5G Trusted And seCure network servICes

Grant Agreement no. 101127973

Deliverable D3.2

5G building blocks and security analysis, evaluation and testing

Programme:	DEP
Project number:	101127973
Project acronym:	5G-TACTIC
Start/End date:	12/2023-12/2026

Due date:	31/5/2025
Actual submission date:	30/6/2025
Responsible Editor:	ACC
Related WP(s):	WP3
Contact person:	Simon Pryor

Dissemination level:	Public
Nature:	Deliverable
Version:	V1.0
Number of pages:	90

Author List:

Organization	Author
ACC	Simon Pryor
IASA	Markos Anastasopoulos, Thanassis Economou, Viktoria Alevizaki, Ilias Floudas, Alexandros Manolopoulos, Anna Tzanakaki
UCAN	Alberto Eloy García, Luis Diez, Ramón Agüero
ORO	Razvan Mihai, Cristian Patachia
OTE	Ioanna Mesigiti, Elina Theodoropoulou
BBRAN	Navid Nikaein, Alessandro Pacini
NASK	Mateusz Krzysztoń, Bartosz Wysocki, Diana Starodąb, Paweł Kostkiewicz, Michał Marks, Bartosz Bok
DNSC	Valeria Popescu, Valentin Constantinescu, Cristian Nistor, Robert Șandru and Ștefan Tănase
DV	Guillaume Pelletier
DSA	Sotiris Koullapis, Constantinos Fellas, Charalambos Oxinos

Keywords: 5G Cybersecurity Toolbox, 5G-RAN, 5G Core, Open RAN, Vulnerability assessment, eBPF, deep packet inspection, jamming attack RIC, xAPPs

Table of Contents

List of Figures

List of Abbreviations

Acronym	Description
3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
APT	Advanced Persistent Threat
BPF	Berkeley Packet Filter
CI/CD	Continuous Integration / Continuous Deployment
CNF	Cloud-Native Network Function
CRA	Cyber Resilience Act
CSIRT	Computer Security Incident Response Team
CTI	Cyber Threat Intelligence
CU	Centralized Unit
CUPS	Control and User Plane Separation
CVE	Common Vulnerabilities and Exposures
CP	Control Plane
DDoS	Distributed Denial of Service
DIAMETER	DIAMETER Protocol (authentication, authorization, and accounting)
DL	Downlink
DOA	Direction of arrival
DoS	Denial of Service
DU	Distributed Unit
DUT	Device Under Test
E2AP	E2 Application Protocol
E2E	End-to-End

E2ESO	End-to-End Security Orchestration
E2SM-RC	O-RAN E2 Service Model: *RAN Control*
E2SM-KPM	O-RAN E2 Service Model: Key Performance Measurement
eBPF	Extended Berkeley Packet Filter
eCPRI	Enhanced Common Public Radio Interface
ENISA	European Union Agency for Cybersecurity
EPS	Evolved Packet System
ETSI	European Telecommunications Standards Institute
EU	European Union
FH	Fronthaul
FTPES	FTP over Explicit TLS
GDPR	General Data Protection Regulation
GTP	GPRS Tunneling Protocol
gNB	Next Generation Node B (5G base station)
gRPC	Google Remote Procedure Call
IETF	Internet Engineering Task Force
IEC	International Electrotechnical Commission
IDS	Intrusion Detection System
IKE	Internet Key Exchange
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
JSON	JavaScript Object Notation

LSM	Linux Security Module
LSTM	Long Short-Term Memory
LVMM	Live Virtual Machine Migration
MAC	Media Access Control
MACsec	Media Access Control Security (IEEE 802.1AE)
MANO	Management and Orchestration
M-Plane	Management Plane
MFA	Multi-Factor Authentication
MISP	Malware Information Sharing Platform
MITM	Man-in-the-Middle (Attack)
ML	Machine Learning
MNO	Mobile Network Operator
MTD	Moving Target Defence
N3IWF	Non-3GPP InterWorking Function
NAT	Network address translation
NEF	Network Exposure Function
NESAS	Network Equipment Security Assurance Scheme
NIS2	Network and Information Security Directive 2
NIST	National Institute of Standards and Technology
NF	Network Function
NFV	Network Function Virtualization
NFV-SEC	Network Functions Virtualization – Security
NG-AP	Next-Generation Application Protocol
NG-RAN	Next-Generation Radio Access Network

NR	New Radio
NRF	Network Repository Function
NTIA	National Telecommunications and Information Administration
OAI	Open Air Interface
OAuth	Open Authorization
O-CU	O-RAN Centralized Unit
O-DU	O-RAN Distributed Unit
OFDM	Orthogonal frequency-division multiplexing
O-RAN	Open Radio Access Network
O-RU	O-RAN Radio Unit
O-Cloud	O-RAN Cloud
OPA	Open Policy Agent
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
PBCH	Physical Broadcast Channel
PTP	Precision Time Protocol
QoS	Quality of Service
QUIC	Quick UDP Internet Connections
RAN	Radio Access Network
RBAC	Role-Based Access Control
REST	Representational State Transfer
RFC	Request for Comments (IETF document)
RF	Radio Frequency
RIC	RAN Intelligent Controller

RL	Reinforcement learning
RTP	Real-time Transport Protocol
RU	Radio Unit
S-Plane	Synchronization Plane
SBA	Service Based Architecture
SBIs	Service-Based Interfaces
SBOM	Software Bill of Materials
SCAS	Security Assurance Specifications
SCTP	Stream Control Transmission Protocol
SDN	Software Defined Networking
SDR	Software Defined Radio
SECAM	Security Assurance Methodology
SELinux	Security-Enhanced Linux
SEPP	Security Edge Protection Proxy
SIEM	Security Information and Event Management
SIP	Session Initiation Protocol
SMO	Service Management and Orchestration
SOC	Security Operations Center
SOAR	Security Orchestration, Automation, and Response
SPDX	Software Package Data Exchange
SSH	Secure Shell
SSDLC	Secure Software Development Life Cycle
SSB	Synchronization Signal Block
SUT	System Under Test

SWID	Software Identification Tag
TAXII	Trusted Automated Exchange of Indicator Information
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TS	Technical Specification
UE	User Equipment
UL	Uplink
UPF	User Plane Function
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network
xApp	RIC Application (for Near-RT RIC)
XDP	eXpress Data Path
ZTA	Zero Trust Architecture

1. Executive Summary

5G-TACTIC focuses on the development of secure Open-Radio Access Network (RAN) solutions addressing some of its fundamental challenges such as security, integration, interoperability & disaggregation. As such it clearly promotes the relevant deployments enabling the capability to integrate multi-vendor Open 5G RAN platforms in commercial environments with significant benefits in the EU supply chain and the potential to release MNOs from single or limited vendor lock-in. To achieve this, the present deliverable uses as input the output of WP2 specification (Security Assurance Specifications (SCAS)) and the evaluation methodology (3GPP Security Assurance Methodology (SECAM)) defined in D2.3 in order to analyse, test and assess cybersecurity aspects of Open RAN (O-RAN) solutions provided by ACC and BBRAN, open-source RAN (OpenaAirInterface (OAI)) and Core Networks (CNs) (OAI, free5GC, Open5GS) implementations. The overall work is divided into two parts: Part I focusing on the RAN and Part II on the CN. For the RAN segment, the initial tests performed investigate vulnerabilities and interoperable operations. These include:

- Interoperability testing across multi-vendor RAN and CN functions. To achieve a combination of commercial and open source platforms are deployed testing the performance of service slices offered over multi-vendor RAN networks. Availability and integrity of gNBs under varying loading conditions as well as termination of the operation of gNB functions components under unexpected data inputs is performed.
- Testing security functions that may lead to failure in ensuring control plane data. Specifically, RAN emphasis is given on the security analysis of the O-RAN E2 protocol and A1 exposing interfaces to Near Real Time (RT) Radio Intelligent Controller (RIC) and Non Real Time (NRT)-RIC controllers.
- Vulnerable mechanisms for authentication and authorization of gNB, vulnerabilities in the N2 interface, unauthorized access to the RIC controller and improper management of authentication policies.
- Mechanisms protecting RAN elements against physical layer attacks. To achieve this, spectrum monitoring mechanisms of the RAN have been developed which are combined with novel defense applications running in the RIC controllers protecting the network against jamming attacks.
- Compute domain related vulnerabilities (virtualization, web services. hardware). These are addressed through the deployment of a deep packet inspection system monitoring RAN functions

For the core segment, the activities focused on the identification of vulnerabilities related to security functionalities of the Core Network elements. The SECAM methodology is applied to different functions to explore vulnerabilities and then develop solutions for the:

- Access and Mobility Management Function (AMF) targeting incorrect implementation of mechanisms such as authentication and key agreement procedures. The security evaluation focuses on N2 interface, as this interface employs the Stream Control Transmission Protocol (SCTP). Specifically, the N2 interface operates between the gNB and the AMF entity within the core network. Vulnerability assessment was performed on all three 5G core network implementations (Free5GC, Open5GS, and OpenAirInterface), with each evaluation producing identical outcomes regarding network behavior, frame exchange patterns, and attack impact on the core network entities.
- User Plane Function (UPF) assessing vulnerabilities in user data and signalling protocols. To achieve this, an experimental framework has been developed launching Denial of Service (DoS) and SYN flood attacks targeting the UPF interfaces. These attacks are then used to assess resilience and performance of UPF nodes under stress.
- Security functionalities that involve other Core Network elements. These vulnerabilities are detected using deep packet inspection strategies which continuously monitor information flows in the service-based interfaces. This information combined with suitable AI models can detect anomalies and then apply strategies to mitigate risks (remove users from the Unified Data Repository (UDR)).

- Collecting, storing, distributing and sharing cyber security indicators and threats regarding cyber security incidents detected in RAN and CN elements.

2. Introduction

The European Union (EU) has identified digital technologies as essential tools for addressing core societal and economic challenges. These include enhancing citizens' quality of life, promoting sustainable development, and driving economic growth. In this context, the global rollout of 5G technologies presents new opportunities for governments, businesses, and individuals by enabling connected products, collaborative and automated processes across various sectors, and optimized resource and energy usage. It also opens the door to improved services related to safety, security, resilience, and user experience.

To ensure widespread adoption of 5G in society, the EU has made the timely and secure deployment of 5G networks a top priority. To support this, National Authorities (NAs) from multiple EU member states, in collaboration with the European Commission and the EU Agency for Cybersecurity (ENISA), have developed a coordinated framework for identifying potential risks that could severely impact the large-scale deployment of 5G networks. Key risk areas include:

- Network Security Risks: These involve inadequate security measures such as network misconfigurations or lack of access controls. Such vulnerabilities could expose critical 5G components through interfaces accessible to unauthorized users, leading to breaches of confidentiality, integrity, or availability.
- Supply Chain Risks: These stem from poor-quality products integrated into operational mobile networks, which may degrade service performance. Additionally, heavy reliance on a single supplier increases vulnerability and the risk of supply disruptions.
- Interdependency Risks: These arise from the close links between 5G networks and other critical infrastructures. Such interdependencies can lead to cascading failures due to both intentional (e.g., cyberattacks, sabotage) and unintentional (e.g., natural disasters causing power outages or cable cuts) events.
- End-User Risks: These involve malicious exploitation of system resources by hackers who can bypass device security and initiate network attacks.

To mitigate these risks, 5G-TACTIC assists in the development of the 5G cybersecurity toolbox through solutions that

- Enhance security requirements for Mobile Network Operators (MNOs). This is achieved through investigation of mechanisms applied in Open 5G products ensuring strict access control, secure operation and monitoring.
- Apply policies that limit access for key assets defined as critical and sensitive in the EU coordinated risk assessment. This will include core network functions, network management and orchestration functions, and access network functions.
- Facilitate deployment of multi-vendor 5G mobile networks avoiding dependency on high-risk suppliers or situations of lock-in with a single supplier, which clearly introduces the need to promote increased interoperability of equipment.

To achieve these goals, the present deliverable leverages the specifications from WP2 (Security Assurance Specifications - SCAS) and the evaluation methodology (3GPP Security Assurance Methodology - SECAM) defined in **D3.1** to analyze, test, and assess the cybersecurity aspects of various Open RAN solutions. These include implementations provided by ACC, BBRAN, and open-source platforms such as **OpenAirInterface (OAI)** for both Radio Access Network (RAN) and Core Network **(**CN) components (including **free5GC** and **Open5GS**).

