Technology
Arts Sciences
TH Köln

Evaluation of Attack Tools against 5G Open RAN Simulation Environments

Arnova Abdullah

Research Project
Master of Science
Communication Systems and Networks

Supervisor: Professor Dr. Andreas Grebe

June 28, 2023

Abstract

This research project aims to evaluate open-source attack tools for securing 5G Open Radio Access Network (Open RAN) from the perspective of an inside attacker. Different open-source attack tools such as Kali Linux, Infection Monkey, Caldera, and Atomic Red Team have been assessed and evaluated as relevant to 5G Open RAN. Based on the attack use case and attack structure, the Atomic Red Team has been selected for penetration testing. The implementation of 5G Open RAN and attack tests have been created on the Computer Networks Research Group lab server at TH Köln as a proof of concept. The evaluation and analysis of the applied atomic tests in the implemented 5G Open RAN provide valuable insights into the security aspects of the 5G Open RAN simulation environment. Additionally, a mapping of common vulnerabilities and exposures (CVE) IDs to MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) IDs has been conducted, enhancing the understanding of specific vulnerabilities and their corresponding MITRE-defined attack techniques. The findings and insights gained from the penetration testing process will inform future security measures, enabling the development of robust defenses against potential attacks in the 5G Open RAN ecosystem.

Contents

List of Figures

List of Tables

1 Introduction

The rapid evolution of communication technologies has led to the advent of the fifth-generation (5G) wireless network, bringing with it promises of unprecedented speed, capacity, and connectivity with different use cases. Open Radio Access Network (Open RAN) architectures have emerged as a disruptive approach in the telecommunications industry to enable the seamless integration and interoperability of various network components. Open RAN offers a software-driven, open interface-based approach that enables network disaggregation and promotes vendor diversity (1).

This research project focuses on conducting a comprehensive risk analysis of 5G Open RAN by specifically evaluating various open-source attack tools. The objective is to assess the vulnerabilities and potential risks that arise when an attacker gains insider access to the Open RAN infrastructure. By simulating the behavior of an inside attacker, the aim is to uncover weaknesses in the system using the existing security frameworks and open-source attack tools and create attack traces for IT Forensics to ensure the security implementation of Open RAN. Understanding these risks and vulnerabilities is essential to develop robust security frameworks, enhancing system resilience, and protecting the integrity and confidentiality of critical 5G network infrastructure.

Throughout this research, various attack simulation tools will be employed to simulate real-world attack scenarios and evaluate the effectiveness of the existing security mechanisms deployed in Open RAN. The outcomes of this study will provide a comprehensive understanding of the risks associated with insider threats in 5G Open RAN and enable stakeholders to strengthen their security practices and develop proactive measures to safeguard their networks.

1.1 Motivation

The motivation behind this research project stems from the increasing adoption of 5G Open RAN architectures and the critical need to assess and mitigate the security risks associated with this emerging technology. As 5G networks become the backbone of our increasingly interconnected world, it is imperative to ensure their robustness and resilience against potential insider threats. While the advantages of Open RAN, such as flexibility, interoperability, and cost-effectiveness, are well recognized, the inherent openness of Open RAN architectures introduces new security challenges and potential vulnerabilities (2). While significant efforts have been made to address external threats (3), such as malicious actors from the outside, the risks posed by insider attackers cannot be overlooked. An inside attacker, an authorized entity within the network, can exploit their privileged access to compromise system integrity, and user privacy, disrupt services, or cause significant financial losses.

This research project aims to address this critical knowledge gap by conducting a thorough risk analysis of 5G Open RAN from the perspective of inside attackers. Through this analysis, the effectiveness of existing security mechanisms can be assessed, and areas for improvement of 5G Open RAN deployments can be identified.

1.2 Outline

This research project report is organized into several sections to provide a comprehensive analysis and the report follows a logical flow that allows readers to understand the research methodology, findings, and recommendations in a structured manner.

The report begins with a description of the technical fundamentals that provide an overview of the research topic, underlying technologies, and security frameworks. The next section describes the selection and evaluation of various open-source attack tools used to simulate inside attack scenarios in the 5G Open RAN environment. The implementation and result analysis section outlines the research approach and methodology employed to conduct the risk analysis. Details about the implemented setup, parameters, and metrics are provided in two phases. The research findings are also presented and analyzed in this section. The outcomes of the security testing using open-source attack tools are discussed, including vulnerabilities discovered in the 5G Open RAN architecture. The conclusion section concludes the report by summarizing the broader impact of this research and its contribution to the field of secure telecommunication. Finally, the Future Work section provides an in-depth analysis of the research findings, contextualizing them within the broader scope of 5G Open RAN security. It highlights the implications

and limitations of the research, identifies areas for future exploration, and encourages further research and development in the field.

2 Technical Fundamentals

2.1 Evolution of Mobile Network Technology

Since its introduction in 1980, mobile wireless network technology has advanced through each generation, from the first (1G) to the fifth (5G), with the sixth generation (6G) (4) already in development. With each generation came more features and increased speed. Prior to 5G, the fourth generation of broadband cellular network technology, or 4G LTE, represents a major improvement over 3G with the update of an all-IP flat networking structure initiated in 2008 (5). For real-time applications like video conferencing and online gaming, 4G LTE is the best choice since it offers users lower latency, less network congestion, and better sound quality (6). While 5G networks, which provide even higher speeds and lower latency for the future of mobile communication, are still being developed and rolled out, 4G LTE is still the basis for their continued development and widespread use.

2.1.1 5G Architecture

With the launch of 5G (7), the fifth generation of wireless technology, mobile communication has evolved dramatically. Its revolutionary potential, as well as a variety of cutting-edge technologies including autonomous vehicles, smart cities, virtual reality, and the Internet of Things (IoT), made possible by 5G networks, stand to benefit a number of businesses (8). It is anticipated that the deployment of 5G networks would create new opportunities, encourage innovation, and change the ways in which we connect, communicate, and interact with our surroundings.

The Third Generation Partnership Project (3GPP) (9), an international partnership of telecommunications firms in charge of creating mobile communication standards, established the 5G standard. The 3GPP specifications outline the overall architecture, network elements, interfaces, and protocols for 5G systems. These specifications are developed through a collaborative process involving industry experts from telecom operators, equipment manufacturers, and other stakeholders. The goal is to create a standardized framework that enables the deployment of 5G networks and facilitates the development of innovative services and applications. The architecture of 5G consisting of three main components: User Equipment (UE), 5G Radio Access Network (RAN), and 5G Core Network is displayed in figure 1.

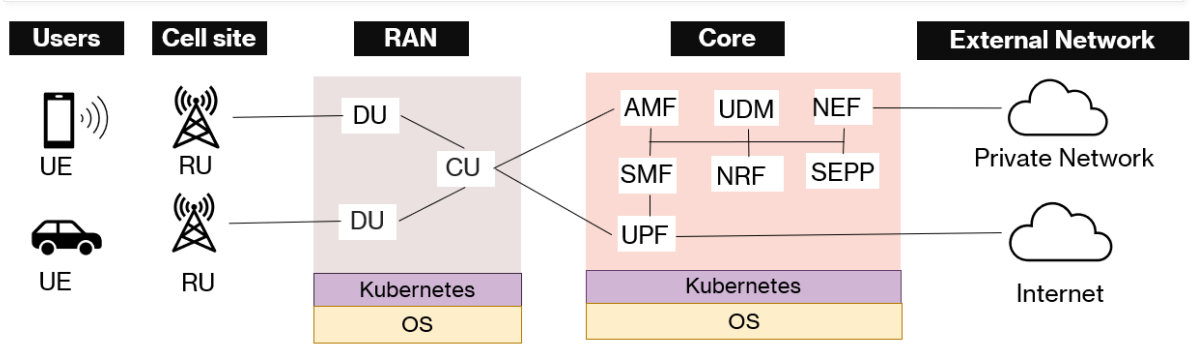

Figure 1: The architecture of 5G consisting of UE, RAN, and core network.

1. **User Equipment (UE):** The UEs are the end-user devices, such as cellphones, Laptops, tablets, or IoT devices, that connect to the 5G network. UEs are equipped with 5G-compatible modems and antennas to establish wireless connections with the 5G RAN.

2. **5G Radio Access Network (RAN):** The 5G RAN is responsible for wireless communication between the UE and the network consisting of different entities that work together to enable communication in the 5G network. The main entity is the gNB (5G Node B), which is responsible for transmitting and receiving radio signals. It is the base station in 5G, and serves as the access point for UEs. The gNB connects to the UE over the air interface and communicates with the 5G Core

Network. The radio interface, called NR-Uu (New Radio-Uu), is used to establish communication between the gNB and user devices. The gNB can be divided into the gNB-Central Unit (gNB-CU) and one or more gNB-Distributed Units (gNB-DUs). These units are connected allowing them to collaborate and efficiently manage the radio access.

3. **5G Core Network:** The 5G Core Network serves as the central part of the 5G architecture, responsible for various network functions and services. It comprises several key elements.
   (a) AMF (Access and Mobility Management Function): The AMF manages the mobility and session handling for UEs, including authentication, security, and handover between different gNBs.
   (b) SMF (Session Management Function): The SMF handles the establishment, modification, and termination of user sessions in the 5G network. It manages the IP address assignment and quality of service (QoS) for user data flows.
   (c) UPF (User Plane Function): The UPF handles the forwarding of user data packets in the 5G network. It performs functions such as packet routing, forwarding, and traffic management.
   (d) NRF (Network Repository Function): The NRF provides a repository for network function information, enabling service discovery and assisting in network function selection and management.
   (e) AUSF (Authentication Server Function): The AUSF handles the authentication and security functions for UEs, verifying their identities and providing secure access to the network.
   (f) UDM (Unified Data Management): The UDM manages user-related data, such as subscriber profiles and subscription information, in a centralized database.
   (g) SEPP (Security Edge Protection Proxy): The SEPP acts as a security gateway or proxy at the edge of the 5G network, providing secure connectivity and protecting the network from various security threats. It is responsible for implementing security measures such as authentication, authorization, and encryption to ensure the confidentiality, integrity, and availability of data and services.
   (h) NEF (Network Exposure Function): The NEF provides open interfaces for third-party applications and services to access 5G network resources and capabilities.

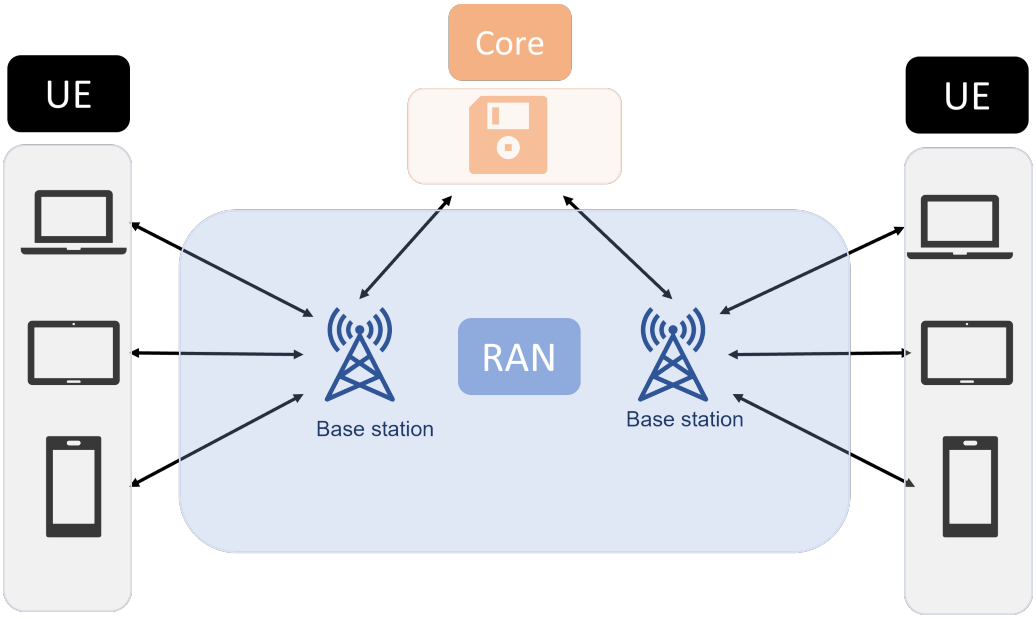

Figure 2: Basic Radio Access Network architecture.

