O-RAN Downloads - WG4: Open Fronthaul Interfaces Workgroup (O-RAN Management Plane Specification 13.0).” [Online]. Available: <https://orandownloadswb.azurewebsites.net/specifications>
- [63] A. Zrahia, “Threat intelligence sharing between cybersecurity vendors: Network, dyadic, and agent views”, doi: 10.1093/cybsec/tyy008.
- [64] R. Di Pietro and F. Lombardi, “Virtualization Technologies and Cloud Security: advantages, issues, and perspectives”.
- [65] A. K. Y. S. Mohamed, D. Auer, D. Hofer, and J. K ung, “A systematic literature review for authorization and access control: definitions, strategies and models,” *International Journal of Web Information Systems*, vol. 18, no. 2–3, pp. 156–180, Oct. 2022, doi: 10.1108/IJWIS-04-2022-0077/FULL/PDF.
- [66] L. Kim, “Cybersecurity: Ensuring Confidentiality, Integrity, and Availability of Information,” pp. 391–410, 2022, doi: 10.1007/978-3-030-91237-6_26.
- [67]  . Aslan, S. S. Aktuđ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions,” *Electronics 2023, Vol. 12, Page 1333*, vol. 12, no. 6, p. 1333, Mar. 2023, doi: 10.3390/ELECTRONICS12061333.
- [68] F. Sampson, “Data Privacy and Security: Some Legal and Ethical Challenges,” *Advanced Sciences and Technologies for Security Applications*, pp. 109–134, 2021, doi: 10.1007/978-3-030-72120-6_4/COVER.
- [69] S. G. Gollagi *et al.*, “A Study on Secure Software Development Life Cycle (SSDLC),” pp. 801–809, 2021, doi: 10.1007/978-981-33-4367-2_76.
- [70] M. Sharma, “Review of the Benefits of DAST (Dynamic Application Security Testing) Versus SAST,” *INTERNATIONAL JOURNAL OF MANAGEMENT AND*

- ENGINEERING RESEARCH*, vol. 1, no. 1, pp. 05–08, Jun. 2021, Accessed: Dec. 19, 2023. [Online]. Available: <https://www.ijmer.org/index.php/journal/article/view/2>
- [71] “Enterprise Open Source and Linux | Ubuntu.” [Online]. Available: <https://ubuntu.com/>
- [72] “Docker: Accelerated Container Application Development.” [Online]. Available: <https://www.docker.com/>
- [73] “Kubernetes.” [Online]. Available: <https://kubernetes.io/>
- [74] “ChartMuseum - Helm Chart Repository.” [Online]. Available: <https://chartmuseum.com/>
- [75] “Helm.” [Online]. Available: <https://helm.sh/>
- [76] “Installation Guide for Non-RT RIC.” [Online]. Available: <https://wiki.o-ran-sc.org/display/IAT/Automated+deployment+and+testing+-+using+SMO+package+and+ONAP+Python+SDK>
- [77] “Installation Guide for Near-RT RIC.” [Online]. Available: <https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-ric-dep/en/latest/installation-guides.html#installing-near-realtime-ric-in-ric-cluster>
- [78] R. Ross, M. McEvelley, and J. Carrier Oren, “NIST SP 800-160 Volume 1: Systems Security Engineering,” 2018, doi: 10.6028/NIST.SP.800-160v1r1.
- [79] “OWASP Dependency-Check | OWASP Foundation.” [Online]. Available: <https://owasp.org/www-project-dependency-check/>
- [80] “Mend Bolt: Find & Fix Open Source vulnerabilities.” [Online]. Available: <https://www.mend.io/free-developer-tools/bolt/>
- [81] “Codacy - Code Quality and Security for Developers.” [Online]. Available: <https://www.codacy.com/>
- [82] “Aikido — AppSec Platform For Code & Cloud Security.” [Online]. Available: <https://www.aikido.dev/>
- [83] “Embold | Static Code Analysis Platform.” [Online]. Available: <https://embold.io/>

- [84] “Code Quality, Security & Static Analysis Tool with SonarQube | Sonar.” [Online]. Available: <https://www.sonarsource.com/products/sonarqube/>
- [85] “Nessus Vulnerability Scanner: Network Security Solution | Tenable®.” [Online]. Available: <https://www.tenable.com/products/nessus>
- [86] “Trivy Home - Trivy.” [Online]. Available: <https://trivy.dev/>
- [87] “Nikto 2.5 | CIRT.net.” [Online]. Available: <https://cirt.net/Nikto2>
- [88] “OpenVAS - Open Vulnerability Assessment Scanner.” [Online]. Available: <https://www.openvas.org/>
- [89] “Nmap: the Network Mapper - Free Security Scanner.” [Online]. Available: <https://nmap.org/>
- [90] “Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit.” [Online]. Available: <https://www.metasploit.com/>
- [91] “kube-hunter: Hunt for security weaknesses in Kubernetes clusters.” [Online]. Available: <https://github.com/aquasecurity/kube-hunter>
- [92] “CVE Website.” [Online]. Available: <https://www.cve.org/>
- [93] “CWE - Common Weakness Enumeration.” [Online]. Available: <https://cwe.mitre.org/>
- [94] “CVSS - Vulnerability Metrics.” [Online]. Available: <https://nvd.nist.gov/vulnerability-metrics/cvss>
- [95] “NVD.” [Online]. Available: <https://nvd.nist.gov/>
- [96] “What is an ORM – The Meaning of Object Relational Mapping Database Tools.” [Online]. Available: <https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>