The work is structured into two main parts:

- **Part I**: Focused on the **RAN** segment
- **Part II**: Focused on the **CN** segment

Initial testing in the RAN targeted gNB vulnerabilities, specifically:

- Improper implementation of security functions, which may compromise control plane data confidentiality over the N2 interface, and inadequate enforcement of security policies or encryption mechanisms—especially concerning the A1 and E2 protocols used for interfacing with Real Time (RT)-Radio Intelligent Controller (RIC) and Non Real Time (NRT)-RIC controllers.
- Weak protection of gNB component data, including vulnerabilities in the secure storage and transfer of sensitive information.
- Availability and integrity issues under stress conditions, such as high loads or malformed data inputs, potentially leading to functional disruptions.
- Flawed authentication and authorization mechanisms for gNBs, including vulnerabilities in the N2 interface, unauthorized access to RIC controllers, and mismanagement of authentication policies.
- Monitoring capabilities of gNB components, assessing how effectively they provide accurate system status information to both the core network and RIC controllers.
- Compute domain vulnerabilities, including weaknesses related to virtualization, web services, and hardware security.

For the CN, the focus was on identifying vulnerabilities in the security functionalities of core network elements. The SECAM methodology was adapted and applied to various functions to detect issues and propose mitigation strategies:

- Access and Mobility Function (AMF): Assessing the robustness of authentication, key agreement mechanisms, and intra-RAT mobility procedures.
- User Plane Function (UPF): Evaluating security and scalability of user data handling and signaling protocols.
- Unified Data Management (UDM) and Unified Data Repository (UDR): Focusing on integrating response mechanisms with UDM/UDR rand applying suitable policies to isolate attacks.
- Service-Based Architecture (SBA) / Service-Based Interfaces (SBI): Investigating vulnerabilities in 5G Core components with emphasis on access protection and secure data transmission. This includes ensuring appropriate implementation of transport layer security, as well as authentication and authorization mechanisms for SBI Network Functions (NFs) applying deep packet inspection strategies.

The remainder of this document is summarized as follows: Section 3 provides a high-level overview of the overall 5G-TACTIC System Architecture. Section 4 describes and discusses the 5G-TACTIC Security testing methodology. Section 5 focuses on Security Testing and presents a set of relevant and representative testing scenarios addressing different architectural domains. Vulnerability assessment for the RAN and CN functions along with some tools developed to mitigate identified vulnerabilities and threats is presented in Section 6 and Section 7, respectively, while Section 8 concludes the document.

3. 5G-TACTIC Reference Architecture

As discussed in detail in D2.3 "5G-TACTIC End-to-End Security Architecture" the main objective of 5G-TACTIC is to support public (market surveillance authorities/notifying authorities/national accreditation/ Security Incident Response Teams (CSIRTs)) and private organisations (manufacturers of 5G elements, equipment vendors) in the development of the 5G cybersecurity toolbox providing guidance for the selection of measures which can be prioritized in mitigation plans at national and at European Union level.

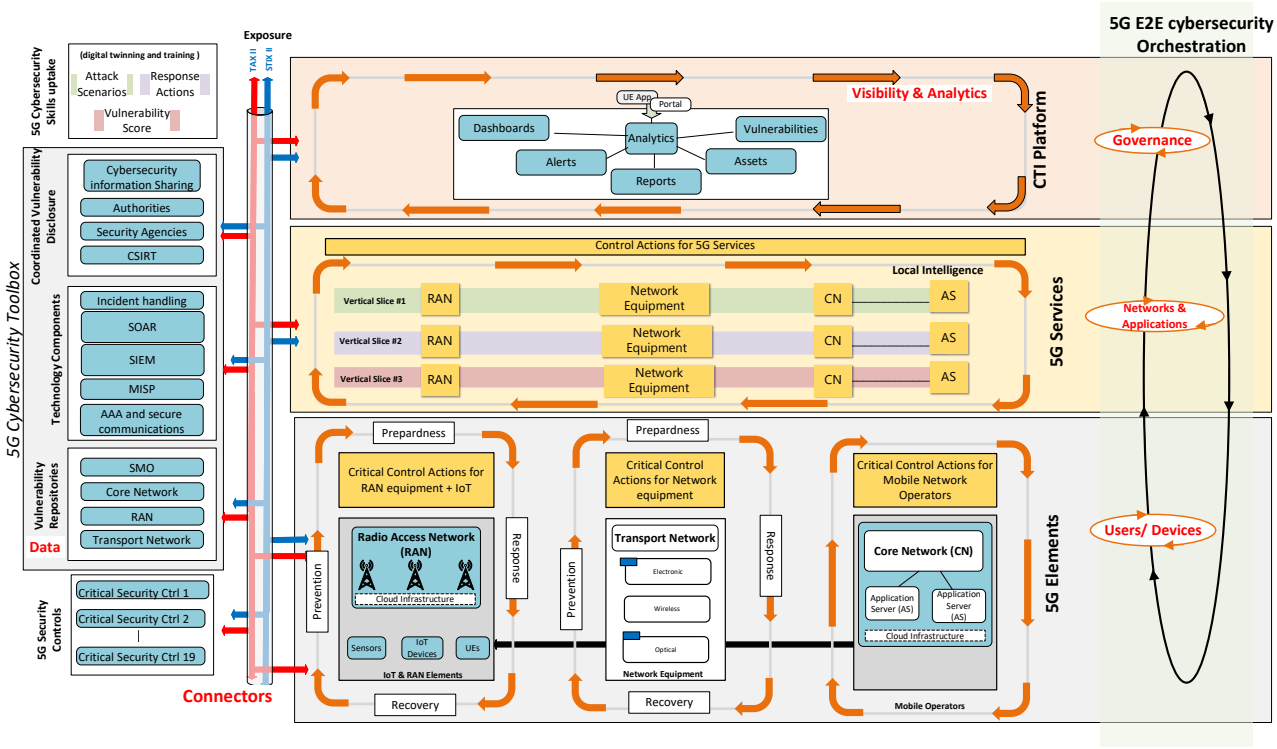

Figure 1 5G-TACTIC overall architecture covering 5G devices and services

In this context, 5G-TACTIC is designing and developing a cross-layer/cross -domain Cyber Threat Intelligence (CTI) sharing platform covering in a complementary manner the application scopes of both Cyber Resilience Act (CRA) and Network and Information Systems Directive 2 (NIS2) as shown in Figure 1. This architecture builds on top of the Zero Trust Architecture (ZTA) which has been expanded to include all enterprise assets and subjects covered by the 5G ecosystems including end-users, 5G devices, infrastructure elements, cloud components, virtualized services and applications.

To enable compliance with CRA and NIS2 in a unified manner, the 5G-TACTIC architecture adopts multiple control loops located at different layers, exploiting emerging technologies such as big data analysis, analytics, and Artificial Intelligence (AI) mechanisms to provide cybersecurity lifecycle management for devices and services. These control loops apply appropriate Critical Control Actions as defined by ETSI TS 103 992[1] and ETSI TR 103 305[2] in order to prevent, prepare, respond, and recover their managed domains from cybersecurity incidents[3]. As 5G-TACTIC relies on ZTA no implicit trust is assumed, therefore the closed-loop operations instantiated at every critical component of the system continuously analyse and evaluate the risks to its assets and business functions. Following this, suitable policies and algorithms apply the necessary protections to mitigate these risks. In zero trust, these protection mechanisms usually try to *minimize the attack surface restricting access to resources* (such as the execution of system processes, file access, networking resources, minimize coverage area etc.) to those that are necessary for the execution of services.

[1]ETSI TS 103 992 V1.1.1 (2024-05)
[2] ETSI TR 103 305-1 V4.2.1 (2024-10)
[3] Best Practices for Cyber Crisis Management, ENISA 2024, URL

To prevent and prepare the 5G system against possible cyberattacks, 5G-TACTIC extends and deploys cybersecurity tools ensuring that each component of the system complies with established functionalities and protections. To achieve this, local cybersecurity tools targeting the 5G-RAN, the core network, the cloud infrastructure etc. are combined with well-known open-source platforms and tools. These ensure that all products and services deployed in the 5G-TACTIC ecosystem:

- have the necessary authorization and authentication mechanisms
- minimize the attack surface providing role-based access control to compute/network resources
- implement confidentiality and integrity checking algorithms on all network interfaces
- support encryption mechanisms
- support mechanisms for cybersecurity attack prevention and recovery
- implement auditable security log generation and reports
- expose their resources to an external security information and event management system, which analyses these reports against multiple vulnerability databases and repositories listing possible threat scenarios that can potentially affect the 5G networks.

Specifically, detected cyberattacks and vulnerabilities against 5G elements are reported in the "vulnerability disclosure marketplace[4]" which in the context of 5G-TACTIC provides the following services:

- ***aggregation of*** vulnerabilities from various sources including product vendors, government agencies, CSIRTs, service providers and external users (i.e., hacker conferences)
- ***storage*** to sector/organisational, and provider/vendor repositories
- ***exposure of the identified*** vulnerabilities to the target entities (authorities, vendors, international CSIRTs etc.) in a coordinated manner.

Exchange of information among all interested parties is performed through vulnerability discovery tools and sharing protocols (such as Structured Threat Information Expression (STIX) and Trusted Automated eXchange of Intelligence Information (TAXII)) used to exchange cybersecurity information in a secure and structured way.

[4] ETSI TR 104 003 V1.1.1 (2024-09) , Cyber Security (CYBER); The vulnerability disclosure ecosystem

4. Analysis, Testing & Assessment Methodology

4.1. Overview of Evaluation Processes

The security assurance framework for 5G networks encompasses multiple layers and evaluation processes, critical for safeguarding the integrity of modern telecommunications. Network Equipment Security Assurance Scheme (NESAS) provides a common framework for assessing network equipment and vendor security [1] . Multi-layer security assurance is crucial for 5G systems, involving new evaluation methodologies and cooperation between applications, networks, and infrastructures [2] . End-to-End security is also essential in 5G, addressing authentication, integrity, key management, and confidentiality while considering attack resistance in protocol design [4] . Additionally, a comprehensive security and trust framework for virtualized networks and software-defined networking in 5G has been proposed, incorporating adaptive trust evaluation, sustainable trusted computing, and cloud-based security services [3] . These approaches collectively ensure robust security measures across various 5G building blocks and applications.

Towards this end, the SECAM evaluation process is designed to ensure that 5G building blocks meet stringent security requirements, adapting to the evolving landscape of telecommunications. This process involves several key steps that are critical for maintaining the integrity and security of 5G networks [6] ,[5] . These steps are outlined as follows:

1. **Preparation**: This is the initial phase where the scope of evaluation based on SCAS specifications is defined. This includes identifying which network functions and components will be assessed, as well as establishing criteria for success. The preparation phase also involves gathering relevant documentation, including existing security policies, compliance requirements, and technical specifications that will guide the evaluation process [7] ,[8] .
2. **Testing**: In this phase various tests are conducted to assess compliance with security requirements, including vulnerability assessments and functionality tests. During vulnerability tests potential weaknesses in the network infrastructure that could be exploited by malicious actors are identified. During functionality tests, all components are ensured to operate as intended and that security protocols are adhered to. Finally, tests can include automation via relevant tools to conduct continuous testing throughout the product lifecycle, which is essential given the rapid pace of updates and changes in 5G networks. Automation can significantly enhance efficiency and accuracy in testing, allowing more frequent assessments without compromising quality [9] , [10] .
3. **Reporting**: After testing is complete, detailed evaluation reports are generated with relevant results. These include detailed evaluation reports that document (a) compliance levels i.e. assessment of how well each component meets SCAS requirements, (b) vulnerabilities identified, where a comprehensive list of vulnerabilities discovered during testing is included, and categorized by severity and potential impact, and (c) recommendations for remediation, where actionable suggestions for addressing identified vulnerabilities and improving overall security posture are performed [7] , [1] , [3] .
4. **Continuous Improvement**: This is a continues for all lifecycle, and thus this phase refers to the ongoing adaptation to methodologies as necessary to address new threats or changes in technology. It includes (a) feedback loops, where lessons learned from previous evaluations are incorporated to refine methodologies and improve future assessments, (b) updates to SCAS, which may need to be updated to reflect current best practices and emerging threats as new vulnerabilities are discovered or as technology evolves, and (c) integration with Continuous Integration/Continuous Deployment (CI/CD) workflows, where SECAM methodologies are incorporated to ensure security evaluations keep pace with development cycles in 5G environments [9] , [10]

4.1.1. Methodology Extensions

In light of the dynamic nature of 5G technology and its associated risks, certain generic methodological extensions may be necessary within the SECAM evaluation framework:

- **Increased Automation:** The shift towards automated testing processes is crucial for managing the frequency and complexity of updates in 5G networks [10] . Automated tools can provide real-time feedback and facilitate continuous monitoring, significantly enhancing security assurance efforts. Automation also allows for comprehensive testing across multiple network functions simultaneously, improving efficiency and reducing time to market for new services.
- **Enhanced Collaboration with Stakeholders:** Engaging with various stakeholders—including vendors, operators, and regulatory bodies—throughout the evaluation process can foster a more comprehensive understanding of security requirements and challenges. This collaboration is essential for aligning security measures with industry best practices and ensuring that all parties are informed about emerging threats and vulnerabilities [4] ,[1]
- **Adaptation to Emerging Threats:** As new vulnerabilities are identified in the evolving threat landscape, SECAM methodologies must adapt to incorporate these insights, ensuring that evaluations remain relevant and effective. This includes staying updated with threat intelligence reports from organizations like ENISA and integrating findings from ongoing research into security assessments [9] .
- **Integration of Industry Standards:** Aligning SECAM methodologies with established industry standards such as those from 3GPP, ETSI, and NIST can enhance the credibility and acceptance of the evaluation process. By incorporating these standards, SECAM can ensure that its evaluations are consistent with global best practices in cybersecurity [1] , [4] .
- **Cloud-Based Evaluation Frameworks:** Given the complexity of 5G architectures, adopting cloud-based two-level frameworks for performance evaluation can help address challenges associated with system-level simulations. This demand is even more eager considering the fact that all infrastructures are host in the cloud. This approach allows for scalable testing environments that can simulate real-world conditions more effectively [7]
- **Focus on Key RAN Enablers:** The evaluation process should emphasize assessing key RAN enablers such as interference management, traffic steering, and network slicing. These components are critical for ensuring optimal performance and security within 5G networks [11] .
- **Continuous Feedback Mechanisms:** Establishing continuous feedback loops between evaluation teams and stakeholders can facilitate ongoing improvements in methodologies. This iterative process ensures that lessons learned from previous evaluations inform future assessments [9] .