2.2 Radio Access Network (RAN)

An essential part of 5G networks is the RAN, which links consumer devices to the core network. In comparison to earlier generations, the RAN architecture has undergone considerable changes in 5G. Network disaggregation, which enables the separation of hardware and software operations, is one of the fundamental principles of 5G RAN. By establishing open interfaces and defined protocols, this strategy promotes better flexibility, scalability, and interoperability. The implementation of network functions as software instances is made possible by the introduction of virtualization and cloud-native principles in 5G RAN, which promotes resource efficiency and dynamic resource allocation (10). The core objective of the 5G RAN is to offer a very effective, low-latency, and dependable wireless connectivity platform to serve a variety of applications, such as increased mobile broadband, significant Internet of Things (IoT) deployments, and mission-critical communication services.

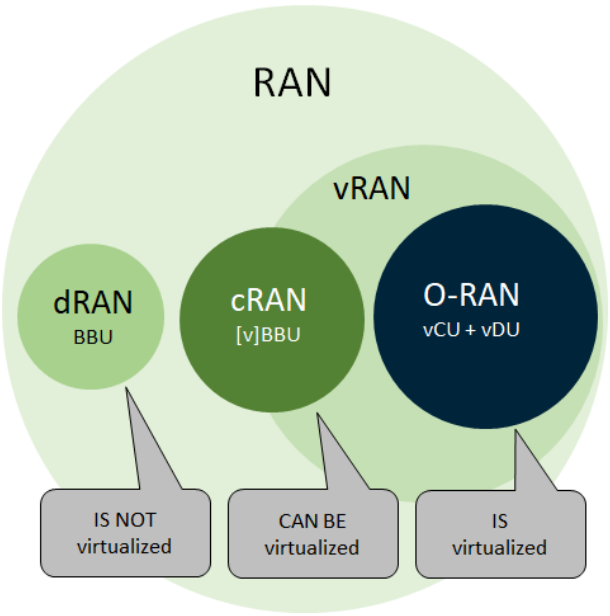

Figure 3: Different types of RAN evolution (11)

2.2.1 Types of RAN

The evolution of RAN has witnessed significant advancements over the years. Figure 3 shows the evolution of RAN technology. Different types of RAN technologies are described in the following:

- **Centralized RAN (C-RAN):** C-RAN is an architecture where the baseband processing functions are centralized in a central unit (CU) while the remote radio heads (RRHs) are distributed at the cell sites. The CU handles the baseband processing tasks, such as modulation, coding, and scheduling, while the RRHs handle the radio frequency (RF) functions. This architecture allows for resource pooling, efficient resource allocation, and easier coordination between base stations.
- **Distributed RAN (D-RAN):** D-RAN also known as Cloud RAN (CRAN), is an architecture that distributes both the baseband processing functions and the RF functions across multiple sites. This architecture enables the processing to be distributed closer to the edge of the network, reducing latency and increasing capacity. D-RAN employs virtualization techniques and cloud-based technologies to enable efficient resource sharing and dynamic allocation of resources.
- **Open RAN (ORAN):** ORAN is an architecture that aims to disaggregate and open up the interfaces between different RAN components, allowing for multi-vendor interoperability and flexibility. It promotes the use of open interfaces and standards-based protocols, enabling network operators to select and combine RAN components from different vendors. Open RAN fosters innovation, and vendor diversity, and reduces dependency on a single vendor, thereby potentially lowering costs and accelerating the deployment of 5G networks.

- **Virtualized RAN (vRAN):** vRAN is an architecture that leverages virtualization technologies to separate the baseband processing functions from the underlying hardware. It enables the virtualization of baseband units (BBUs) and runs them as software instances on standard servers or cloud infrastructure. By virtualizing the RAN functions, vRAN offers flexibility, scalability, and efficient resource utilization.

These different RAN architectures offer various advantages and trade-offs in terms of scalability, flexibility, resource efficiency, and vendor diversity.

2.2.2 Open RAN Full Architecture

An industry initiative called Open RAN, supported by the O-RAN Alliance (12), aims to create an open, efficient, and interoperable Radio Access Network (RAN) architecture for next-generation wireless networks. In order to promote the standardization and adoption of open RAN solutions, network operators, technology providers, and research organizations from around the world have formed the O-RAN Alliance. The development of open interfaces and protocols that permit the functional separation and interoperability of RAN components is one of the main areas of attention of the O-RAN Alliance. Open RAN provides information sharing and coordination between these components from many suppliers by creating open interfaces between the Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU). The flexibility, scalability, and ability to more effectively optimize and deploy RAN resources are all made possible by this disaggregation of RAN functions. As part of the Open RAN architecture, the O-RAN Alliance defines a framework (13) that includes two key components both utilizing the E2 interface:

1. Near Real-Time RAN Intelligent Controller (near-RT RIC)
2. Non-Real-Time RAN Intelligent Controller (non-RT RIC)

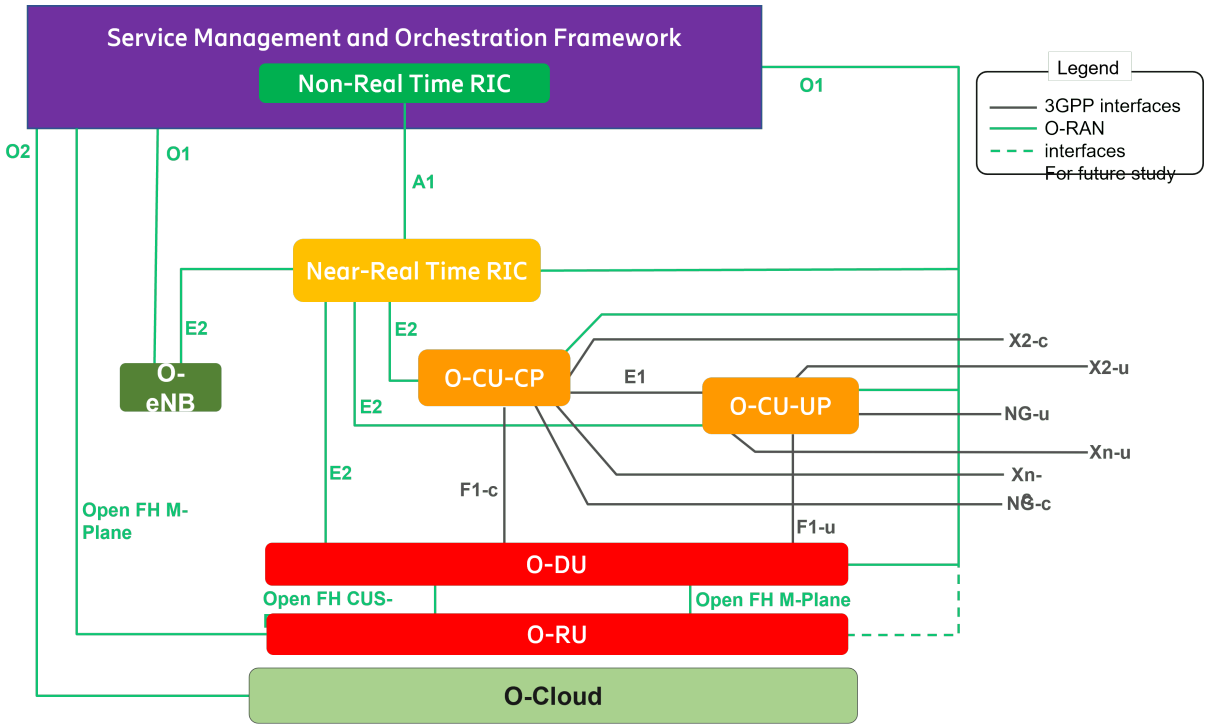

Figure 4: O-RAN logical architecture by O-RAN Alliance (13)

Near-RT RIC: The Near Real-Time RAN Intelligent Controller is responsible for making real-time decisions and optimizations within the RAN. It leverages near real-time data and analytics to dynamically adjust the behavior of the RAN components, such as radio resource allocation and optimization. The near-RT RIC enables intelligent and dynamic management of the RAN to improve performance, capacity, and user experience.

Non-RT RIC: The Non-Real-Time RAN Intelligent Controller focuses on longer-term optimizations and planning. It analyzes historical and aggregated data to make strategic decisions related to network planning, capacity expansion, and overall network optimization. The non-RT RIC plays a crucial role in optimizing the RAN performance over a longer time horizon, based on data-driven insights and predictions. Both the near-RT RIC and non-RT RIC components communicate with the RAN elements through the E2 interface.

A1 interface: The A1 interface serves as a key interface between the Real-Time RIC (RAN Intelligent Controller) and the Non-Real-Time RIC (RAN Intelligent Controller). The A1 interface enables communication and coordination between the Real-Time RIC and the Non-Real-Time RIC for the orchestration and management of the RAN resources. It facilitates the exchange of control, policy, and configuration information to optimize the RAN's performance and enable advanced functionalities.

E2 interface: The E2 interface is a critical component of the Open Radio Access Network (Open RAN) architecture, facilitating communication between the Radio Intelligent Controller (RIC) and the Baseband Unit (BBU) or Radio Unit (RU). It enables near real-time and non-real-time exchange of control messages and configuration information, allowing the RIC to dynamically manage and optimize the RAN performance. The E2 interface promotes interoperability and vendor-neutral collaboration, unlocking the potential for flexible and innovative RAN deployments within the Open RAN ecosystem.

By integrating near real-time and non-real-time intelligence through the near-RT RIC, non-RT RIC, and the E2 interface, Open RAN enables enhanced network optimization, improved resource allocation, and efficient planning. This architecture promotes interoperability, innovation, and vendor diversity, ultimately leading to more flexible and intelligent mobile networks.

2.3 Virtualization Technology

The popular open-source virtualization program VirtualBox, created by Oracle (14), enables users to construct and run numerous virtual machines (VMs) on a single physical computer. Regardless of the host operating system, it offers a framework for executing many operating systems concurrently. Users can build virtual computers with different guest operating systems, including Windows, Linux, macOS, and more, using VirtualBox. In order to allow the virtual machines to function independently, software that emulates computer hardware, such as CPUs, memory, storage, and network interfaces, is used.