By implementing these methodological changes, the SECAM evaluation framework enhances its effectiveness in addressing current and future security challenges in the rapidly evolving 5G ecosystem. These adjustments improve the robustness of security assessments and foster greater trust among stakeholders in the telecommunications landscape.

4.2. 5G-TACTIC Security Assurance Approach

As already discussed, the overall Security Assurance Process to be followed by 5G-TACTIC will be in accordance to the SCAS approach. This approach first includes the description of the network element (HW and SW) under evaluation with sufficient detail to ensure that the security requirements can clearly describe what data and functions are intended to be protected and which functionalities are required. Then possible security issues are defined. This involves definition of the security objectives for the network element under analysis (i.e., which assets require what type of protection) and the possible threats that the system under

test is supposed to resist to. This step also contains the threat analysis employed to understand how the identified threats may affect the system under test. Following this, the security requirements and test cases are derived. Once the security requirements have been identified it is verified whether the security objectives are satisfied by these security requirements. If any mismatch is found (e.g. security objective not covered through the existing security requirements or security requirements which don't resolve any security objectives), the list of security requirements needs to be updated accordingly by removing or adding security requirements. Finally, a set of use cases will be defined to be used for the system level evaluation.

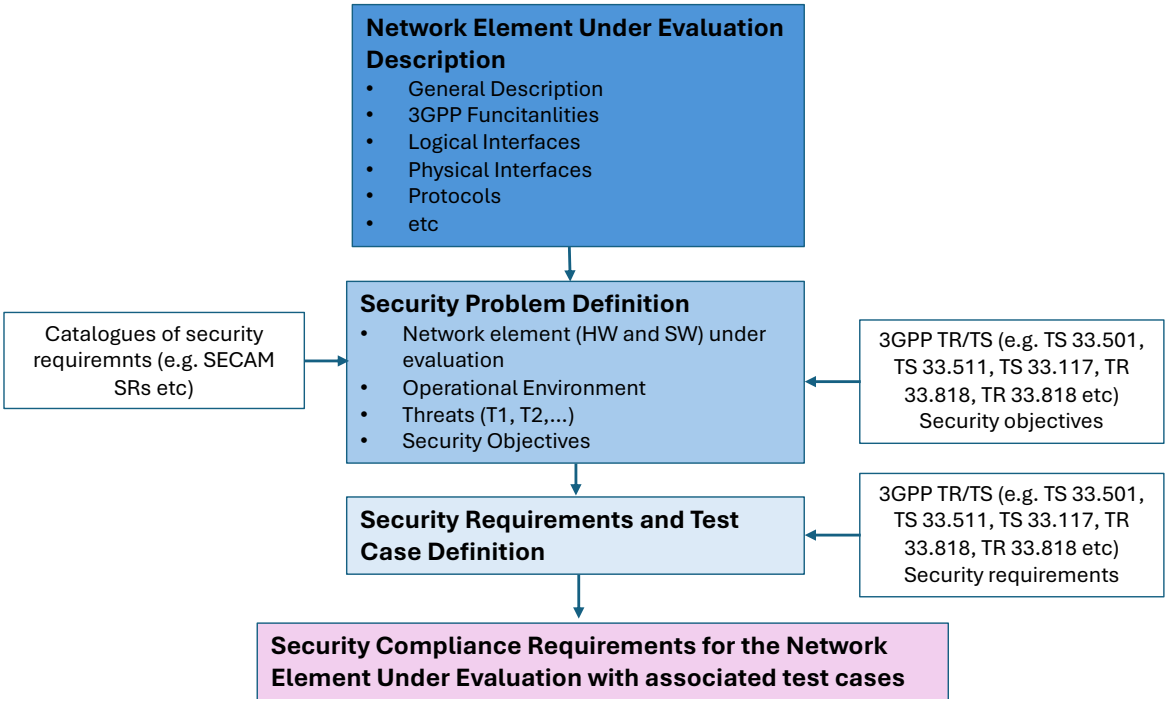

Figure 2: Process for deriving SCAS security compliance requirements

The work performed in the context of T3.2 leverages on the work performed in the context of T3.1, and reported in Deliverable D3.1. In particular, it takes as input the 5G-TACTIC Security Assurance Methodology which (1) adhered to the 3GPP SECAM and the cybersecurity toolbox developed by ENISA, and (2) leveraged on the 3GPP SCAS specifications and the O-RAN security specifications. The proposed framework (ref. D3.1) aimed at developing a holistic security assurance methodology -in terms of processes- addressing the 5G-TACTIC overall architectural approach and building blocks.

The developed methodology aimed to fill in the gaps of existing frameworks, and underscored the necessity of tailoring the (3GPP) SECAM methodology to address the security challenges specific to the 5G-TACTIC technical approach. More specifically, it provided an analysis of the application of SECAM in 5G and beyond network architectures and identified the security requirements (including the ones defined by ENISA) that necessitate extensions and reconsiderations of the methodology to address vulnerabilities and capabilities specifically related to O-RAN, virtualization of RAN and CN functions, SDN and relevant deployment environments. Special focus was given in multi-vendor environments. The proposed methodology considered not only the complexity of 5G-Ecosystems in terms of technologies and integration of multi-vendor environments, but also the number of actors-roles-stakeholders (which contributes to the increase of the security surface) that undertake ownership and responsibility of operating technologies, segments and domains in 5G deployments.

For this purpose, the security assurance methodology/ processes were defined -at high level- in terms of objectives, input assets and output results. In addition, main roles involved to address not only technology security aspects but also inter-domain (roles interfacing) aspects were described. O-RAN, CN, virtualization,

SDN and End-to-End security assurance tasks and processes were defined. In particular, high-level security assurance processed were defined for:

1. O-RAN Evaluation - Security Assurance Processes
    - **Process 1-O-RAN: O-RAN Security Threat and Risk Assessment**
    - **Process 2-O-RAN: O-RAN Compliance and Certification Evaluation**
    - **Process 3-O-RAN: O-RAN Secure Deployment Process**
    - **Process 4-O-RAN: O-RAN Incident Response Planning and Execution**
2. Core Evaluation Security Assurance Processes
    - **Process 1-CN: CN Security Threat and Risk assessment**
    - **Process 2-CN: CN Compliance and Certification Evaluation**
    - **Process 3-CN: CN Secure Deployment and Operation**
    - **Process 4-CN: CN Incident Response Planning and Execution**

Along with an analysis on virtualization related aspects in Security Assurance Processes of O-RAN and CN building blocks, as well as risk and security compatibility assessment aspects in multi-vendor deployments.

3. SDN related Security Assurance Processes
    - **Process 1-SDN: SDN Security Threat and Risk Assessment (for SDN along with security evaluation)**
    - **Process 2-SDN: SDN Incident Response Planning and Execution**
4. End-to-End Network related Security Assurance Processes
    - **Process 1-E2E: End-to-End Threat Assessment**
    - **Process 2-E2E: End-to-End Compliance Evaluation**

Besides, the methodology emphasized on the need for tight collaboration among MNOs, equipment manufacturers, regulatory authorities, and evaluation teams, to ensure a holistic approach to security evaluation, respecting best practices, guidelines and standardization procedures.

Following this work, the activities performed in the context of Task 3.2 further focus on the analysis, testing and assessment of security aspects of the O-RAN and 5G Core network implementations (opensource), incorporating the aforementioned processes.

5. Open RAN security test scenarios

This section provides an overview of the main security test scenarios that are performed investigating vulnerabilities in the implementation of main RAN and CN functions. In the RAN segment, the main tests under evaluation include:

- Insufficient mechanisms protecting data and information of gNB components including physical layer attacks over the wireless air interface. The relevant tests are conducted using open source RAN implementations and custom jammers developed targeting Remote Units (RUs).
- Availability and integrity of gNB under varying loaded conditions as well as termination of the operation of gNB functions/components under unexpected data inputs.
- Incorrect implementation of security functions that may lead to failure in ensuring control plane data confidentiality protection over the N2 interface, improper mechanisms enforcing security policies, encryption mechanisms in the control plane of O-RAN (A1 and E2 protocols exposing interfaces to RT RIC and NRT-RIC controllers). T
- Vulnerable mechanisms for authentication and authorization of gNB, vulnerabilities in the N2 interface, unauthorized access to the RIC controller and improper management of authentication policies
- Monitoring mechanisms of gNB components and to what extend these can provide full and accurate knowledge of system status to the core network and the RIC controllers.
- Compute domain related vulnerabilities (virtualization, web services, hardware).

For the core segment, the activities focus on the identification of vulnerabilities related to security functionalities of the Core Network elements. The SECAM methodology will be extended and applied to different functions to explore vulnerabilities and then develop solutions for the: AMF targeting incorrect implementation of mechanisms such as authentication and key agreement procedures, as well as mechanisms for intra- Radio Access Technology (RAT) mobility, UPF assessing vulnerabilities in user data and signalling protocols, UDM targeting synchronization and authentication mechanisms. Security Edge Protection Proxy (SEPP) focusing on incorrect implementation of end-to-end core network interconnection security. Vulnerabilities for security functionalities that involve other Core Network elements Network Exposure Function (NEF) and Network Repository Function (NRF) will be also covered. Additionally, vulnerabilities in terms of Service Based Architecture (SBA) / Service Based Interface (SBI) 5G Core components will be examined and adequate protection of access and data in transit will be provided for the relevant SBI NFs ensuring the adoption of transport layer protection, authentication and authorization mechanisms.

The main test security scenarios that are also presented in D2.3 [12] are summarized below.

5.1. Multi-vendor Interoperability tests

Requirement Name: Interoperability tests across multi-vendor RAN and CN

Requirement Description: Network operators shall be able to deploy and operate mobile network infrastructures comprising multi-vendor RAN and CN elements

Purpose: To verify that an operational network can be established integrating multi-vendor Open 5G RAN platfoforms.

Procedure and execution steps

Preconditions:

- This test requires access to multiple RAN and CN solutions.
- The test requires a compute platform with sufficient compute resources hosting the virtualized/containerized network elements.

- RAN and CN should be interconnected with a transport network
- Availability of commercial and programmable O-RU .

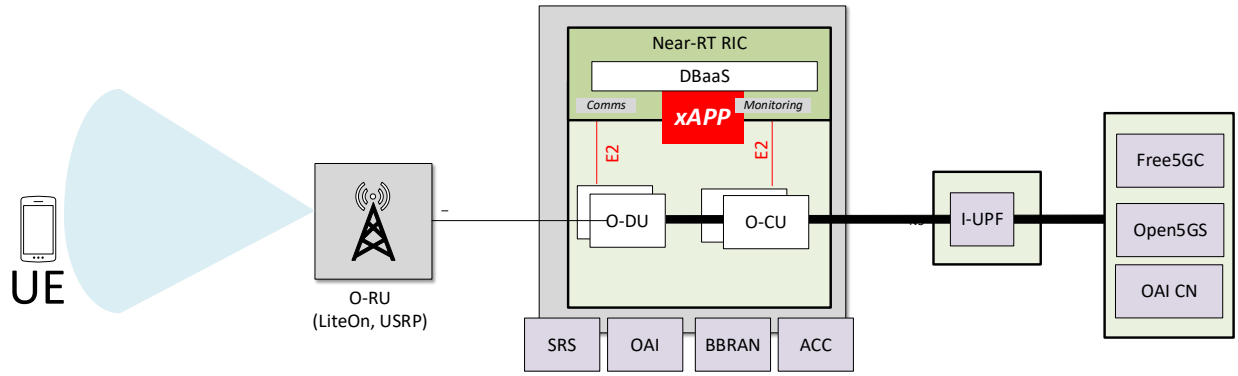

Figure 3: Multi-vendor RAN and CN deployment

Execution steps

1. The 5G platform is deployed and the UE is attached to the network.
2. A service slice between the UE and an external network is provided. The throughput for this connection is tested using speed measurements tools i.e., iperf.
3. The reconfigured and different combinations of RAN and CN are tested.
4. The UE throughput is continuously measured. Best and on the deployment option various performance metrics are calculated

Expected results

1. During the test, the UE is able to connect with the application server with the required throughput.
2. Service availability and performance of the UE should be similar under the same network configurations.

Expected format of evidence: Traffic captures and/or report files.

5.2. Wireless Physical Layer Attacks

Requirement Name: Service slice recover from physical layer attacks

Requirement Description: The 5G system shall be able to recover, without catastrophic failure, from a jamming attack instantiated by an external radio interface.

Purpose: To verify that a predefined jamming attack against the RAN segment will not crash the device under test, returning to the required service level **during the attack**.

Procedure and execution steps

Preconditions:

- This test requires access to a programmable radio interface (ie., Software Defined Radio (SDR)) that will be used to program and instantiate jamming attacks.
- The test requires an operational O-RAN compliant 5G system providing end-to-end services to the UEs
- UE and jammers should be physically separated (not collocated) as shown in *Figure 4*.

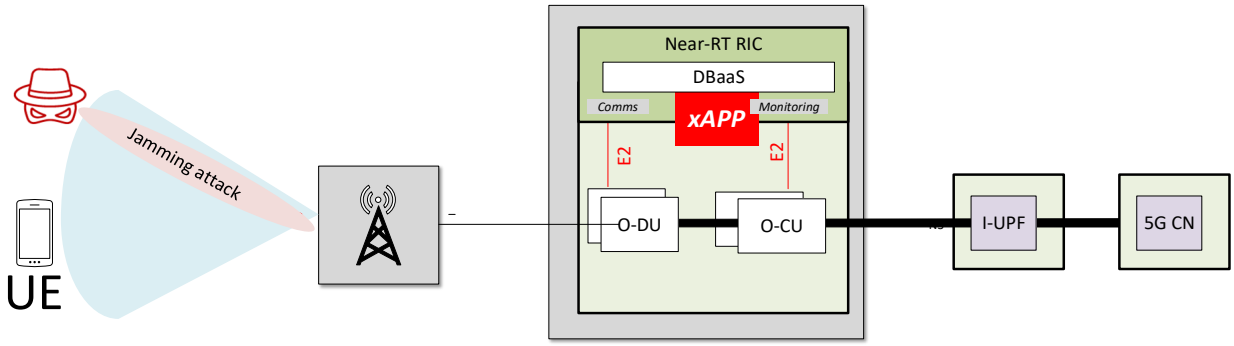

Figure 4: Physical Layer Attack Test Setup

Execution steps

1. The 5G platform is deployed and the UE is attached to the network.
2. A service slice between the UE and an external network is provided. The throughput for this connection is tested using speed measurements tools i.e., iperf.
3. The jammer is activated, and the attack is instantiated.
4. The UE throughput is continuously measured. A drop in throughput is expected
5. The xAPP providing protection against jamming attacks is activated.
6. The UE throughput returns in its normal state.

Expected results

1. During the test, the UE maintains an operational level.
2. After the activation of the xAPP, the degradation of service availability and performance of the UE is not noticeable.

Expected format of evidence: Traffic captures and/or report files.

5.3. Near-RT RIC Denial of Service (DoS) Attack

This scenario will implement the Near-RT RIC DoS Attack test case provided by the **O-RAN.WG11.TS.Security-Test-Specifications-R004-v09.00**

Requirement Name: Near-RT RIC E2 interface DoS recover

Requirement Description: "The Near-RT RIC shall be able to recover, without catastrophic failure, from a volumetric DDoS attack across the A1 interface, due to misbehaviour or malicious intent."

Purpose: to verify that a predefined volumetric DoS attack against Near-RT RIC A1 interface will not crash the SUT, returning to service level after the attack.

Procedure and execution steps

Preconditions

- The test requires access IP address information of the Near-RT RIC's E2 interface and a routable path to the target from the emulated attacker.
- The test requires the normal UE procedures and user-plane traffic can be handled properly through the SUT.

Refer to the diagram shown in *Figure 5* for the test setup and configuration.

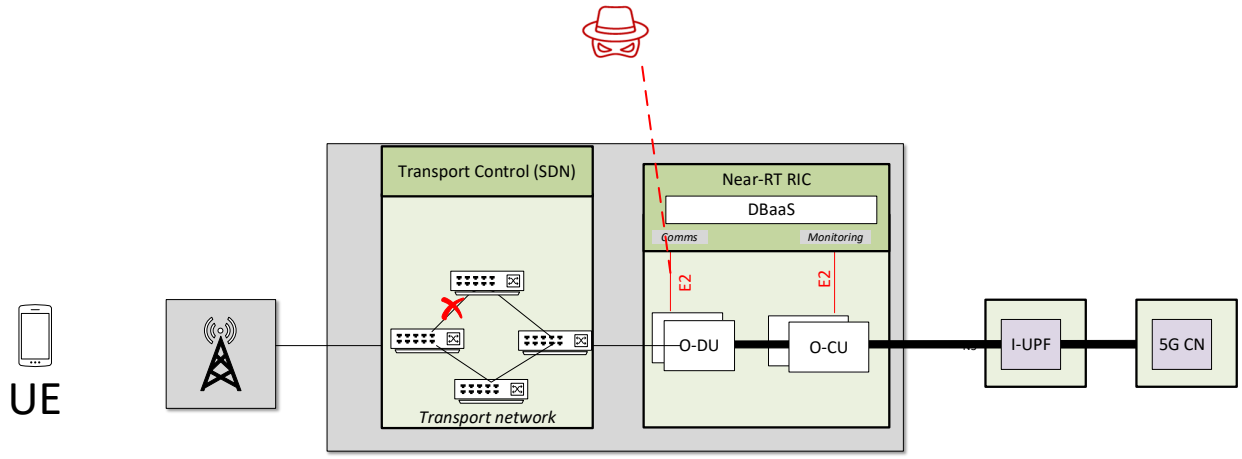

Figure 5: Near-RT RIC E2 Interface DoS Attack Test Setup

Execution steps

1. The tester uses a test tool to generate several types of volumetric DoS attack against the IP address of the Near-RT RIC E2 interface:
2. Volumetric tiers: 10Mbps, 100Mbps, 1Gbps
3. DoS Traffic random mixed of: generic UDP packets, HTTP/HTTPs REST API calls
4. DoS source address: spoofed IP of Non-RT RIC, random source IPs or broadcast IP (UDP only)

Expected results

1. During the test, the System Under Test (SUT) maintains an operational level.
2. After the execution of the test, the degradation of service availability and performance of the SUT is not noticeable.

Expected format of evidence: Traffic captures and/or report files

5.4. 5G End-to-End Cybersecurity Orchestrator

Ensuring that security measures are transversally integrated across all network domains, from RAN to the Core Network and edge computing components, is made possible by the End-to-End Security Orchestration (E2ESO) engine. Ensuring E2E security for services provided not only at the application layer but also at all lower layers is crucial for 5G networks.

Requirement Name: End-to-End Security Orchestration and cyberthreat information sharing

Requirement Description: The End-to-End security orchestrator performs the following actions:

- Performs security checks and validates 5G network elements before these are deployed in the field.
- Deploys the 5G system only if all security requirements are met.
- Collects monitoring statistics and alerts from local agents.
- Orchestrates network reconfiguration policies to minimize security risks
- Shares cybersecurity threat intelligence with other operators and authorities.

Purpose: To verify that the overall 5G system can proactively and reactively respond to cybersecurity threats.

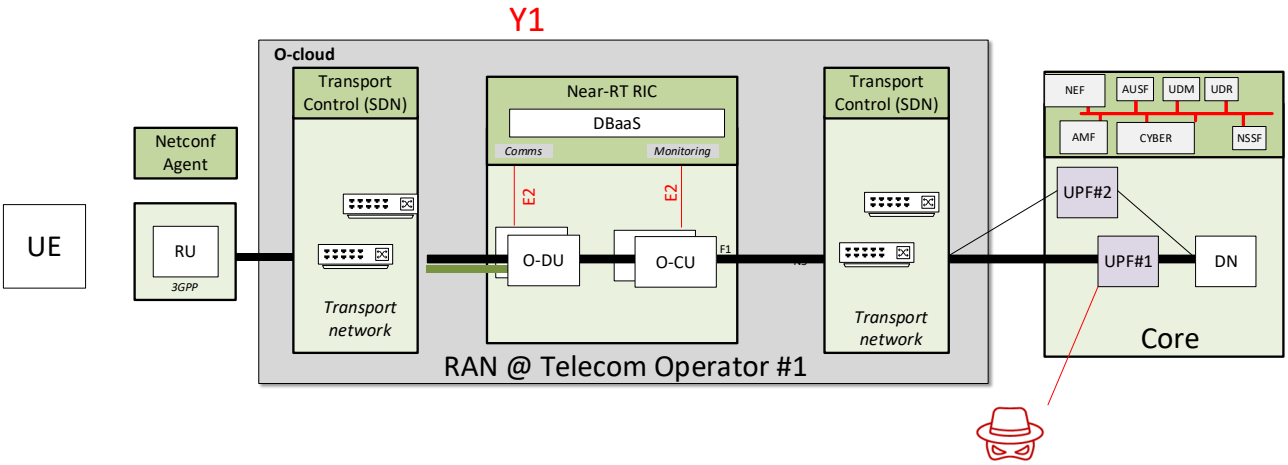

Figure 6: Security Orchestration.

Procedure and execution steps

Preconditions

- Deployment of the MANO managing the cloud infrastructure.
- The test requires the normal UE procedures and user-plane traffic can be handled properly through the SUT.
- Logging and alerts in the O-cloud are enabled.

Execution steps

1. The tester uses test tool (through O-Cloud MANO) to instantiate random attacks to a 5G network elements (i.e. UPF1 shown in *Figure 6*).
2. Measure performance degradation of the attacked element.
3. The orchestration triggers a migration of the 5G network elements changing the locations where these are hosted (moves to new UPF#2) adopting the moving target defense paradigm.
4. Run the benchmark test again to evaluate the performance of the 5G element in the new location.

Expected results

- After the execution of the test, the degradation of service availability and performance of the SUT is not noticeable.

Expected format of evidence: Logs, results, screenshots, report

5.5. Vulnerability Assessment of service flows handled by UPF

Requirement Name: Detection and mitigation of malicious service flows at UPF

Requirement Description: The UPF shall be able to detect and reject service flows generated from malicious UEs

Purpose: To verify that a service flows from malicious UEs are detected and dropped from the 5G network with minimal overhead for the network and without affecting the performance of the existing services.

Procedure and execution steps

Preconditions:

- This test requires the establishment of end-to-end service flows from multiple UEs that will be terminated at the application server.
- The test requires an operational O-RAN compliant 5G system providing end-to-end services to the UEs
- Legit and malicious UEs generate service flows traversing the UPF node as shown in Figure 7.