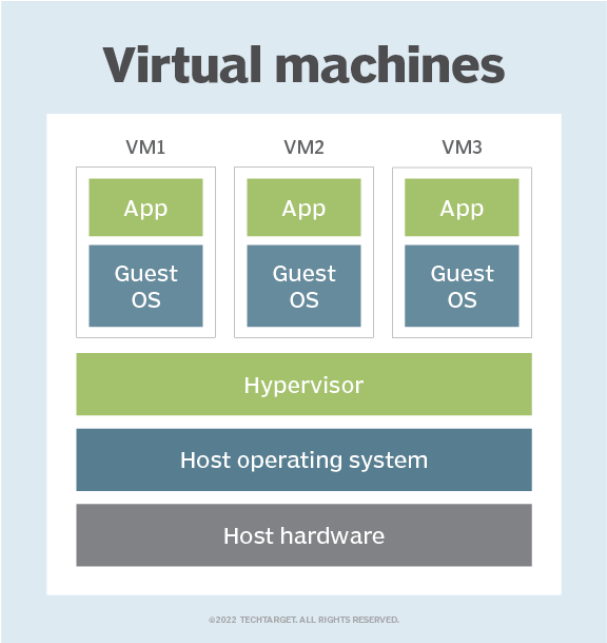

Figure 5: Multiple VMs each running its own applications and guest OS sharing the same hypervisor, host OS (15)

2.4 Containerization Technology

The term "containerization technology" describes the packaging and running of applications and their dependencies inside small, separated containers. Without being impacted by variations in underlying infrastructure or operating systems, it enables programs to be deployed and performed reliably across various computing environments, such as development, testing, and production. Containers give applications a standardized and portable environment by enclosing the required configuration files, libraries, and software in a single, standalone package. Regardless of the host environment, this isolation makes sure that applications function consistently and dependably across various systems.

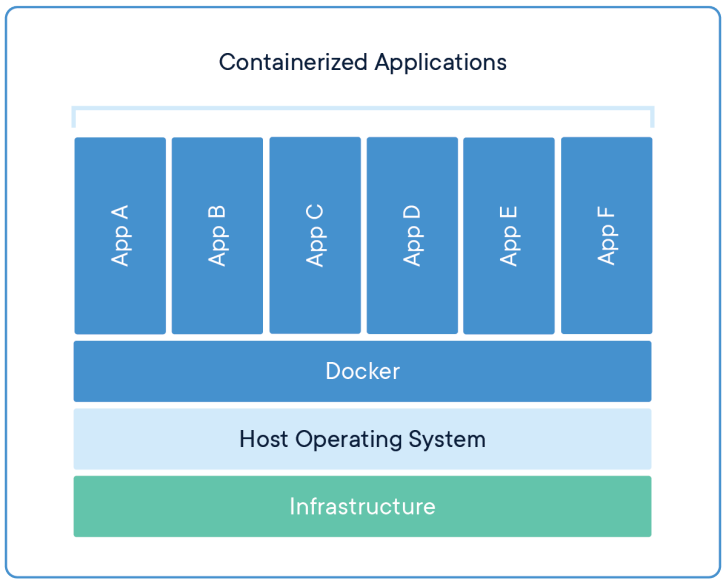

Figure 6: Architecture of Docker containerization technology (16)

2.4.1 Docker

Developers can automate the deployment and administration of applications within software containers using the open-source framework known as Docker (18). It makes use of containerization technology to build compact, mobile, and separated containers that function reliably in a variety of computer settings. Developers can bundle an application with all of its libraries, dependencies, and configuration files into a single container using Docker. These containers ensure that the program can operate consistently and reliably on any machine that supports Docker because they are self-contained and contain everything needed to run the application. The basic architecture of the Docker container is displayed in Figure 6.

For configuring and specifying containers, Docker offers a command-line interface and a declarative language called Dockerfile. Containers are simple to construct, deploy, and manage for developers, enabling quick workflows for developing, testing, and deploying applications.

2.4.2 Kubernetes

K8s, or Kubernetes (19), is a popular abbreviation for this open-source container orchestration system. It offers a framework for managing, scaling, and automating containerized application deployment. Kubernetes, which was created by Google and is currently managed by the Cloud Native Computing Foundation (CNCF), has taken the lead in the container orchestration market. Kubernetes' major goal is to simplify complex distributed system administration by hiding the underlying technology. It frees developers from worrying about the specifics of infrastructure management so they can concentrate on creating and deploying apps. Kubernetes integrates with container runtimes like Docker and provides a rich ecosystem of tools and extensions for monitoring, logging, and networking. Difference between Docker and Kubernetes is displayed in Figure 7. Kubernetes has become a critical component of cloud-native application development, enabling organizations to build scalable, resilient, and portable applications that can run consistently across various cloud and on-premises environments.

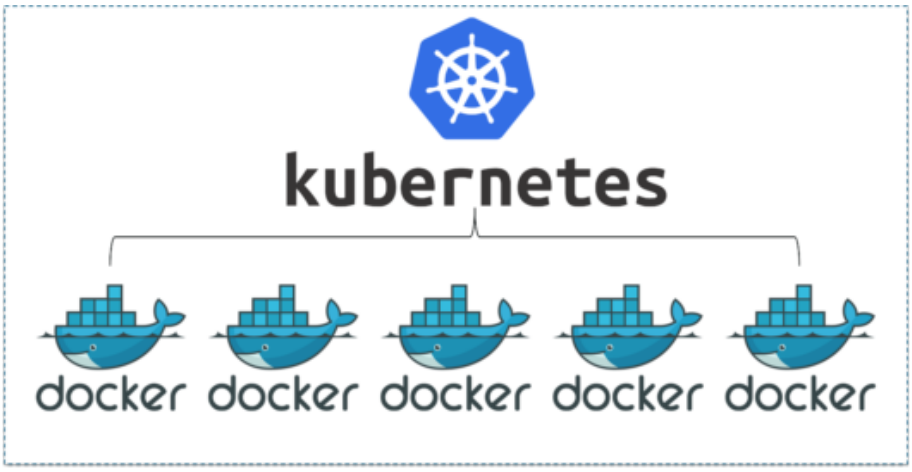

Figure 7: Differenece between Kubernetes and Docker container (17)

In this project, 5G open RAN is implemented in the Kubernetes environment, and attack tests have been implemented against the Kubernetes deployment.

2.5 Security Frameworks

2.5.1 CVE

CVE stands for Common Vulnerabilities and Exposures. It is a standardized system used to identify and track publicly known information security vulnerabilities and exposures. Each susceptibility or exposure is commissioned a unique identifier comprehended as a CVE ID.

The CVE system is maintained and overseen by the non-profit organization MITRE Corporation (20), in collaboration with various industry stakeholders and security researchers. When a vulnerability is discovered, it can be submitted to MITRE for assignment of a CVE ID. The CVE ID serves as a common reference point, allowing security professionals, researchers, and vendors to easily identify and discuss specific vulnerabilities across different platforms and products. The CVE system helps improve communication and coordination within the cybersecurity community, enabling the sharing of vulnerability information and facilitating the process of vulnerability management. It provides a standardized approach for organizations to track and prioritize vulnerabilities, ensuring that appropriate patches or mitigations are applied in a timely manner. CVE IDs are widely used in vulnerability databases, security advisories, and vulnerability management tools. They play a crucial role in enabling the identification, reporting, and remediation of security vulnerabilities to enhance overall cybersecurity.

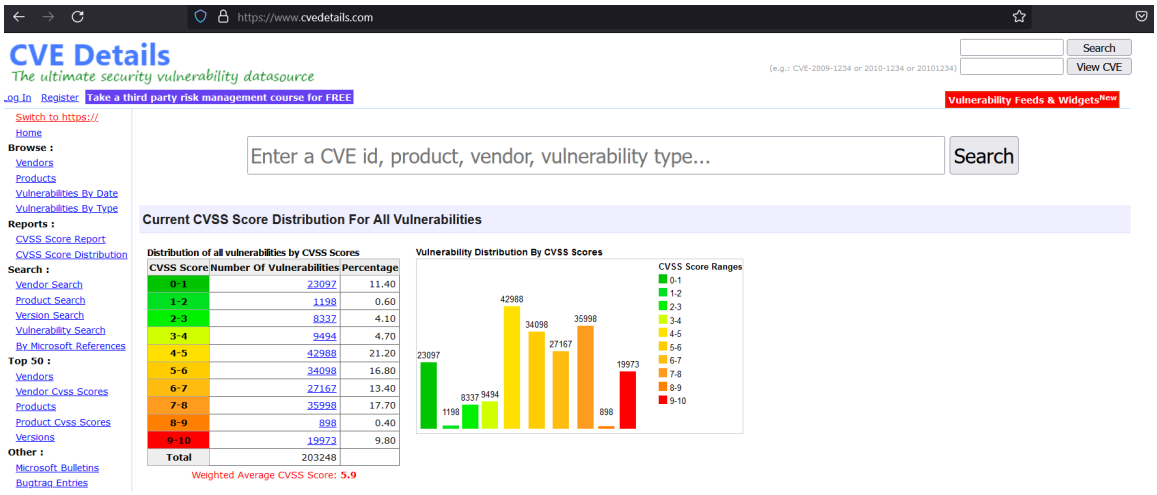

Figure 8: An easy-to-use Web Interface of the CVE database to search for CVE IDs (21)

For this project, another website (21) has been utilized as a valuable resource for conveniently searching for CVE vulnerabilities. This website offers a user-friendly web interface that allows browsing for vendors, products, and versions, providing access to CVE entries and related vulnerabilities. The comprehensive display of CVE details on a single page greatly facilitated my research process.

2.5.2 MITRE ATT&ACK

Adversarial Tactics, Techniques, and Common Knowledge, or MITRE ATT&CK, is a framework that provides a comprehensive knowledge base of cyber threat tactics and techniques used by adversaries (22). It was developed by MITRE, a not-for-profit organization that works in the fields of cybersecurity and defense. ATT&CK consists of a matrix that categorizes and describes various attack techniques across different stages of a cyberattack, from initial access to exfiltration. The matrix is organized based on different platforms (Windows, Linux, macOS) and includes information about specific adversary behaviors, tools, and procedures associated with each technique evolving landscape of cyber threats and adversarial techniques.

The goal of ATT&CK is to assist organizations in understanding and mitigating cyber threats by providing a standardized language and framework for describing and analyzing adversary behaviors. It helps security teams improve their threat detection and response capabilities by mapping defensive measures to specific attack techniques. MITRE ATT&CK has gained widespread adoption in the cybersecurity com-

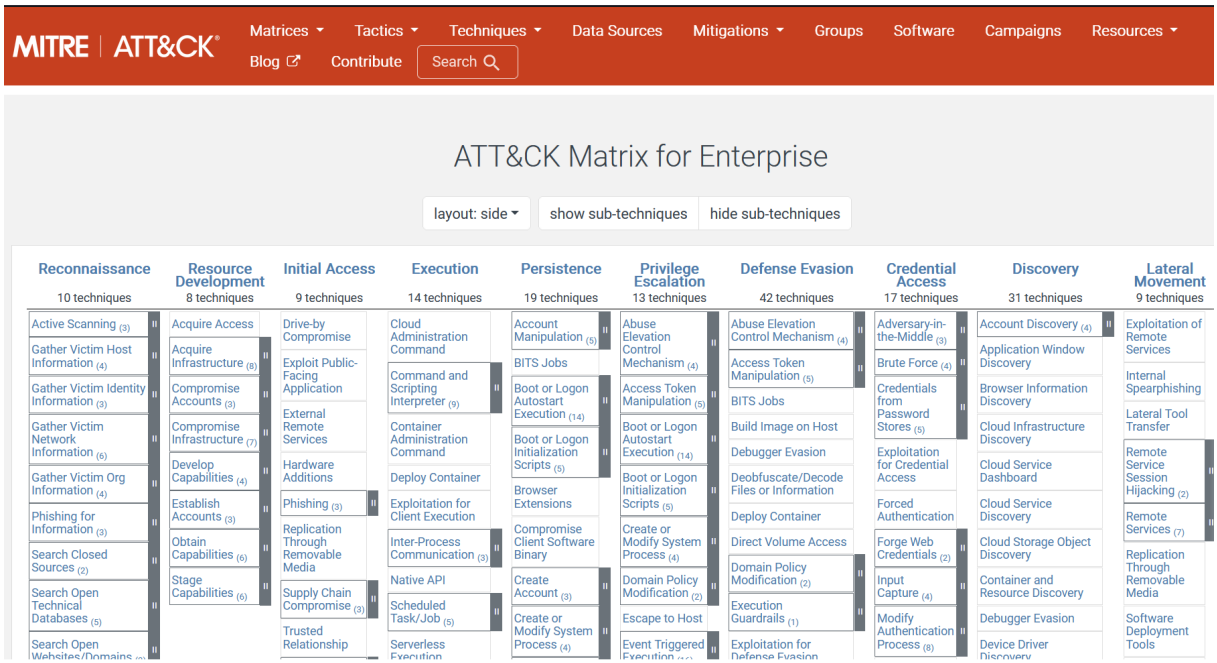

Figure 9: MITRE ATT&CK matrix framework including tactics and techniques (22)

munity and is used by organizations, researchers, and security vendors as a reference for understanding, analyzing, and countering cyber threats. It is regularly updated and maintained to reflect the evolving landscape of cyber threats and adversarial techniques.