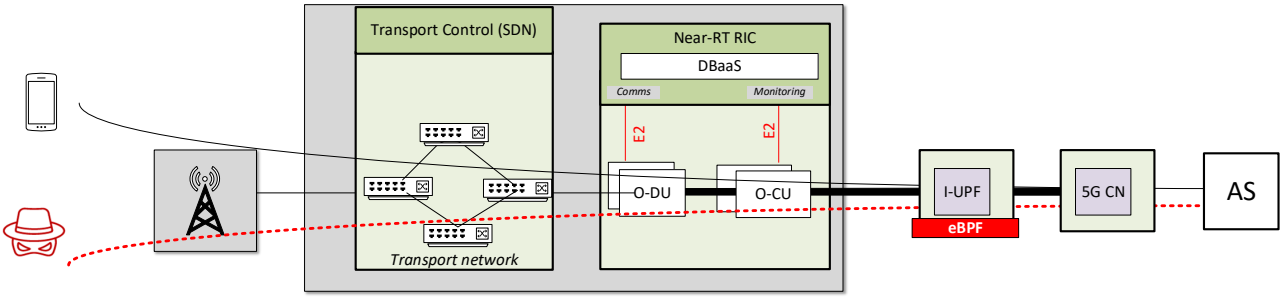

Figure 7: Malicious flows filtering

Execution steps

1. The 5G platform is deployed and the UE is attached to the network.
2. A connection between multiple UEs and an external application server is provided. The throughput for these connections is tested using speed measurements tools i.e., iperf.
3. A UE infected by malicious software attacks the network. The compute resources allocated to UPF increase and an increase in service delays is observed.
4. The deep packet inspection filter monitoring 5G service flows installed at UPF is activated.
5. The eBPF provides real time packet statistics
6. The eBPF detects and discards malicious packets at UPF
7. An alert is sent to the core network setting the malicious UE to an inactive state.

Expected results

1. During the test, all UE maintains an operational level.
2. After activation of the eBPF malicious packets are discarded.
3. The malicious UE cannot access the 5G network.

Expected format of evidence: Traffic captures and/or report files.

6. Open RAN evaluation

6.1. Open RAN interoperability testing

As shown in Figure 7, open RAN transforms 5G towards multi-vendor disaggregated software-based RAN, deployable in cloud-native environments, with open interfaces where 3rd party applications and AI agents could operate. O-RAN is an alliance producing specifications based on 3GPP specification and has introduced (a) new network entities on top of the 5G network (namely Service Management and Orchestration (SMO) and RIC), (b) open interfaces (O1, O2, E2, A1, R1), and (c) 3rd party applications (xApps and rApps). It is acknowledged that open and cloud-based RAN architectures increase the attack and vulnerability surface and extra measures are required.

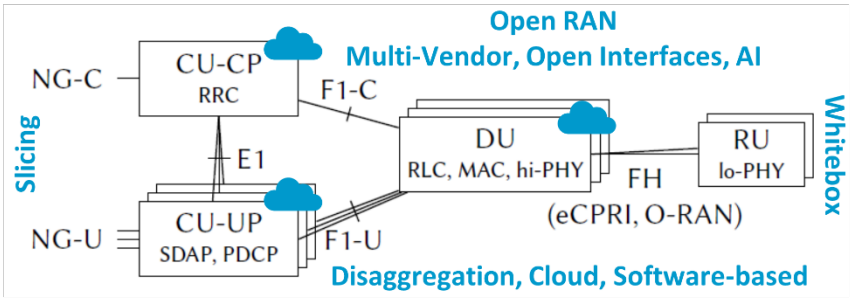

Figure 8: Open RAN model. Interfaces include O1: SMO to all, O2: SMO to O-Cloud, A1: Non-RT RIC to Neal-RT RIC, E1: O-CU UP and CP, E2: Neal-RT RIC to E2 Nodes, F1-C: O-CU CP to O-DU CP, F1-U: O-CU UP to O-DU UP, Open Fronthaul(7.2): O-DU to O-RU

At a software level, when deploying a 5G O-RAN network, different level of threats and vulnerabilities exist that may need specific measures.

- **Level 0:** Cloud infrastructure and Hardware components (e.g. O-Cloud)
- **Level 1:** Network functions in RAN and CN (e.g. GNB/CU/DU/UPF/AMF/SMF)
- **Level 2:** Management and control subsystems (e.g. SMO/RIC/xApp/rApps/NMS) as well as Artifacts and Artifact registry and Data Lake
- **Level 3:** Open Interfaces and APIs.

The transition from traditional, vendor-locked RAN systems to O-RAN promises to bring unprecedented flexibility and innovation. However, this transition also introduces a new set of security challenges, particularly those arising from O-RAN's extensive integration with cloud-native technologies. In contrast to traditional monolithic legacy architectures, O-RAN is founded on the principles of disaggregation, virtualization and software-defined infrastructure. This means that critical components are distributed across a variety of platforms and often hosted in multi-tenant public or hybrid cloud environments. Virtualized components such as the Central Unit (CU), Distributed Unit (DU) and the RIC can be deployed on commercial hardware in both centralized data centers and edge locations, with optimization according to latency needs. An example is also shown with the SMO framework, which is designed to run in the cloud and govern the lifecycle, configuration, and optimization of all O-RAN components. In conjunction with orchestration platforms such as Kubernetes, the SMO facilitates dynamic scaling, closed-loop automation, and streamlined multi-vendor management. While this architecture promotes agility and vendor neutrality, it also intensifies the need for robust cloud security and careful orchestration of trust boundaries.

When looking from a pure cloud-native perspective, four levels of security exist:

Level 0: code security, where code static analysis is required together with all the libraries.

Level 1: container Security, where image scanning and signing are required coupled with rootless privilege.

Level 2: cluster Security, where runtime security, policy enforcement, regular scanning, and best practices are required. Best practices include

- Labeling and segmentation of elements/pods/sections/namespaces
- Runtime security: Allowed/blocked/Audit rules, Mandatory Access Control (MAC), Real-time observability
- Continuous Policy lifecycle management: Audit ->Analyze-> Refine ->enforce
- Performance vs security trade-offs**.** For example, encryption / authentication for time-sensitive use-cases

Level 3: Cloud security, where securing the infrastructure, hardware vulnerability scanning and data encryption are required.

Each level requires specific strategies and tools to ensure comprehensive protection.

BubbleRAN's cloud-native E2E 5G O-RAN deployment model is depicted in Figure 9 , where different threads with different severity levels can happen.

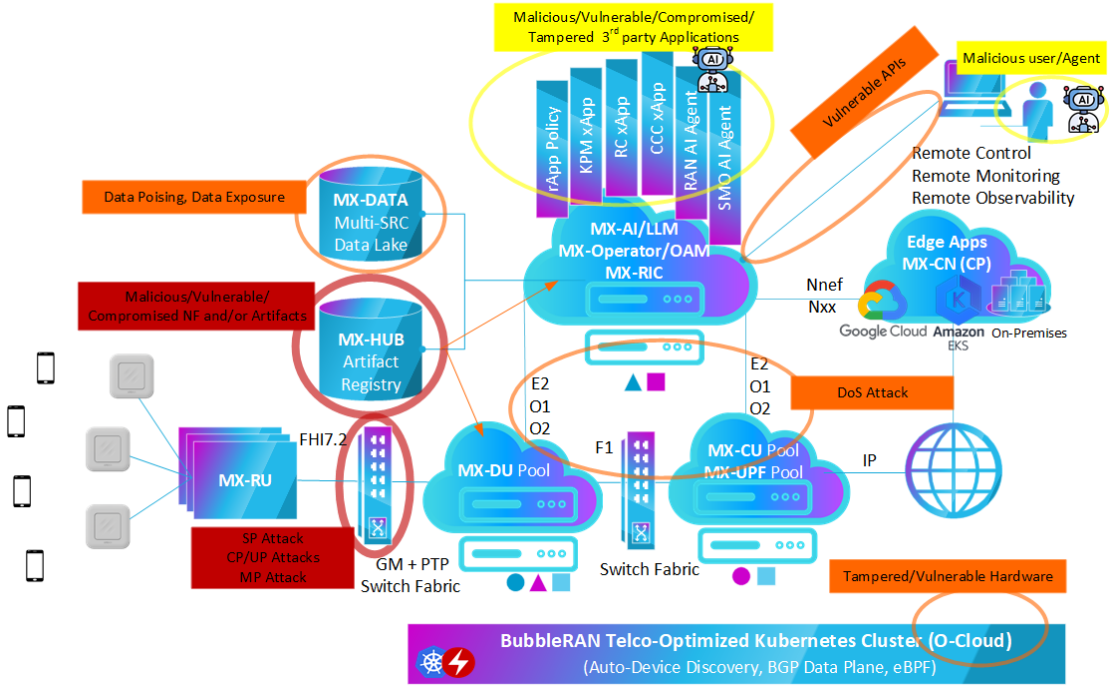

Figure 9 BubbleRAN cloud native O-RAN deployment model

Level 0 – Critical Attacks

Malicious/Vulnerable/Compromised Network Functions or Artifacts: This is related to all the NFs (CU/DU) and to O-RAN specific functions, i.e. SMO, RIC, and xApp/rApps, that could bring new forms of attacks in the overall system. For example, a malicious or compromised rApp could enforce a policy that violated SLA or xApp could perform a DoS attack by continuously subscribing to available E2SMs. In addition to the NF, container/virtualization environment could also bring its own additional level of threads, especially if certain privilege is granted. Yet another layer of attack could stem from the open-source codes where they may include malware or backdoors.

7.x Fronthaul Interface: This is another critical attack, and it is related to the interface between the base station (gNB/DU) and the cell site (RU). Attack on the synchronization plane could immediately break the network connectivity where the other planes (CP/UP/MP) could introduce sever network service degradation to the end users.

Level 1 – High Severity Attacks

- **Malicious/Vulnerable/Compromised/Tampered O-Cloud/Hardware:** This family of attacks could happen in two cases: Vulnerable/Compromised/Tampered specific hardware component (e.g. Sync-aware NiC) or Compromised/Tampered Kubernetes control plane nodes, each could cause partial disruption in the network connectivity.
- **Vulnerable Interfaces (F1, E2, O1, O2, A1, R1) and/or APIs:** These interfaces and APIs may allow unauthorized access to the network interfaces, resources, data, and/or network function inputs and outputs and could cause (a) partial network disruption or service degradation, and (b) user and data privacy and integrity issues.
- **Unauthorized Data Access (RD or RW):** This could cause data posing and exposure to an unauthorized 3rd party and could potentially cause (a) network performance degradation, and (b) data privacy and integrity issues.

Level 2: Medium Severity Attack

- **Malicious/Vulnerable/Compromised/Tampered 3rd party Applications:** Mainly due to inadequate or mis-configured access control mechanisms. This type of attack can cause service degradation in the network (e.g. SLA violation) as well as unauthorized data access and exposure.
- **Malicious user or AI Agent:** This is similar to the previous case with access to additional interfaces (monitoring) and tool (e.g. CLI and Dashboard) where more complex attacks could be realized. The surface of such attacks is large.

Table 1 Summary of Zero Trust solutions

Component/Domain	**Zero Trust Enabler**
Interfaces and APIs	mTLS, OAuth2, API gateway, IPsec/mTLS, RBAC enforcement, interface integrity policies
Workloads/Containers	KubeArmor, AppArmor, seccomp, signed containers, deny-by-default policies
Runtime Enforcement	AppArmor, BPF-LSM, eBPF-based detection, kernel monitoring
xApps / rApps	Code signing, behavioral monitoring, sandboxing, RBAC isolation
SMO and RIC (Near-RT/Non-RT)	AuthN/Z, API logging, signed app onboarding, continuous policy review, Role-based access, tenant isolation/segmentation,
Supply Chain	SBOMs, image verification, software provenance enforcement
Policy Lifecycle Management	Audit → Analyze → Refine → Enforce (automated + human-in-the-loop)

If all network functions and models deployed by the operator or service owner are trusted in BubbleRAN's architecture, the primary attack surface within the O-RAN architecture is its open interfaces and the integration of third-party applications. It is possible that these interfaces, which have been designed with the aim of facilitating interoperability between disaggregated components, may be susceptible to unauthorized access, misconfiguration, or malicious exploitation if not properly governed. At the same time, third-party

xApps and rApps enable innovation and advanced RAN intelligence but also present a possible vulnerability in terms of supply chain compromise and runtime abuse.

To mitigate these risks, as shown in Figure 10 KubeArmor is employed as a key security enforcement tool, providing runtime protection from the container level up to the Kubernetes cluster level. KubeArmor implements fine-grained access control, behavior monitoring, and policy enforcement directly within the system kernel, using eBPF for high-performance, context-aware security. This allows operators to define and enforce least-privilege policies for containers, ensuring that both trusted and untrusted workloads behave within expected bounds.

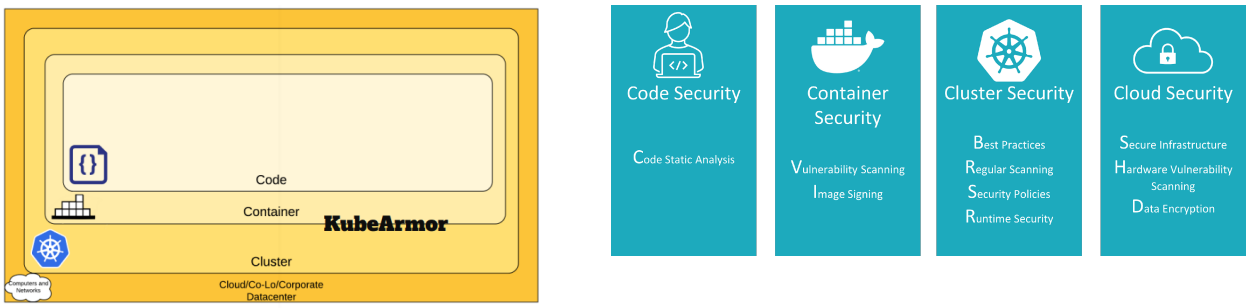

Figure 10 KubeArmor overarching design

6.1.1. BRAN benchmarking testing

If we take the assumption that the O-RAN stack is trusted as it will be provided by the operators, then the main vulnerability comes from the xApps and to a lesser extent from rApps. xApps having access to the network APIs (privileged access) could exercise a variety of actions on the underlying network ranging from performance degradation by means of conflicting control actions to network disruption via DoS attack. xApps are also able to infer and extract network information (e.g. topology) or interfere with other xApps if they are not sandboxed or access controlled. rApps, on the other hand, enforce high level policies (R1 in O-RAN architecture) that is later translated to a feasible network policy. Therefore, the main vulnerability comes from the xApps as they could break the zero-trust policy if they are not sandboxed and their access to the network API (privilege) is not properly set. Based on this analysis, xApps must either be only provided by the vendors or at provided by a trusted entity followed by the validation by the vendor, whereas rApps could be defined by the operators or 3rd party developers.

A security analysis of the O-RAN and BubbleRAN's architecture was conducted, which revealed that the biggest risks are posed by xApps, due to their deep integration with and control over RAN functionalities. xApps provide an intelligent control at the near-real-time layer, operate with privileged access to sensitive network parameters such as scheduling policies, mobility management, interference mitigation, and power control. While these capabilities empower advanced automation and optimization, they also significantly raise the stakes in terms of security. In the present study, it was determined that the most significant threat does not originate from conventional attacks, such as file system breaches or container escape. These concerns are predominantly mitigated by containerized deployment and robust cloud-based sandboxing. The primary threat is instead identified as the integrity and trustworthiness of the xApps themselves.

More specifically, the prevalence of third-party xApps is the primary source of security concerns. In contrast to operator-developed or certified applications, third-party xApps may not fully comply with operational standards or security best practices. The potential usage of xApps for malicious or negligent behavior may manifest in a variety of ways, including subtle network degradation (e.g., biased scheduling that favors certain users or cells). More severe actions include denial-of-service through excessive signaling, falsified RAN statistics, or unauthorized overrides of mobility decisions. These actions can have a considerable impact

on network performance and user experience, and in some cases, may even compromise network availability and regulatory compliance.

It is therefore strongly recommended that the deployment of third-party xApps in production RAN environments be restricted. It is paramount that only xApps that are cryptographically signed and vetted by the network operator are permitted, ensuring that any deployed software aligns with operational policies and that can be traced. The operator should be the only one validating access control mechanisms, enforcing resource usage limits, and continuously monitoring runtime behavior. These factors are critical for safely enabling programmable RAN functionality.

In contrast, rApps, which interact with non-real-time data and have a looser coupling to critical RAN control loops, present a reduced risk profile. A more permissive policy can be allowed for rApps, enabling for innovation and third-party participation, provided that adequate observability frameworks are in place.

6.2. Interoperability testing in operational mobile network environments

In the scope of the 5G-TACTIC project ORO has deployed a fully-featured O-RAN trial platform by combining ACC's virtualised baseband (dRAX CU, DU and near-RT RIC) with Benetel's RAN550 split-7.2x radio unit and an open-source 5G Core Network from Open5GS. The system architecture is showcased in Figure 11, while Figure 12 and Figure 13 showcase the dRAX interface for O-RAN system management.

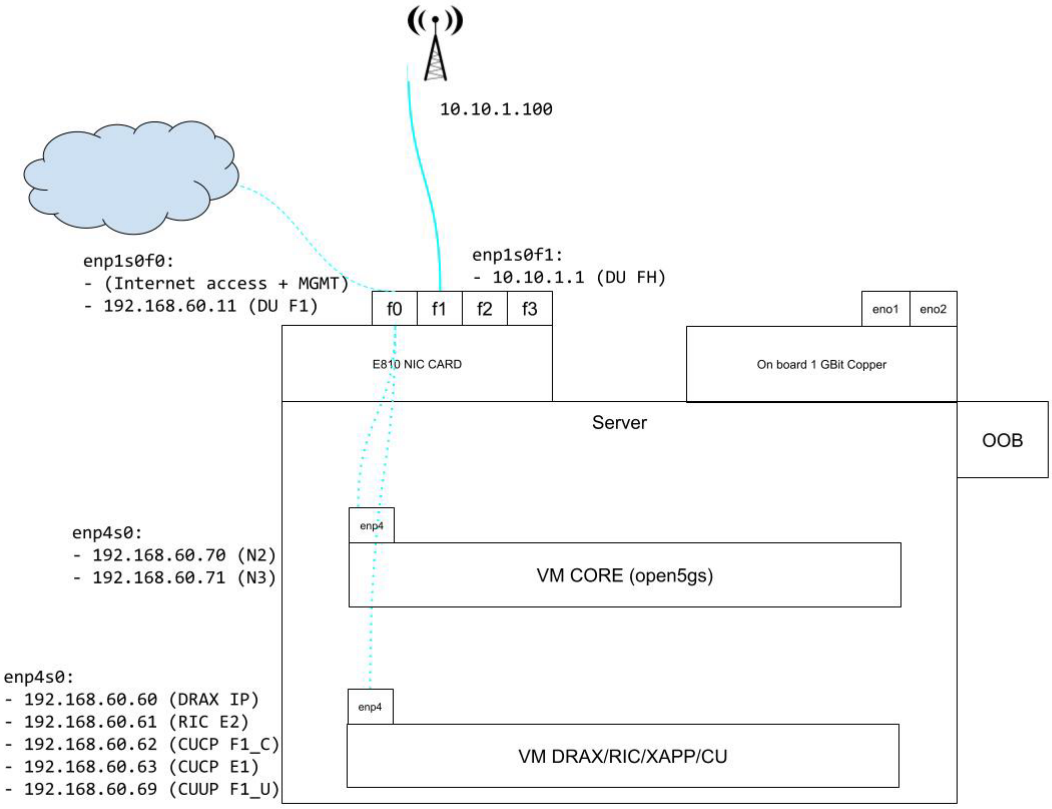

Figure 11 ORO-ACC O-RAN deployment architecture

ACC delivered a turn-key Supermicro 1U server that hosts two KVM virtual machines: one running the CU/DU/RIC stack and the other running the core network. The server's Intel E810 100 GbE NIC carries all logical interfaces: a dedicated 10GbE port (10.10.1.1) connects the RU over eCPRI/F1, another virtual port at 192.168.60.70/71 handles N2 and N3 traffic between the Core VM and the configured DN, while a third port at 192.168.60.11 provides internet break-out and management; a separate link offers out-of-band control. Moreover, the Intel NIC provides PTP synchronization for the TDD RU, using a GPS reference signal.

Prior to the integration, ORO upgraded the Benetel RU to firmware v1.3.11 to align with its 2024.2.3 system release. In the integration phase, ACC pre-loaded netplan files that pin each traffic type to the correct interface, and enabled a WireGuard tunnel so their engineers could remotely provision radio parameters.

Until this moment only small-scale performance tests were performed over the system using two COTS smartphones, 1x Samsung Galaxy S24 Ultra and 1x Motorola Edge 50 Pro—the RU was configured to support the n78 carrier and 100 MHz bandwidth. Initial speed-tests to the public internet reached roughly 500 Mbps downlink and 50 Mbps uplink, confirming the functional integrity of the end-to-end chain.

In regards to the security auditing tasks, running every network function on isolated VMs allows the research team to instrument hypervisor tap points for detailed control-plane tracing, while the integrated RIC opens the door to deploying custom xApps for anomaly detection and policy enforcement.

The next phase will target vulnerability scans of the CU/DU containers and the RU's eCPRI interface, benchmark the security of the split-7.2x front-haul interface, and broaden the UE matrix to stress scheduler behaviour under security-focused traffic loads—advancing the 5G-TACTIC project's goal of rigorously assessing O-RAN cyber-resilience.

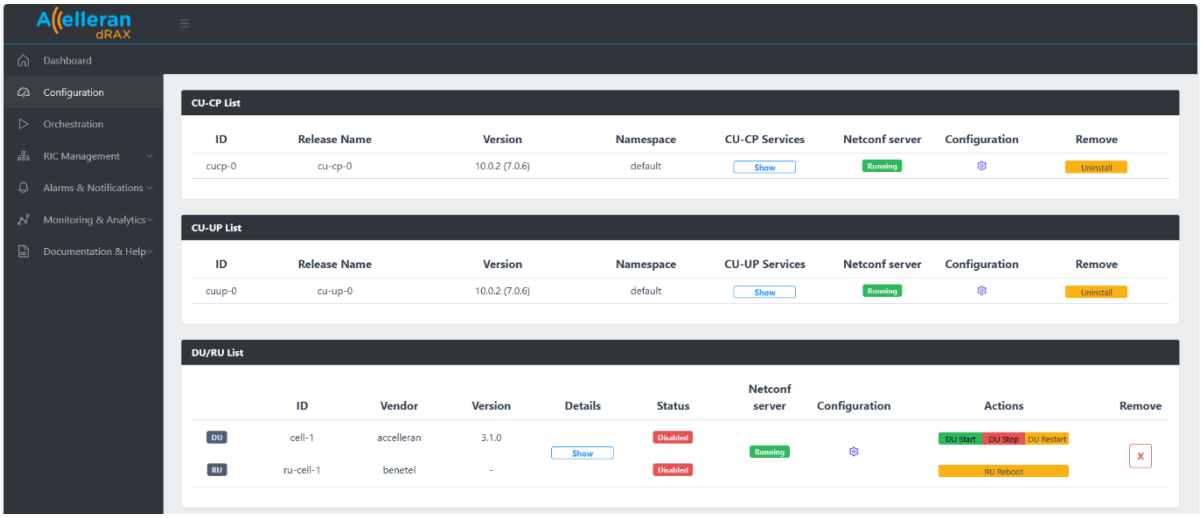

Figure 12 ACC dRAX interface – ORO system configuration

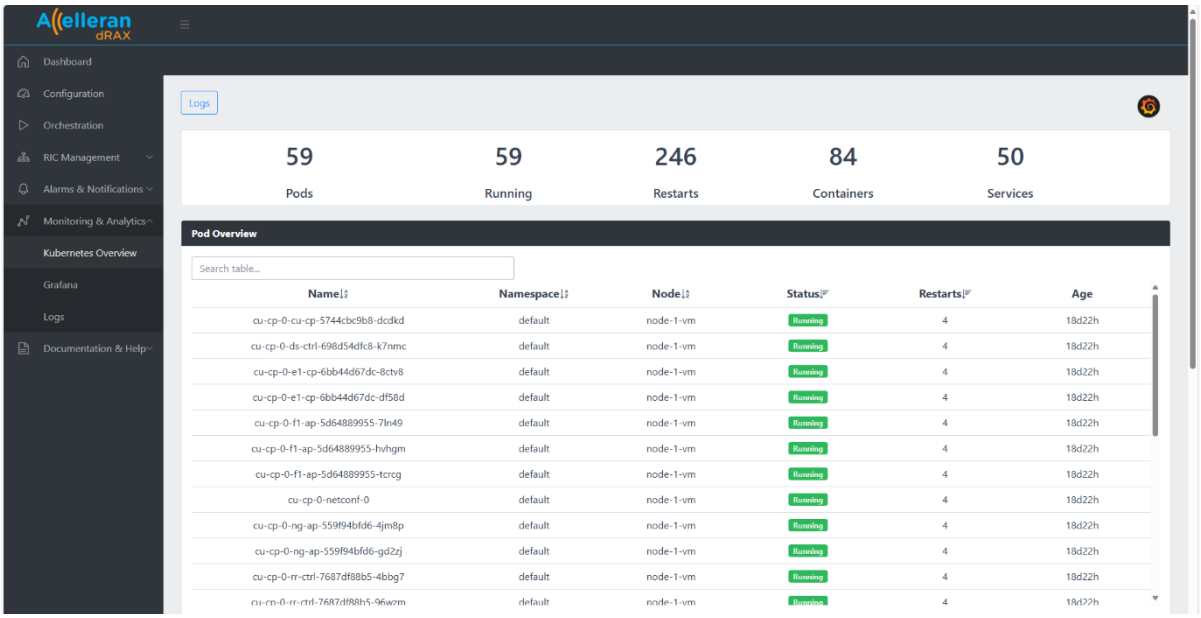

Figure 13 ACC dRAX interface – Kubernetes overview for dRAX/RIC/CU/DU stack

6.3. Addressing Wireless Physical Layer Attacks

6.3.1. Introduction

The transition to 5G and future network generations marks a significant shift in how mobile systems are built and operate. Higher frequencies, tighter cell densities, and advanced beamforming are no longer just theoretical, but they are actively shaping modern radio access networks. These technical advancements open up powerful new capabilities but also bring a fresh set of risks, particularly at the physical layer, where traditional encryption methods offer limited protection.