3 Evaluation of Open-source Attack Tools

Penetration testing, often known as pen testing or ethical hacking, is a methodical, legal procedure for assessing the security of a system, network, or application by mimicking actual attacks. Penetration testing's main goal is to find weaknesses, vulnerabilities, and potential entry points that could be used by bad actors. It involves the use of various tools and techniques to simulate real-world attacks and identify security vulnerabilities.

Figure 10: Kali Linux OS (23)

3.1 Kali Linux

Kali Linux (23) is a specialized Linux distribution designed for advanced penetration testing and cybersecurity auditing. With its comprehensive suite preinstalled security tools, it offers researchers and professionals the ability to assess and strengthen the security of computer systems, networks, and applications. Featuring a solid foundation based on Debian Linux, Kali Linux provides a customizable environment that can be tailored to meet specific testing requirements.
Its diverse range of tools covers various areas including vulnerability assessment, network analysis, password cracking, wireless attacks, forensics, reverse engineering and other topics are all covered by these technologies. Regular updates and security patches ensure that Kali Linux remains up to date with the latest advancements in the field of cybersecurity. This helps researchers stay on top of emerging threats and utilize the most current tools and techniques.
Kali Linux can often be installed on a computer as an operating system or as a virtual machine. More than 600 preinstalled tools come with Kali Linux. Some important tools are Nmap, Hydra, John the Ripper, Metasploit, Lynis, Fierce, King Phisher, Yersinia, etc. Most of the tools in Kali Linux can be executed from the command line interface.

3.2 Atomic Red Team

Atomic Red Team (24) is an open-source and community-developed structured framework and methodology designed to enhance cybersecurity defense capabilities. It provides a comprehensive set of predefined security tests, known as "atomic tests" that organizations can use to assess and validate the effectiveness of their security controls and detection mechanisms. Simulation of real attack scenarios covering MITRE ATT&CK tactics, techniques, and procedures; Atomic Red Teaming helps organizations identify and address weaknesses in their security infrastructure, ultimately improving their overall resilience to cyber threats.
Atomic tests can be run on Linux, macOS, and Windows operating systems using the command prompt or PowerShell. To simplify the process of creating Atomic Red Team tests, another framework called Invoke-AtomicRedTeam (25) has been established by Red Canary, a cybersecurity company. Invoke-AtomicRedTeam is a PowerShell module that provides a framework for executing Atomic Red Team tests directly from the command line. Within the Atomic Red Team project, the atomics folder organizes the tests based on the techniques defined in the MITRE ATT&CK Framework. Each technique has its own folder labeled as "T#" (e.g., T1003) containing YAML files that detail the attack procedures for individual atomic tests. The Invoke-AtomicRedTeam PowerShell module utilizes these YAML files to

facilitate the execution of atomic tests by locally or remotely offering a streamlined approach for security professionals to automate the assessment and validation of their security controls against specific attack techniques defined in the MITRE ATT&CK Framework.

3.3 Caldera

Caldera (26) is an open-source penetration testing tool that helps security professionals simulate and evaluate cyber attack scenarios in a controlled environment. Developed by MITRE, Caldera focuses on providing a framework for automated adversary emulation, allowing organizations to test and validate their security defenses against realistic threat scenarios. It is a platform that is used for automating adversary emulation and red teaming activities allowing security professionals to automate the execution of attack scenarios, monitor and analyze the results, and gain insights into potential vulnerabilities in their systems.
One of the key features of Caldera is its ability to create and execute sophisticated attack campaigns. The tool allows users to define and customize various attack techniques, such as phishing, exploitation, lateral movement, and data exfiltration, among others. By leveraging these capabilities, security teams can assess the effectiveness of their security controls and identify potential vulnerabilities and weaknesses.

3.4 Infection Monkey

Infection Monkey (27) is an open-source security testing tool developed by Akamai Technologies. It is designed to simulate real-world attacks and test the resilience of an organization's network and infrastructure. The tool operates by deploying a variety of attack techniques, known as "monkey actions," which imitate the behavior of different types of adversaries. It can be used for various use cases, including testing the effectiveness of security controls, identifying vulnerabilities, and evaluating the overall security posture of an organization. It provides valuable insights into potential weaknesses and helps organizations strengthen their defenses against cyber threats.
One notable feature of Infection Monkey is its web interface, which provides a user-friendly graphical interface for interacting with the tool. The web interface allows users to easily configure and launch attack simulations, monitor the progress of attacks, and view comprehensive reports and results. This intuitive interface enhances the usability of Infection Monkey, catering to users with different levels of technical expertise. With its web interface, Infection Monkey offers a convenient and accessible way to identify and address potential security weaknesses in network environments.

3.5 Comparative Evaluation of the Attack Tools

A comparison table has been created to compare different aspects of Kali Linux, Atomic Red Team, Infection Monkey, and Caldera. Table 1 provides information on various factors such as purpose, testing capabilities, target system, ease of use, customization, test coverage, reporting, community support, and integration with other tools. The table allows for an easy comparison of these tools based on their respective features and functionalities.
Considering the focus on insider threats, Caldera stands out as it specifically emphasizes adversary simulation, which includes simulating tactics used by inside attackers. Caldera can be used for automation in conjunction with Atomic Red Team. Caldera is an adversary emulation platform that allows users to create and execute attack scenarios providing a framework for automating red team operations, including the execution of Atomic Red Team tests. By integrating Atomic Red Team tests into Caldera, users can automate the execution of specific atomic tests and incorporate them into their overall adversary simulation workflows.
However, in this project, the focus is primarily on manual testing using Atomic Red Team rather than utilizing automation platforms like Caldera. The objective is to perform hands-on assessment and evaluation by manually executing specific atomic tests from Atomic Red Team.
Kali Linux and Infection Monkey can also be utilized in this context, but their primary purpose is not centered around insider threats. They can still be valuable for identifying vulnerabilities and testing security controls but may require customization and adaptation to address specific insider threat scenarios.
Besides, When utilizing Infection Monkey for testing purposes, it's important to consider that certain functionalities may require the use of external services such as AWS, which may incur costs. One attack tool Vulmap (28) was found in Kali Linux which is an open-source online local vulnerability scanning project designed to identify vulnerabilities in various software applications and systems. Vulmap also provides detailed information about the detected vulnerabilities, including their severity level, affected

Aspect	Kali Linux	Atomic Red Team	Infection Monkey	Caldera
Purpose	Penetration testing	Vulnerability analysis	Breach and attack simulation (BAS)	Adversary simulation
Testing Capabilities	600 preinstalled tools	Modules mapped to MITRE	Network-based attacks	Modules mapped to MITRE
Source	Open-source	Open-source Github project	Open-source	Open-source Github project
Focus on Insider Threats	Possible usage	Possible usage	Possible usage	Strong focus
Target System	Local and remote	Local and remote	Local and remote	Local and remote
Ease of Use	Intermediate to advanced	Intermediate	Intermediate	Intermediate
Customization	Highly customizable	Moderate customization	Limited customization	Highly customizable
Test Coverage	Diverse test suite	Specific atomic tests	Network-based attacks	Tactic-based simulations
Reporting	Varied reporting options	Basic reporting	Basic reporting	Detailed reporting
Integration with Other Tools	Broad range of integration options	Limited integration options	Limited integration options	Limited integration options

Table 1: Comparative evaluation of open-source attack tools

versions, and potential remediation steps. However, Vulmap can be installed on other OS as well rather than Kali Linux. For this reason and to stay within the project scope, only Atomic Red Team was considered for implementing attack tests.

4 Implementation and Result Analysis

The practical work has been implemented in two phases. The first phase is for researching existing systems and frameworks and proposing an attack architecture and the second phase is for implementing the proposed framework from the first phase as a proof of concept. Two phases of the practical work are described in the following section.

4.1 Phase 1

4.1.1 Proposed Full Attack Architecture

In this project, the security assessment of Open RAN from inside the network involved the utilization of many open-source attack tools that are currently accessible, and the atomic red team has been chosen to do the testing which offers focused and targeted security tests designed to identify vulnerabilities within the RAN infrastructure. As atomic tests are mapped to the MITRE ATT&CK matrix, an additional study has been evaluated for mapping the known vulnerabilities to the MITRE matrix which provides a standardized approach for categorizing and describing cybersecurity threats. By aligning the CVE IDs with the MITRE IDs structured framework, the security posture of Open RAN was systematically evaluated. The atomic attacks were executed to assess the system's resilience against potential exploits and to uncover any weaknesses or gaps that could compromise the network's security. This approach allowed to gain insights into the vulnerabilities present in Open RAN and provided valuable information for enhancing security measures.

The goal of this project is to use the CVE database to identify Open RAN and its supporting technologies' vulnerabilities. Open RAN is still not vulnerable, according to the CVE database. So, I turned to Kubernetes, the underlying technology where the open RAN is implemented. Because there are numerous vulnerabilities for Kubernetes that have been found in the CVE database, the strategy is to create attack

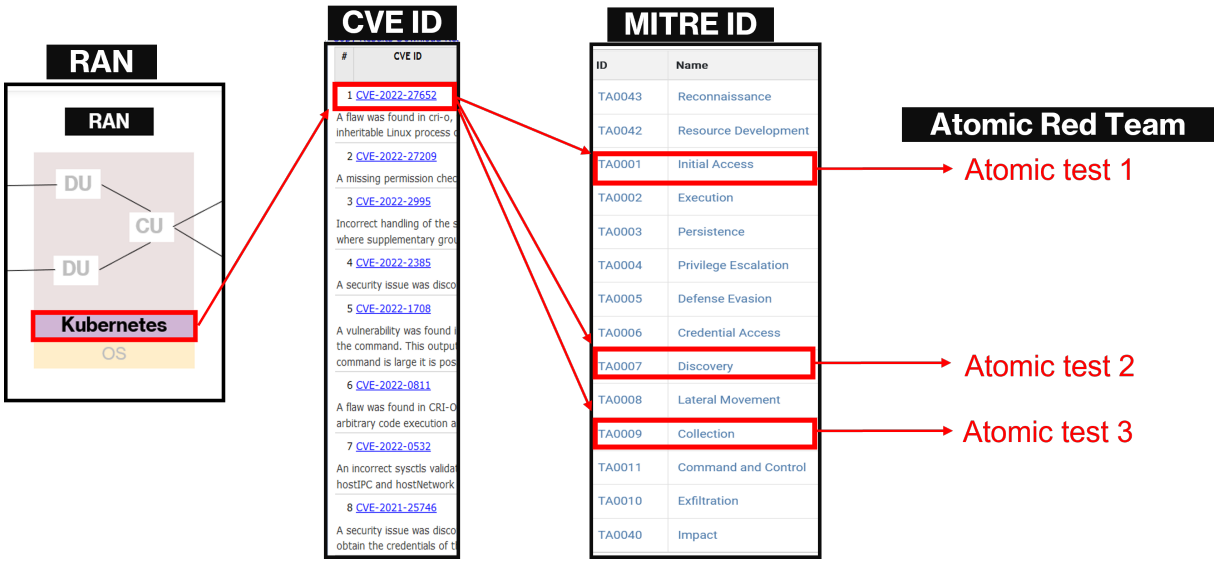

Figure 11: Proposed attack architecture for 5G Open RAN implemented in Kubernetes environment.

tests against the Kubernetes environment as proof of concept. The proposed comprehensive attack architecture entails finding CVE IDs for the Kubernetes environment where the 5G Open RAN would be implemented, mapping CVE IDs to MITRE IDs, and then performing atomic testing using the MITRE IDs. Figure 11 shows the complete attack architecture.

4.1.2 Mapping between CVE and MITRE ATTACK

A valuable resource on GitHub (29) has been discovered that provided a comprehensive methodology for mapping CVE identifiers to MITRE ATT&CK attack IDs. This project offered a systematic approach to aligning known vulnerabilities, represented by CVEs, with the MITRE ATT&CK framework, which focuses on adversarial tactics, techniques, and procedures (TTPs). The project outlined a process that allowed the establishment of connections between specific vulnerabilities and the corresponding ATT&CK techniques, enhancing the understanding of the potential impact and implications of the identified weaknesses. By leveraging this mapping process, the security of Open RAN not only in terms of vulnerabilities but also from the perspective of potential adversarial actions is able to be assessed. This comprehensive approach provided a deeper insight into the potential attack vectors and helped inform the security analysis.

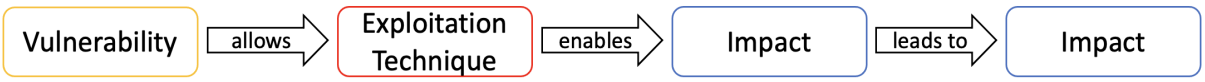

Figure 12: CVE ID to MITRE ID mapping (29)

As described in the GitHub project, the methodology introduces a way to use the MITRE ATT&CK framework to describe vulnerabilities and their exploitation process. It suggests breaking down the steps of an attack into three categories:

1. Exploitation Technique: The specific method employed to exploit the vulnerability.
2. Primary Impact: The initial benefit gained through the exploitation of the vulnerability.
3. Secondary Impact: What the adversary can accomplish by leveraging the primary impact.

The methodology acknowledges that not all categories will have corresponding techniques in the ATT&CK framework. In such cases, either the secondary impact can be used in place of the primary impact or one of the tactic-level techniques can be utilized.

The document also presents three methods for mapping ATT&CK techniques to the CVE vulnerabilities:

1. Vulnerability Type: Grouping vulnerabilities with similar types and common technique mappings, including all three categories described earlier.
2. Functionality: Grouping vulnerabilities based on the functionality the attacker gains access to, including primary and secondary impacts.
3. Exploit Technique: Grouping techniques based on the common steps taken to exploit a vulnerability, focusing on the exploitation technique category.

It is noted that not all categories may be included in each method's mapping, as certain variations and complexities within vulnerability groups may hinder comprehensive mappings.

CVE ID	**Vulnerability type**	**Primary Impact**	**Secondary Impact**	**Exploitation Technique**	**Technology and Year**
CVE-2022-3294	Bypass of Proxy Address Validation in kube-apiserver	T1056, T1005, T1119	T1496, T1489, T1529, T1537	T1548.002	Kubernetes, 2022
CVE-2022-3162	Unauthorized Access to Custom Resources in the Same API Group	T1078	N/A	T1040	Kubernetes, 2022
CVE-2021-25743	Unneutralized Escape Sequences in kubectl Output	T1565	N/A	T1219	Kubernetes, 2021
CVE-2020-8562	Proxy Bypass to Access Private Networks	T1590.002	N/A	T1548.002	Kubernetes, 2020

Table 2: Some CVE IDs for Kubernetes for the year 2022 and 2023 are mapped to MITRE IDs.

Given that no vulnerabilities were found during the 5G Open RAN project's working phase, the list of vulnerabilities for Kubernetes was created as an insider aspect and mapped to MITRE IDs using this methodology. The mapped MITRE IDs are based on the exploitation techniques, and the primary and the secondary impact of the exploitations. Some recent CVEs from the years 2023 and 2022 have been chosen for this mapping and for the atomic test implementation in order to stay within the parameters of this project. Table 2 shows the mapping including the year and the technology.

For instance, the CVE-2022-3294 vulnerability is defined as a flaw in the kube-apiserver in Kubernetes that allowed bypassing of the validation of the proxying address for Nodes. This means that even though Kubernetes validates the proxying address, the bug enabled authenticated requests destined for Nodes to reach the API server's private network, potentially bypassing intended security measures. This implies that a local attacker with the capacity to alter Node objects and issue proxy requests can obtain unauthorized access to the API server's private network, potentially disclosing confidential information or carrying out illegal operations within the cluster's control plane. In the MITRE ID T1056, adversaries may utilize techniques for gathering user input. Because access to the API server's private network allows adversaries to get sensitive data from it, this CVE ID is assigned to T1056 as a primary impact. The rest of the CVE IDs and MITRE IDs are mapped in a similar methodology.

4.2 Phase 2

4.2.1 Open RAN Implementation in Kubernetes

The two main components of Open RAN are near real-time RIC and non-real-time RIC, as mentioned in the 2.2.2. Both near real-time RIC and non-real-time RIC have been implemented by another student in his project. To stay within the scope of my project, only near real-time RIC was considered for investigating the attack approach. To successfully implement the near real-time RIC of Open RAN on our lab server, I collaborated with him who provided valuable assistance throughout the implementation process. This collaboration allowed us to leverage his expertise and work together toward achieving the project goals. I express my gratitude to the student for his invaluable contributions, as his involvement significantly contributed to the successful implementation of the near real-time RIC on our server infrastructure.

```
root@oran:/home/arnova/ric-dep/bin# kubectl get pods -n ricplt
NAME                                                  READY   STATUS    RESTARTS   AGE
deployment-ricplt-a1mediator-74f45b6bc6-rvrwf         1/1     Running   9          36d
deployment-ricplt-alarmmanager-7f7986fd57-q6z7f       1/1     Running   5          36d
deployment-ricplt-appmgr-c47b999bc-gdq92              1/1     Running   5          36d
deployment-ricplt-e2mgr-855fdb9777-lhnpk              1/1     Running   14         36d
deployment-ricplt-e2term-alpha-867f7484c5-gn2bf       1/1     Running   6          36d
deployment-ricplt-o1mediator-6f7d8998cf-4nm6x         1/1     Running   5          36d
deployment-ricplt-rtmgr-5b7965bc8f-q428j              1/1     Running   15         36d
deployment-ricplt-submgr-f8fdfdb54-758gn              1/1     Running   7          36d
deployment-ricplt-vespamgr-84f7d87dfb-bnr59           1/1     Running   5          36d
r4-infrastructure-kong-7995f4679b-s55c6               2/2     Running   14         36d
r4-infrastructure-prometheus-alertmanager-5798b78f48-7rdvn   2/2     Running   10         36d
r4-infrastructure-prometheus-server-c8ddcfdf5-tgzjl   1/1     Running   5          36d
statefulset-ricplt-dbaas-server-0                     1/1     Running   5          36d
root@oran:/home/arnova/ric-dep/bin#
```

Figure 13: ORAN components installed in the pods inside the Kubernetes cluster.

For the implementation, a virtual machine of Ubuntu 20.04.6 LTS was given on the Computer Networks Research Group lab server as the open RAN simulation project is based on a Ubuntu Linux machine. Kubernetes and other dependencies installation were included in the steps of the implementation. After following all the steps, the near real-time RIC part of open RAN was successfully implemented with the pods and services. Installed pods and services with assigned IP addresses are displayed in Figure 13 and 14.

```
root@oran:/home/arnova/ric-dep/bin# kubectl get services -n ricplt
NAME                                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                          AGE
aux-entry                               ClusterIP   10.110.2.107     <none>        80/TCP,443/TCP                   36d
r4-infrastructure-kong-proxy            NodePort    10.108.225.238   <none>        32080:32080/TCP,32443:32443/TCP  36d
r4-infrastructure-prometheus-alertmanager   ClusterIP   10.106.202.174   <none>        80/TCP                     36d
r4-infrastructure-prometheus-server     ClusterIP   10.99.6.89       <none>        80/TCP                           36d
service-ricplt-a1mediator-http          ClusterIP   10.100.115.28    <none>        10000/TCP                        36d
service-ricplt-a1mediator-rmr           ClusterIP   10.105.237.228   <none>        4561/TCP,4562/TCP                36d
service-ricplt-alarmmanager-http        ClusterIP   10.103.83.223    <none>        8080/TCP                         36d
service-ricplt-alarmmanager-rmr         ClusterIP   10.99.41.118     <none>        4560/TCP,4561/TCP                36d
service-ricplt-appmgr-http              ClusterIP   10.105.56.135    <none>        8080/TCP                         36d
service-ricplt-appmgr-rmr               ClusterIP   10.99.171.183    <none>        4561/TCP,4560/TCP                36d
service-ricplt-dbaas-tcp                ClusterIP   None             <none>        6379/TCP                         36d
service-ricplt-e2mgr-http               ClusterIP   10.103.142.245   <none>        3800/TCP                         36d
service-ricplt-e2mgr-rmr                ClusterIP   10.109.234.233   <none>        4561/TCP,3801/TCP                36d
service-ricplt-e2term-prometheus-alpha  ClusterIP   10.110.213.64    <none>        8088/TCP                         36d
service-ricplt-e2term-rmr-alpha         ClusterIP   10.98.97.149     <none>        4561/TCP,38000/TCP               36d
service-ricplt-e2term-sctp-alpha        NodePort    10.98.232.57     <none>        36422:32222/SCTP                 36d
service-ricplt-o1mediator-http          ClusterIP   10.96.226.82     <none>        9001/TCP,8080/TCP,3000/TCP       36d
service-ricplt-o1mediator-tcp-netconf   NodePort    10.100.254.84    <none>        830:30830/TCP                    36d
service-ricplt-rtmgr-http               ClusterIP   10.96.57.76      <none>        3800/TCP                         36d
service-ricplt-rtmgr-rmr                ClusterIP   10.98.105.103    <none>        4561/TCP,4560/TCP                36d
service-ricplt-submgr-http              ClusterIP   None             <none>        3800/TCP                         36d
service-ricplt-submgr-rmr               ClusterIP   None             <none>        4560/TCP,4561/TCP                36d
service-ricplt-vespamgr-http            ClusterIP   10.110.248.104   <none>        8080/TCP,9095/TCP                36d
root@oran:/home/arnova/ric-dep/bin#
```

Figure 14: ORAN services installed inside the Kubernetes cluster.