One of the most critical risks in this domain is Radio Frequency (RF) jamming. By injecting unwanted signals into the wireless channel, attackers can interfere with the connection between user devices and the base station. Synchronization and control signals, such as the Physical Broadcast Channel (PBCH) or Synchronization Signal Block (SSB), are common targets due to their key role in establishing communication. Some attackers use more advanced methods, like smart jamming, which scan the spectrum and focus on active channels. This makes attacks more efficient and harder to notice.

O-RAN adds another layer of complexity. Its modular structure and open interfaces support rapid innovation and interoperability, but they also increase the number of possible entry points. Components that were once part of closed systems are now programmable and interconnected, expanding the potential for exploitation. On the other hand, this architecture also makes it possible to build flexible defense systems that operate directly within the network itself.

In this context, the 5G-TACTIC project introduces a network-based approach protecting the physical layer in O-RAN environments. At the center of this effort is a custom xApp, designed to run on the Near RT-RIC. This application constantly monitors the radio environment and responds when signs of RF jamming appear. It uses spatial filtering and adaptive beam control to reduce interference, preserving service quality during attacks.

In addition to reacting to threats, the xApp continuously optimizes the performance of the system applying suitable RAN policies and coordinating with other parts of the network. This setup allows faster detection, more accurate targeting, and less impact on nearby users.

Instead of depending only on external security systems, 5G-TACTIC brings protection closer to where communication happens. It turns the RAN into an active part of the defense strategy, that is able to detect and handle threats as they occur.

The mitigation method used by the system is beam nulling. This technique reduces antenna gain in the direction of the jammer, while keeping the signal strong toward the UE. To do this effectively, the system constantly tracks where signals are coming from. It uses baseband IQ samples from multiple RUs to estimate the Direction of Arrival (DoA) in real time. Once the jammer's position is estimated, the system quickly recalculates beamforming weights. These updates are applied immediately, allowing the network to suppress interference without dropping active connections.

What sets the system apart is its ability to adapt on the fly. It doesn't just respond once, but it keeps monitoring the situation. It measures performance indicators like SINR and throughput continuously. Based on these metrics, it tests different beamforming configurations and updates the antenna weights as needed. The goal is to achieve the best throughput under jamming attacks. This real-time tuning helps the system maintain high throughput while minimizing service disruption.

In addition to local mitigation, the system also contributes to broader network defense. It sends jamming alerts to the Malware Information Sharing Platform (MISP), allowing other operators to stay informed and react quickly. This kind of threat sharing is especially valuable when dealing with coordinated or repeated attacks.

In the next section, we present the full development of the 5G-TACTIC xApp and its supporting architecture. The solution was tested in an O-RAN testbed that includes a programmable OFDM jammer, software-defined radios (SDRs) acting as both UE and gNB, and real-time monitoring tools. This setup enabled testing under realistic conditions and allowed us to validate the system's ability to detect, adapt, and protect against physical-layer attacks.

6.3.2. System Architecture and Design

The design of 5G-TACTIC combines real-time signal analysis, adaptive beamforming, and closed-loop control. It integrates directly into the near-real-time control layer of the RAN and operates through standardized O-RAN interfaces.

This section outlines the system's architecture, operational workflow, and core optimization logic.

Preconditions

At the core of the system is a custom developed xApp. It runs on the Near RT-RIC and is responsible for identifying jamming threats and responding in real time. The main components of the architecture are listed below:

- User Equipment (UE): A 5G NR device emulated using a USRP B210 software-defined radio, connected to a MIMO-capable 5G antenna array. It generates uplink traffic and serves as the endpoint for monitoring throughput and signal quality.
- Radio Unit (RU): Built using USRPs and connected to MIMO 5G antennas. The RU captures baseband IQ samples and applies beamforming weights sent by the xApp in real time.
- O-RAN CU/DU Stack: This stack handles both control and user data, following the O-RAN split architecture.
- Near RT-RIC and xApp: The Near-RT controller hosts the 5G-TACTIC xApp. This application processes RF data, estimates the direction of jamming signals, and computes the appropriate beamforming response. It coordinates mitigation actions across the RAN via standardized E2SM control messages.
- Jammer: A programmable OFDM jammer built using a USRP B210. It is capable of emitting various types of interference, including wideband (barrage) jamming to overwhelm the entire signal spectrum, making it hard for the receiver to decode any useful information, and control-channel-targeted jamming to block the UE from receiving synchronization or configuration data. This enables controlled and repeatable testing under realistic attack scenarios.
- Threat Intelligence Connector (MISP Module): A module integrated into our framework that exports detected jamming events to the Malware Information Sharing Platform (MISP). Each alert includes information such as the time and direction of the attack, helping operators share information and coordinate responses across networks.

Execution steps

The system operates through six coordinated stages, allowing it to detect, respond to, and report jamming events in real time:

1. **Signal Monitoring**: The RU continuously streams IQ samples from its antenna array to the xApp. These samples capture the current radio environment, including both normal signals and potential interference.
2. **DoA Estimation**: The xApp uses signal processing algorithm where strong interfering signals are coming from. These directions are flagged as possible jamming sources. Our framework is also flexible and can be extended with more advanced techniques for higher accuracy in direction estimation if needed.
3. **Beamforming Weight Calculation**: Based on the estimated interference direction, the xApp computes signal combining weights that reduce the impact of signals arriving from that direction. This spatial suppression helps mitigate jamming while preserving the desired communication link to the user equipment (UE).
4. **Weight Dispatch and Application**: The calculated weights are sent to the RU over the E2 interface, using the E2SM-RC service model. The RU then updates its antenna configuration in the next transmission cycle, applying the new beam pattern.
5. **Performance Feedback Monitoring**: The xApp continuously monitors key performance indicators from the UE, such as SINR and throughput. Throughput is read in real time from iperf logs every 1 sec. Rather than interfacing directly with the iperf3 tool via sockets or APIs, the xApp employs a lightweight file-based method that captures performance data from system-level output. Furthermore every 5 seconds, the xApp evaluates the current performance and applies a new beamforming configuration. This process repeats in a closed loop until the system converges on a setting that restores or maximizes throughput under the given interference conditions.
6. **Threat Intelligence Export**: When a jamming event is confirmed, it is logged and published to the Malware Information Sharing Platform (MISP)[53] . Each report can include information such as the affected frequency band, estimated direction, timestamp etc.

6.3.3. Closed-Loop Optimization Strategy

A key component of the system is its closed-loop optimization capability, where beamforming weights are dynamically adjusted based on real-world performance metrics rather than static configuration or purely model-based assumptions. As described above, the xApp continuously monitors end-to-end throughput between the UE and the gNB, using real-time network measurements such as those obtained from iperf3. If the observed throughput drops below a user-defined threshold, the system triggers an adaptation phase in which new beamforming weights are evaluated and applied iteratively.

This feedback-driven process includes the following steps:

1. Input: Live throughput measurements from the transport layer
2. Evaluation: Runtime comparison with predefined quality-of-service (QoS) thresholds
3. Adaptation: Selection or computation of new beamforming weights through a configurable optimization method (e.g. heuristic, or learning-based)

This approach enables the system to adapt to time-varying jamming conditions, imperfect direction estimation, and environmental changes, thereby enhancing its robustness in real-world deployments.

6.3.4. Timing and Real-Time Constraints

The system is designed to respond to performance degradation within practical timeframes:

Measurement interval: Throughput is sampled approximately every 1 second.

Weight application duration: Each candidate beamforming configuration is applied for a fixed duration (e.g., 5 seconds)

Optimization response time: Depends on the complexity of the optimization strategy employed (e.g., heuristic search, learning-based methods). In typical use cases, the system converges within a few iterations.

This operational structure allows for responsive mitigation of interference before significant service degradation occurs, while remaining flexible enough to accommodate different optimization algorithms in future extensions (e.g., reinforcement learning, model predictive control).

6.3.5. O-RAN-Based System Testbed

The 5G-TACTIC framework uses a closed-loop, near-real-time architecture shown in Figure 14. It connects the physical layer of the network with smart control logic running in the RAN.

The O-RAN Radio Unit (O-RU) collects IQ samples using its MIMO antenna array. These samples contain both normal traffic and possible interference. They are sent to the anti-jamming xApp, which runs inside the near-RT RIC.

The xApp analyzes the signals, estimates the jammer's direction, and calculates beamforming weights to block it. It does this by steering antenna nulls toward the interference source. The weights are sent back to the RU over the E2 interface and applied immediately.

At the same time, the xApp tracks performance metrics like UE throughput. These measurements help confirm if the mitigation works and guide further adjustments. If a jamming event is detected, it is reported to the MISP platform for threat sharing across networks.

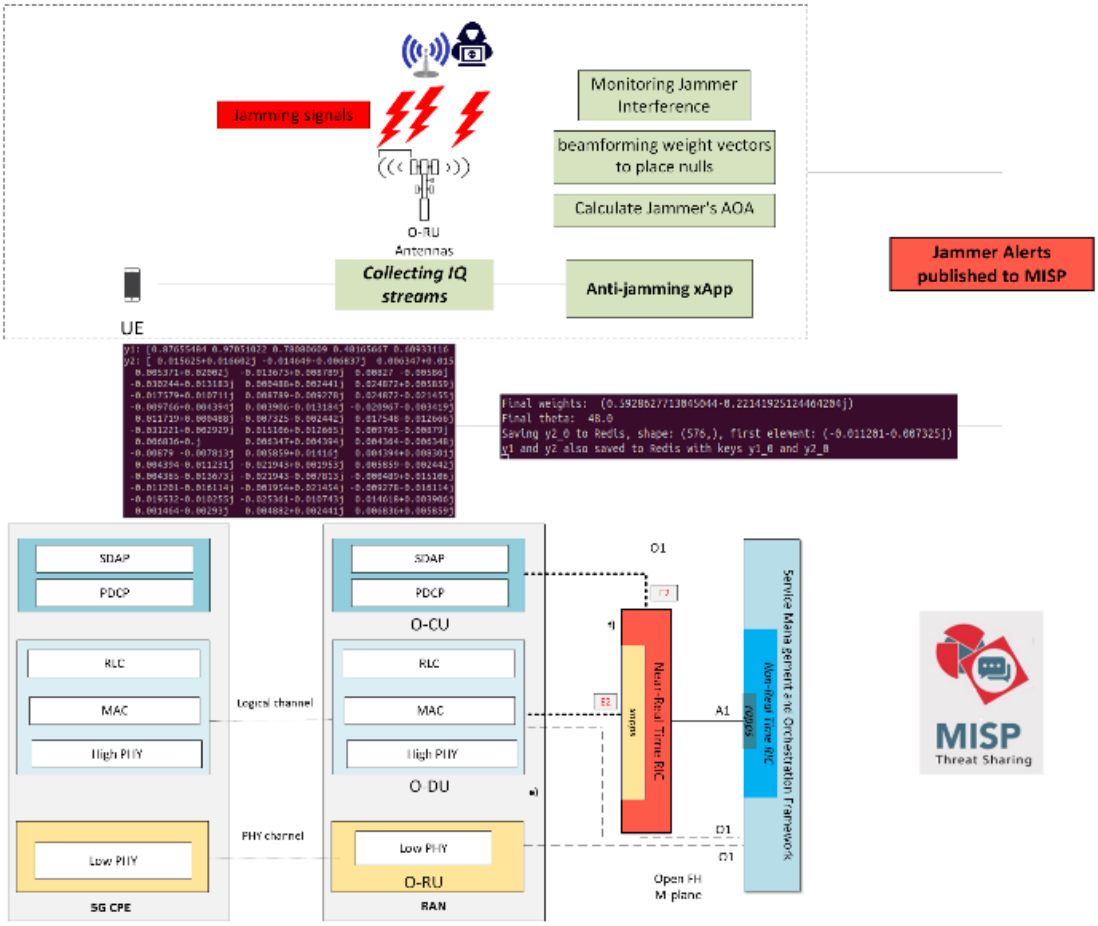

Figure 14 Overall architecture for radio attacks evaluation

The architecture follows O-RAN standards and uses E2SM-KPM and E2SM-RC models for control and telemetry. It also supports integration with other RIC applications running in parallel.
Figure 15 shows the physical layout of the testbed used to validate the 5G-TACTIC system. It consists of three main components: the UE, the RU, and the jammer.