4.2.2 Atomic Test Implementation

Test cases scenarios: Vulnerability analysis is a critical aspect of ensuring the security and robustness of Open RAN deployments. In this project, Atomic Red Team framework is employed, specifically utilizing the Invoke-AtomicRedTeam tool, to perform comprehensive testing and validation of potential vulnerabilities. The Invoke-AtomicRedTeam framework provided two essential test cases: local tests and remote tests. This framework allows the execution of atomic tests on either the local machine or a remote machine through a PowerShell remoting session.

1. **Local tests:** The local test case involved conducting vulnerability analysis on components and systems within the local environment. This encompassed evaluating the security posture of Open RAN components such as near real-time RIC, and non-real-time RIC deployed within the local system. In this scenario, Atomic Red Team can be installed on the local machine where Open RAN components are installed. By executing predefined test modules provided by Atomic Red Team, various attack scenarios can be simulated, and the effectiveness of the security measures at the local level can be validated.

2. **Remote tests:** The basic concept of remote tests in the context of Atomic Red Team is to perform vulnerability analysis and security testing on a target system that is accessed and assessed from a remote machine. The remote tests focus on evaluating the vulnerabilities associated with the connectivity and communication aspects of the target system, including interfaces, protocols, and interactions with other components. When conducting remote tests with Atomic Red Team, there are some prerequisites to ensuring that the target system and the machine from which we will be running the tests meet the necessary requirements. This may include enabling PowerShell remoting on the target system and installing PowerShell Core on the local machine if it's not running Windows.

During the vulnerability analysis of the Open RAN infrastructure, I opted to perform local tests using the Atomic Red Team framework. While remote testing can provide valuable insights in different scenarios, local testing proved to be the most suitable and efficient approach for specific vulnerability analysis. The decision to focus on local testing instead of remote testing was based on the following considerations:

- **Direct Access:** With local testing, adversaries have direct access to the systems and infrastructure being tested which is a good scenario for inside attackers. This allows for a more comprehensive evaluation of the vulnerabilities and security posture of the components in their actual environment.

- **Testing Scope:** By conducting local tests, I was able to concentrate my efforts on assessing the vulnerabilities and security posture of the Open RAN components themselves, without the additional variables for example for establishing remote PowerShell sessions and configuring PowerShell remoting over SSH which was introduced by remote connectivity and communication protocols.

Steps to perform local atomic tests:

1. **Installation:** The Atomic Red Team and Invoke-AtomicRedTeam framework and the necessary dependencies were installed on the local machine (Ubuntu) on the Lab server.

2. **Importing the module:** I imported the Atomic Red Team module into our PowerShell session to gain access to the available atomic tests.

3. **Listing Atomic Tests:** I used the Atomic Red Team module to list all the available atomic tests. This allowed me to explore the different techniques and test modules available for local testing. Available atomic tests are displayed in Figure 15. Some tests were not available (see the red box for examples in Figure 15) and some were available (see the green box in Figure 15).

4. **Checking Prerequisites:** Before executing the tests, I verified that the local machine met the prerequisites for each atomic test. This step ensured that the required tools, configurations, or permissions were in place.

5. **Executing Atomic Tests:** I utilized the Invoke-AtomicTest function to execute specific atomic tests with MITRE IDs against our local Open RAN components. This function simulated various attack scenarios and assessed the effectiveness of our security controls using the MITRE ATT&CK matrix.

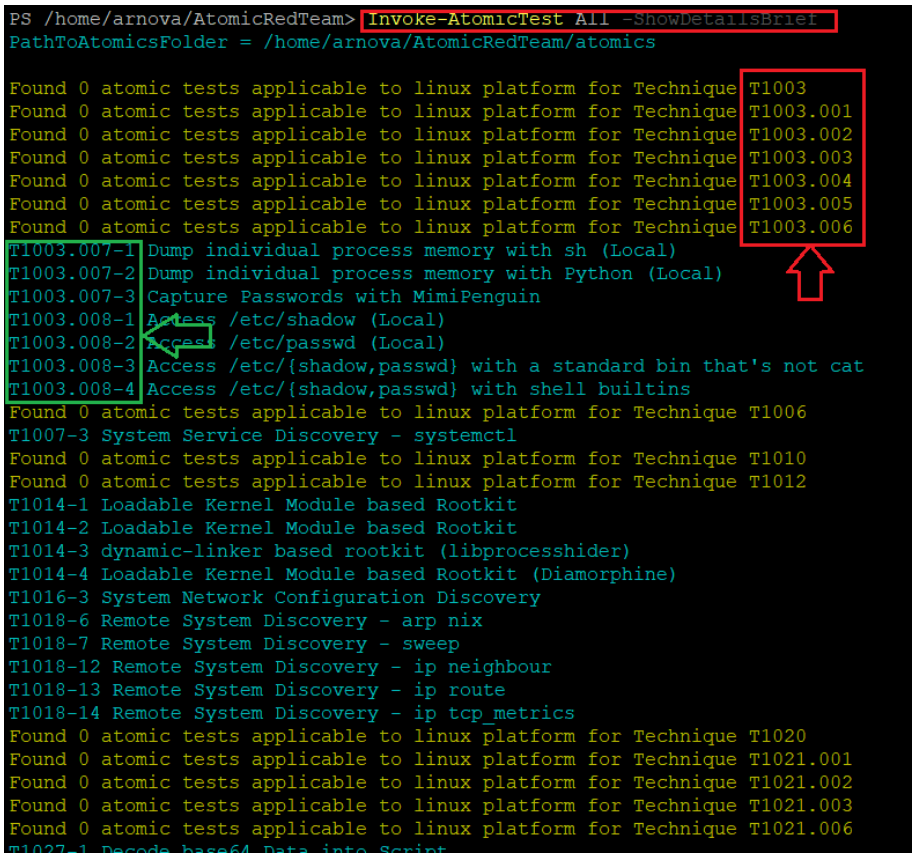

Figure 15: Available atomic tests list in the InvokeAtomicRedTeam framework.

6. **Cleanup:** After executing the atomic tests, I performed the necessary cleanup procedures to restore the systems to their original state and remove any artifacts or traces generated during the testing process.

As shown in the table 2, one CVE ID is mapped to 2 or more MITRE IDs. As Open RAN has been implemented in the Ubuntu Linux OS, some atomic tests mapped were not available due to the limitation which is displayed in Figure 16. Only a few MITRE IDs from a single CVE ID have been selected for test purposes. In the next section, one atomic test result will be analyzed.

4.2.3 Atomic Test Result Analysis

Atomic test T1056.001: MITRE ID T1056.001 refers to the technique of keylogging, where adversaries capture user keystrokes to intercept credentials and gain unauthorized access enabling them to collect sensitive information, such as usernames, passwords, and other credentials, which can provide them with new access opportunities within a targeted system or network. Keylogging is commonly used when other methods of credential dumpings, such as OS Credential Dumping, are not effective. It involves capturing keystrokes over a period of time until valuable credentials are successfully acquired. There are several atomic tests related to input capture and keylogging technique (T1056.001).

- Atomic test 1: Input Capture
- Atomic test 2: Living off the Land Terminal Input Capture on Linux with pam.d
- Atomic test 3: Logging bash history to syslog
- Atomic test 4: Bash session based keylogger
- Atomic test 5: SSHD PAM keylogger
- Atomic test 6: Auditd keylogger
- Atomic test 7: MacOS Swift Keylogger

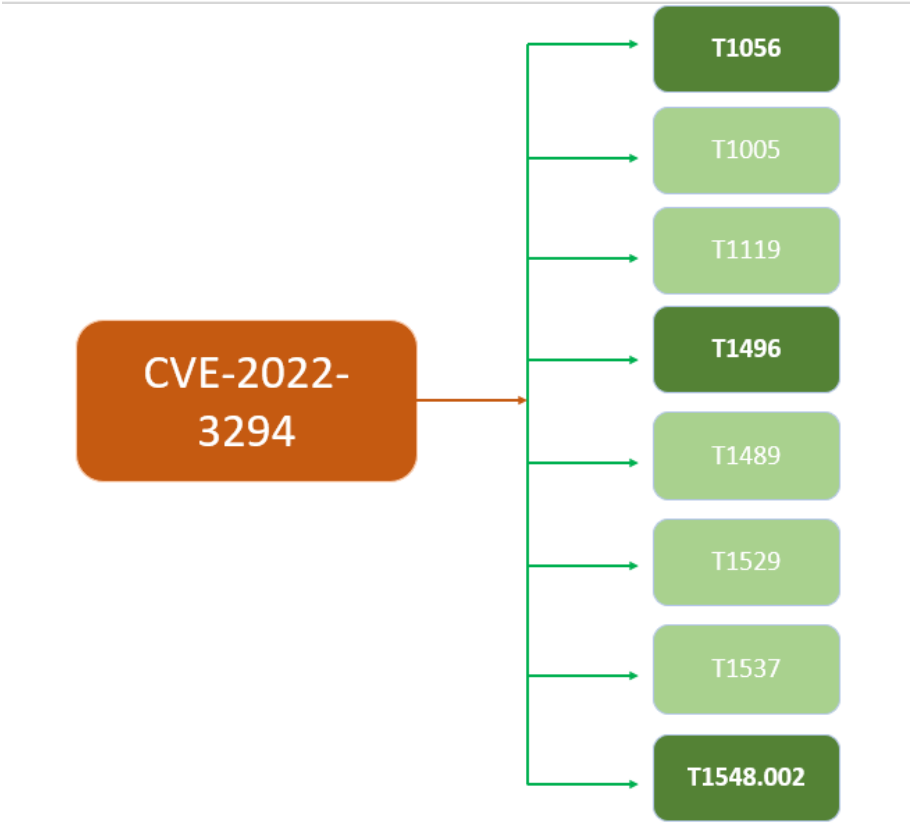

Figure 16: Available atomic tests [dark green box] and unavailable atomic tests [light green box] for the ID CVE-2022-3294.

As an example, the atomic test T1056.001, test number 5 result is explained in the following section.

T1056.001 - Atomic test 5 - SSHD PAM keylogger: Atomic Test #5, "SSHD PAM Keylogger," is an atomic test that focuses on evaluating an organization's ability to detect and respond to keylogging activities specifically targeted at SSH authentication using the SSHD PAM configuration. The test simulates an attacker modifying the SSHD PAM configuration to capture and log user keystrokes, including usernames and passwords, during SSH login attempts. By executing this test, organizations can assess their detection capabilities and incident response procedures when dealing with keylogging attacks on SSH authentication. The results of this test help identify any weaknesses in detecting and mitigating SSHD PAM keylogging techniques, allowing organizations to enhance their security controls and response strategies.

T1056.001 - Atomic test 5 - SSHD PAM keylogger execution result: The result can be seen in Figure 17. The error message indicates that there was a permission denied issue when attempting to create or modify the file and the attack did not successfully achieve its objective of implementing the SSHD PAM keylogger. The encountered error suggests that the necessary permissions or authentication requirements were not met to make the modifications required for the keylogging attack. Atomic tests are designed to simulate attack scenarios and assess the detection and response capabilities of an organization's security measures. In this case, the test result indicates a failure to successfully execute the attack, which is generally desirable from a security perspective. It suggests that the system has implemented proper restrictions and safeguards to prevent unauthorized modifications to critical files like "/etc/pam.d/sshd".

T1056.001 - Atomic test 5 - SSHD PAM keylogger execution cleanup: Cleaning up after executing an atomic test is essential to restore the system to its original state, remove any potential security risks or misconfigurations, and optimize resource usage. It helps maintain a secure and controlled environment while minimizing any impact or disruptions caused by the test execution. The cleanup procedure for atomic test T1056.001 is displayed in Figure 18.

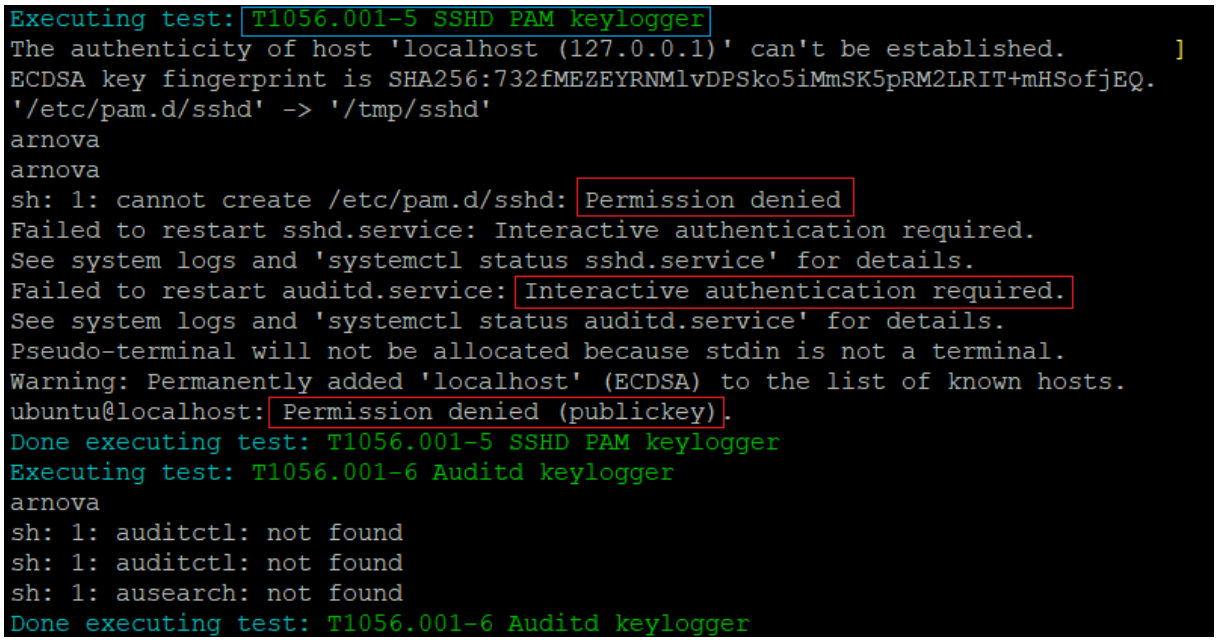

Figure 17: T1056.001 test number 5 execution result

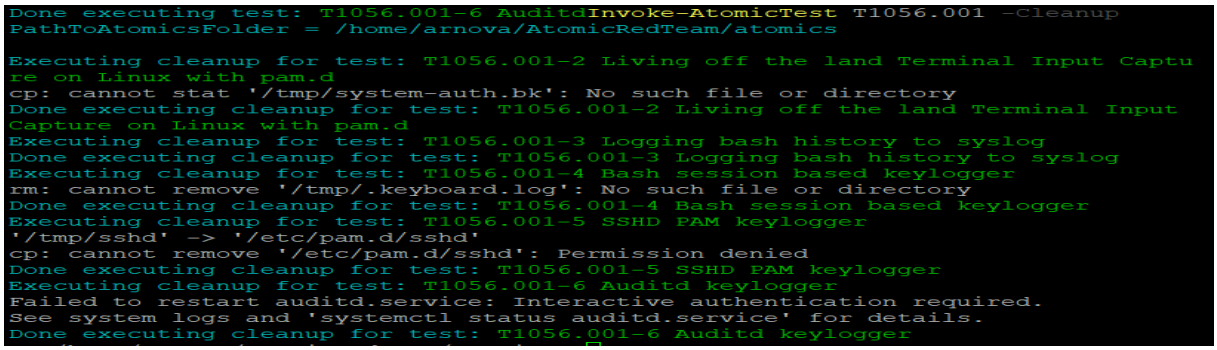

Figure 18: T1056.001 atomic test execution cleanup

Other Atomic tests results: To analyze the security of the implemented Open RAN simulation environment, some other atomic tests were executed. Table 3 displays the executed atomic tests including test ID, name, description, result analysis, and figure references. The table also contains information regarding whether the test is from an inside attacker aspect or not.
From the atomic test result analysis, it can be said that the implemented Open RAN simulation environment in our Lab server is secure against potential attackers inside the same network. Because in most of the cases, atomic tests were unsuccessful to exploit the environment due to properly configured authentication of the 5G Open RAN.

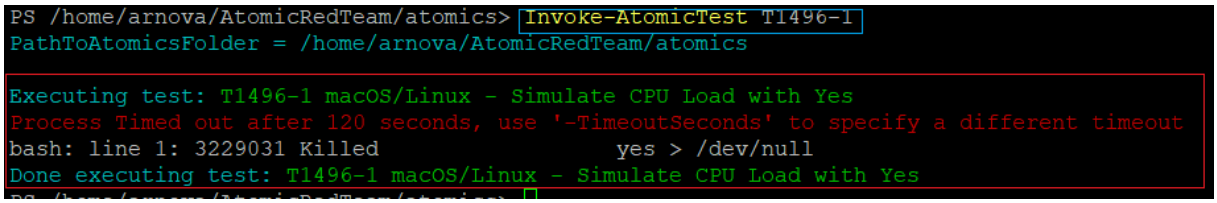

Figure 19: T1496 atomic test execution result.

Test number	**Test name**	**Test description**	**Inside attacker perspective**	**Result summary**	**Result Analysis**	**Figure reference**
T1078.003	Valid Accounts: Local Accounts	Exploit local account credentials for unauthorized access, privilege escalation, and lateral movement	Yes	Successful	Permission denied	20
T1496	Resource Hijacking	Co-opt systems for resource-intensive tasks such as cryptocurrency mining, causing system performance issues and enabling network-based attacks	Yes	Successful	Time out after 120 seconds preventing processes from consuming excessive resources	19
T1529	System Shutdown/Reboot	Initiate system shutdowns or reboots to disrupt access, impede incident response, or expedite the effects of other attacks on system availability	Yes	Successful	Permission denied	21
T1548.001	Abuse Elevation Control Mechanism: Setuid and Setgid	Exploit the setuid and setgid bits in application configurations to execute code with elevated privileges, bypassing restricted permissions and potentially targeting vulnerable binaries	Yes	Successful	Permission denied	22
T1611	Escape to Host	Container breakout allows adversaries to escape from a container and access the underlying host, potentially compromising other resources and escalating privileges within the environment	Yes	Successful	Permission denied	23
T1613	Container and Resource Discovery	Seek to discover containers and related resources in a container environment to gather information for subsequent actions	Yes	Successful	Permission denied	24

Table 3: Executed Atomic tests against 5G Open RAN result summary

```
PS /home/arnova/AtomicRedTeam/atomics> Invoke-AtomicTest T1078.003-8
PathToAtomicsFolder = /home/arnova/AtomicRedTeam/atomics

Executing test: T1078.003-8 Create local account (Linux)
arnova
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
su: user art does not exist
Done executing test: T1078.003-8 Create local account (Linux)
PS /home/arnova/AtomicRedTeam/atomics> Invoke-AtomicTest T1078.003-9
PathToAtomicsFolder = /home/arnova/AtomicRedTeam/atomics

Executing test: T1078.003-9 Reactivate a locked/expired account (Linux)
arnova
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
usermod: user 'art' does not exist
usermod: user 'art' does not exist
usermod: user 'art' does not exist
usermod: user 'art' does not exist
su: user art does not exist
Done executing test: T1078.003-9 Reactivate a locked/expired account (Linux)
```

Figure 20: T1078.003 atomic test execution result.

```
PS /home/arnova/AtomicRedTeam/atomics> Invoke-AtomicTest T1529-3
PathToAtomicsFolder = /home/arnova/AtomicRedTeam/atomics

Executing test: T1529-3 Restart System via `shutdown` - macOS/Linux
Failed to set wall message, ignoring: Interactive authentication required.
Failed to reboot system via logind: Interactive authentication required.
Failed to open initctl fifo: Permission denied
Failed to talk to init daemon.
Done executing test: T1529-3 Restart System via `shutdown` - macOS/Linux
PS /home/arnova/AtomicRedTeam/atomics>
```

Figure 21: T1529 atomic test execution result.

5 Conclusion

In conclusion, this project focused on evaluating open-source attack tools for analyzing the security of a 5G Open RAN simulation environment as an inside attacker aspect. The implemented near real-time RIC was subjected to atomic tests, and the results were analyzed to assess the system's security posture. The comprehensive evaluation was conducted on Kali Linux, Atomic Red Team, Infection Monkey, and Caldera, ultimately selecting Atomic Red Team as the preferred tool for executing the tests. The evaluation of various open-source attack tools showcased their capabilities and suitability for security assessments in the 5G domain. While Kali Linux, Infection Monkey, and Caldera offered unique features, Atomic Red Team stood out as the optimal choice due to its extensive collection of atomic tests specifically designed to assess different attack vectors.

Additionally, as part of this project, a comprehensive research effort was undertaken to leverage an existing GitHub project that provides a mapping between Common Vulnerabilities and Exposures (CVE) identifiers and MITRE IDs. This mapping project, which is maintained by the security community, aims to correlate specific vulnerabilities identified by CVEs to the relevant techniques and tactics documented in the MITRE ATT&CK framework. By utilizing this open-source mapping project, the project was able to enhance the accuracy and relevance of the atomic tests conducted using the selected Atomic Red Team tool. The mapping provided a valuable reference point, aligning specific vulnerabilities with known adversary behaviors and tactics, thus enabling a more targeted and effective evaluation of the 5G Open

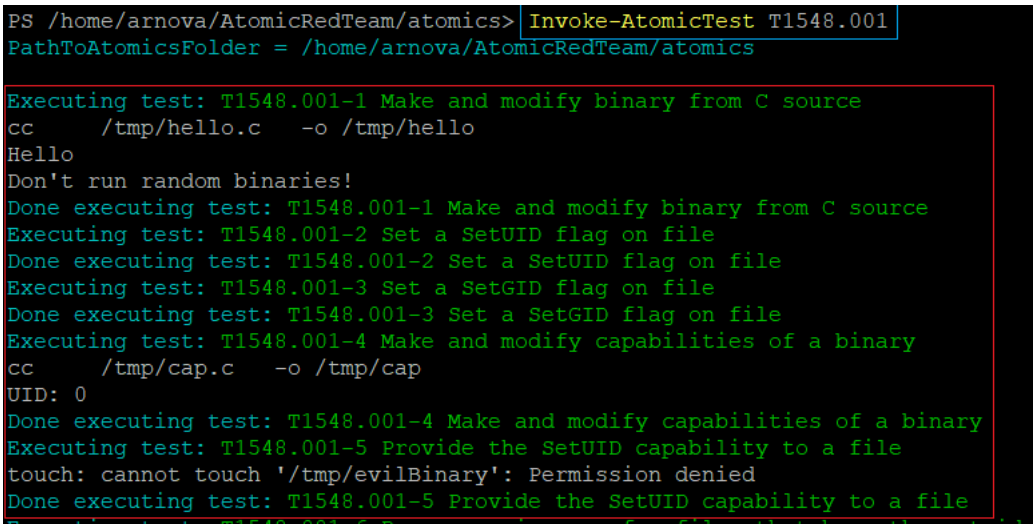

Figure 22: T1548.001 atomic test execution result.

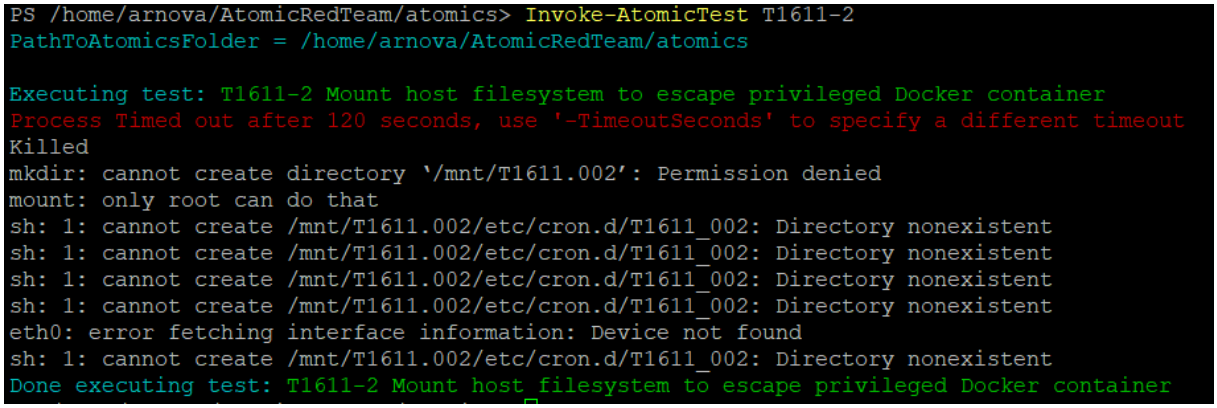

Figure 23: T1611 atomic test execution result.

RAN simulation environment's security.

Through the execution of atomic tests, valuable insights were gained into the vulnerabilities and weaknesses of the near real-time RIC assuming the adversaries inside the network. The analysis of the test results provided a deeper understanding of the system's security strengths and areas that require improvement. This information can guide future security enhancements and help mitigate potential risks in the 5G Open RAN simulation environment.

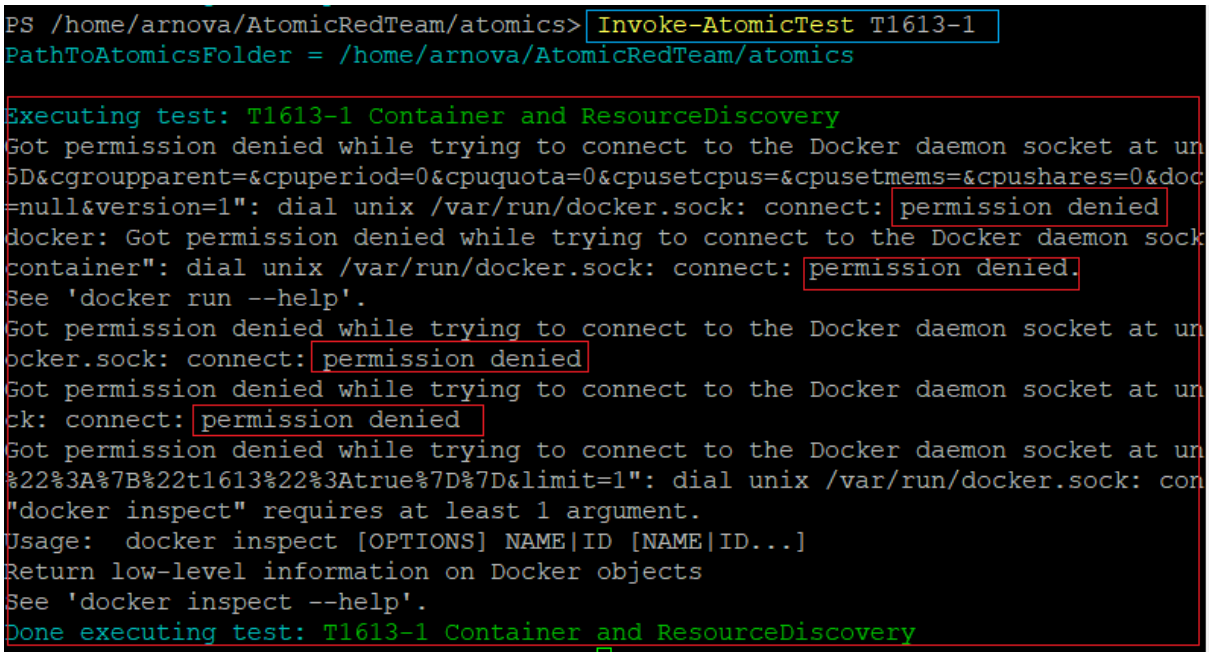

Figure 24: T1613 atomic test execution result.

6 Future Work

Based on the findings and outcomes of the project, several potential areas for future work and research can be identified:

- Multi-Stakeholder Analysis: As this project was limited to only one stakeholder which is as an inside attacker, the evaluation framework can be expanded to include multiple stakeholders involved in the 5G Open RAN simulation environment. The security requirements, concerns, and perspectives of various stakeholders such as network operators, system administrators, third-party vendors, and end-users can be identified and analyzed in the future. The impact of potential attacks from different perspectives, including insider threats, external adversaries, and unintentional security breaches caused by human error could be assessed.

- Integration with Additional Tools: The integration of other open-source security tools and frameworks can be explored that can complement the evaluation process by providing a more holistic view of the security landscape and enable more advanced threat detection and mitigation.

- Security Automation and Orchestration: The implementation of security automation and orchestration techniques can be explored in the future to streamline the evaluation process and improve incident response capabilities. This can involve leveraging tools or frameworks for automated threat detection, response orchestration, and security event correlation, enabling faster and more efficient mitigation of security incidents.

- Diversify Vulnerability Assessment: Extend the scope of the project to include not only Kubernetes but also other critical components such as operating systems (OS) and virtualization technologies commonly used in the 5G Open RAN simulation environment. Incorporate a comprehensive analysis of CVEs specific to these areas, exploring potential vulnerabilities and their impact on the overall security posture.

References

1 O-RAN Alliance. "O-RAN Alliance." Retrieved April 24, 2023, from `https://www.o-ran.org/`

2 Federal Office for Information Security [BSI]. (n.d.). *5G Radio Access Network Risk Analysis*. Retrieved January 31, 2023, from `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/5G/5GRAN-Risk-Analysis.html`

3 CCC Media. (2022). *OpenRAN 5G Hacking Just Got a Lot More Interesting*. Retrieved from `https://media.ccc.de/v/mch2022-273-openran-5g-hacking-just-got-a-lot-more-interesting`

4 M. Z. Chowdhury, M. Shahjalal, S. Ahmed and Y. M. Jang, "6G Wireless Communication Systems: Applications, Requirements, Technologies, Challenges, and Research Directions," in IEEE Open Journal of the Communications Society, vol. 1, pp. 957-975, 2020, doi: 10.1109/OJCOMS.2020.3010270.

5 Y. -j. Choi, K. B. Lee and S. Bahk, "All-IP 4G Network architecture for efficient mobility and resource management," in IEEE Wireless Communications, vol. 14, no. 2, pp. 42-46, April 2007, doi: 10.1109/MWC.2007.358963.

6 D. Pratiwi and I. J. Matheus Edward, "Analysis Efficiency Network Performance of 4G LTE in Video Conference Applications," 2022 16th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Lombok, Indonesia, 2022, pp. 1-6, doi: 10.1109/TSSA56819.2022.10063921.

7 3GPP. "5G System Overview." Retrieved June 5, 2023, from `https://www.3gpp.org/technologies/5g-system-overview`

8 F. Raissi, S. Yangui and F. Camps, "Autonomous Cars, 5G Mobile Networks and Smart Cities: Beyond the Hype," 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Napoli, Italy, 2019, pp. 180-185, doi: 10.1109/WETICE.2019.00046.

9 3GPP. "About Us." Retrieved June 5, 2023, from `https://www.3gpp.org/about-us`

10 M. A. Habibi, M. Nasimi, B. Han and H. D. Schotten, "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," in IEEE Access, vol. 7, pp. 70371-70421, 2019, doi: 10.1109/ACCESS.2019.2919657.

11 Kasparec, F. (2021). Telecom: Virtualizing Radio Access Networks (RAN) - Why and How? LinkedIn. Retrieved June 1st, 2023, from `https://www.linkedin.com/pulse/telecom-virtualizing-radioaccess-networks-ran-why-how-franz-kasparec/`

12 O-RAN Alliance. (2018). O-RAN Alliance. Retrieved June 1st, 2023, from `https://www.o-ran.org/`

13 Open RAN Alliance. (2022). Open RAN Specifications. Retrieved June 1st, 2023, from `https://orandownloadsweb.azurewebsites.net/specifications`

14 Oracle Corporation. (2007). VirtualBox. Retrieved April 20th, 2023, from `https://www.virtualbox.org/`

15 TechTarget. (2023). Virtual Machine (VM) Definition. Retrieved January 1st, 2023, from `https://www.techtarget.com/searchitoperations/definition/virtual-machine-VM`

16 Docker. (n.d.). What is a Container? Docker. Retrieved June 4th, 2023, from `https://www.docker.com/resources/what-container/`

17 Edureka. (n.d.). Kubernetes Tutorial: A Comprehensive Guide for Beginners. Edureka. Retrieved June 4th, 2023, from `https://www.edureka.co/blog/kubernetes-tutorial/`

18 Docker. Retrieved June 4th, 2023, from `https://www.docker.com/`

19 Kubernetes. (n.d.) Kubernetes Documentation. Retrieved June 5th, 2023, from `https://kubernetes.io/`

20 MITRE. (1999). Common Vulnerabilities and Exposures (CVE). Retrieved June 5th, 2023, from `https://cve.mitre.org/`

21 CVE Details. (n.d.). CVE Details. Retrieved June 5th, 2023, from `https://www.cvedetails.com/`

22 MITRE. (n.d.). MITRE ATT&CK. Retrieved June 5th, 2023, from `https://attack.mitre.org/`

23 Kali Linux. (n.d.). Retrieved June 5th, 2023, from `https://www.kali.org/`

24 Atomic Red Team. (n.d.). Retrieved June 5th, 2023, from `https://atomicredteam.io/`

25 Red Canary. (n.d.). Invoke-AtomicRedTeam. Retrieved June 5th, 2023, from `https://github.com/redcanaryco/invoke-atomicredteam/wiki`

26 MITRE. (n.d.). Caldera. GitHub. Retrieved June 5th, 2023, from `https://github.com/mitre/caldera`

27 Akamai. "Infection Monkey." Retrieved June 5th, 2023, from `https://www.akamai.com/infectionmonkey`

28 Vulmon. "Vulmap: Open Source Online Local Vulnerability Scanners Project." GitHub, 2023, Retrieved April 20th, 2023, from `https://github.com/vulmon/Vulmap.`

29 Center for Threat-Informed Defense. (2021). attack_to_cve [GitHub repository]. Retrieved June 5th, 2023, from `https://github.com/center-for-threat-informed-defense/attack_to_cve`