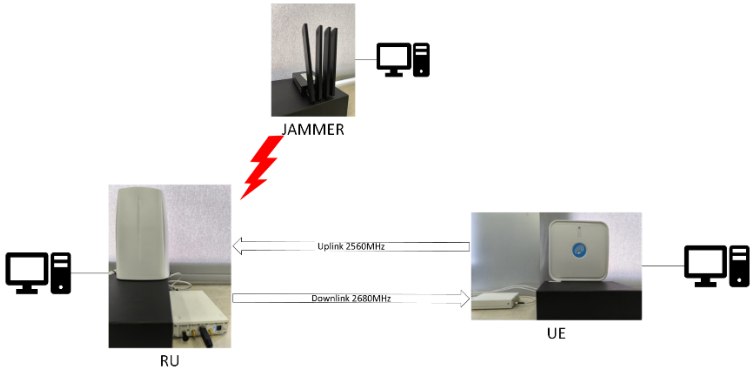

Figure 15 Physical setup for radio attacks evaluation

The UE communicates with the RU: The downlink operates at 2680 MHz, while the uplink uses 2560 MHz. Both the UE and RU are based on USRP B210 SDRs, each connected to a dedicated computer and equipped with MIMO antennas.
A programmable Orthogonal Frequency Division Multiplexing (OFDM) jammer (built using a USRP B210) is positioned between these, creating interference in the uplink band can emulate targeted jamming scenarios. All nodes are spatially separated to allow accurate direction estimation. This setup makes it possible to test the xApp's ability to detect the jammer, steer antenna nulls, and restore UE performance in real time.

6.3.6. Experimental Results

The goal of the evaluation is not only to confirm that interference is suppressed, but also to understand how the xApp adapts when it uses live performance feedback to fine-tune its beamforming decisions in a closed-loop setup.

Scenario 1: Reactive xApp Activation
In this scenario, the xApp calculates the beamforming weights in advance, using its full signal processing pipeline. It collects IQ samples from the RU, estimates the jammer's direction, and computes spatial nulls accordingly. These weights are stored and ready to be applied during the experiment.
The test begins with the UE connected to the RU under normal conditions. Communication is stable. Once the jammer is activated, the connection quality degrades significantly, confirming the presence of interference. At that point, the xApp is enabled. It immediately applies the computed weights and as a result, the throughput is restored and the system returns to normal operation.
Later in the test, the xApp is temporarily disabled. The interference effect reappears, and performance drops again. When the xApp is re-enabled, the same weights are applied, and the connection stabilizes once more.

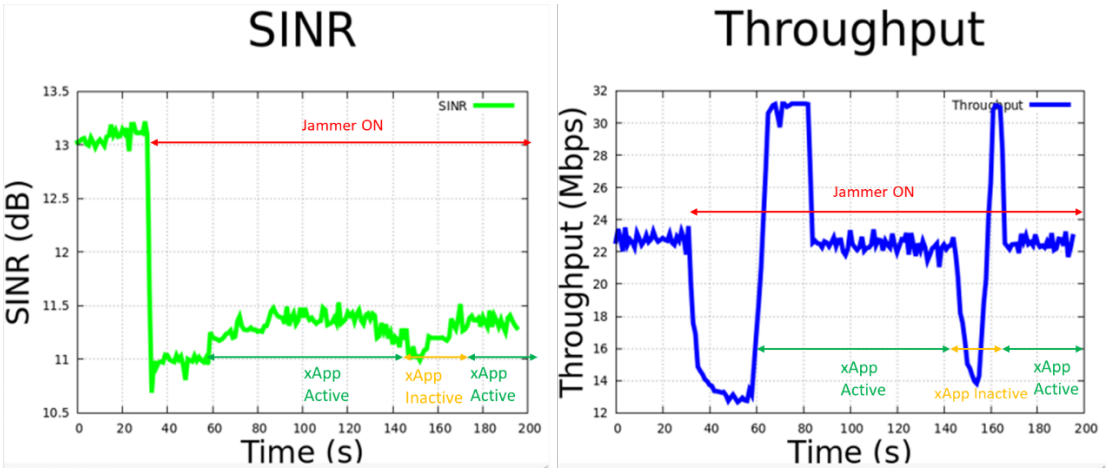

Figure 16: Jamming and xAPP developed to recover against physical layer attacks

Scenario 2: Adaptive Optimization with Real-Time Feedback

In the second scenario, the xApp runs in a fully autonomous mode. Instead of relying on precomputed configurations, it initially uses a heuristic-based model to randomly explore beamforming weights and observe their impact on throughput. The idea is simple: when a jamming event occurs, the system doesn't just react, but also it learns and adjusts until the problem is resolved.

As presented in Figure 16 once the UE is connected and stable, the jammer is activated. Throughput quickly drops, signaling that interference has impacted the link. At that point, the xApp is enabled. Initially, it enters an exploration phase where it applies random beamforming configurations using a heuristic model. This phase results in fluctuations in throughput, as seen in the early stage of the recovery window.

After several iterations, the xApp transitions to a reinforcement learning (RL)-based model. This model continuously reads throughput values and learns which beamforming weights yield better performance. Through repeated interaction with the environment, it gradually converges to a configuration that restores and improves throughput beyond the pre-jamming baseline.

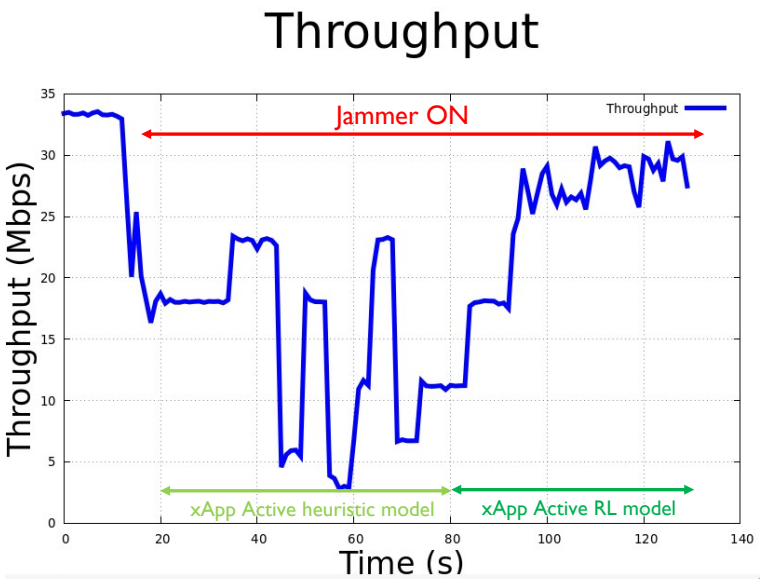

Figure 17: Recovery from physical layer attacks with xAPP

Figure 18 illustrates the real-time operation of the xApp during the RL-based optimization phase. The left terminal shows iperf3 throughput logs, while the right terminal shows the RL agent loop, where each throughput reading is used as a reward signal to guide beamforming weight selection. The agent iteratively chooses complex weight vectors and evaluates their performance advantage over a baseline.

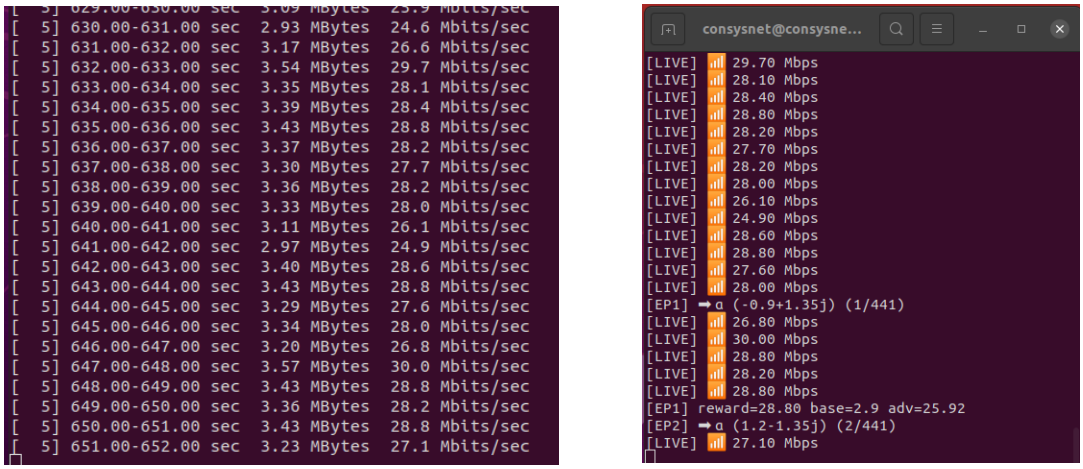

Figure 18: Throughout over time during jamming attacks

Each 5-second interval represents one "adaptation step." The goal is to reach a configuration where throughput returns to acceptable levels, close to the baseline measured before jamming began. In this scenario, the system recovers close to the original throughput, demonstrating the strength of learning-based optimization. The strength of this approach lies in its flexibility: it doesn't depend solely on angle estimation or static responses. Instead, it lets the network itself decide what works best, based on live feedback.

This method is especially useful when conditions change, like when the jammer's position drifts, multipath effects vary, or the DoA estimate isn't perfect. Also, it focuses on real user impact rather than theoretical assumptions. To evaluate the system's behavior under jamming conditions, a set of well-defined metrics is used:

- Recovery Time: Measures how long the system takes to restore stable throughput after jammer activation
- Adaptation Steps: Indicates the number of beamforming updates applied until recovery is achieved.
- Response Latency: Captures the delay between performance degradation and the first mitigation action.
- Beam Configuration Log: Tracks the sequence of beamforming weights tested during the optimization process.
- Threat Reporting Output: Structured event shared with MISP, including estimated DoA, timestamp, and affected frequency band.

When a jamming incident is confirmed, a structured threat report is generated and sent to the MISP platform. This includes metadata such as the estimated direction of arrival (DoA), time of detection, and affected frequency band. Together, these metrics provide a comprehensive and practical framework for assessing system responsiveness, efficiency, and situational awareness during interference scenarios.

The reinforcement learning component enhances system's adaptability to learn which beamforming patterns yield the highest throughput under jamming. In such a setup, the xApp acts as a learning agent. Its goal is to choose beamforming actions, such as which weight vector to apply, based on observations from the network. These observations might include recent throughput, signal strength patterns, or direction estimates. The RL agent interacts with the network in repeated cycles. At each step, it observes the current state, selects an action (e.g., apply a new beam configuration), and receives a reward based on the resulting performance. For instance, higher throughput or better SINR leads to a higher reward.

This learning-based control builds on an initial heuristic model, where the RL agent uses the measured throughput and weight that are applied as training data to guide its learning process. By starting from these observations, the agent can quickly identify effective beamforming configurations without relying on random exploration. Once trained, the RL agent generalizes to more complex environments. For example, when the jammer moves or when multiple interferers are present. It can also reduce the time needed to find a working configuration by avoiding repeated testing of poor-performing weights.

Integrating RL into the current framework shifts the system from simply reacting to interference toward learning how to handle it more intelligently. This can lead to faster recovery, fewer unnecessary beam switches, and more stable service for the user.

6.4. Spoofing attacks over E2 interface

The primary motivation of this work lies in the identification and experimental analysis of security vulnerabilities of the SCTP protocol in real implementations of O-RAN architectures, with particular emphasis on the E2 interface with is implemented over the SCTP transport protocol.

For this evaluation FlexRIC has been adopted as a near-RT RIC implementation. It is as an open-source implementation specifically designed to facilitate research, development, and validation of concepts within the O-RAN domain. It is implemented as a Software Development Kit (SDK) designed to build specialized controllers for software-defined radio access networks, and it comprises both server and agent libraries.

The study conducted in this section follows the SCTP protocol description presented in Section 7.2.2 . That section analyzes the detailed operation of the SCTP protocol, establishing the foundations for identifying its potential vulnerabilities, and presents the methodology followed for executing the attacks.

6.4.1. E2 Interface messages in SCTP association

This section deepens the analysis of the SCTP association establishment phase, as during this stage the E2 Setup procedure is executed. This procedure acquires special relevance in the present study, since through it the E2 node transmits its identification information to the Near RT-RIC.

The SCTP association process, which enables reliable and secure communication between two network nodes (such as between the nearRT-RIC and the E2 node in O-RAN), is initiated when the client sends an INIT chunk to the server to request connection establishment. The server responds with an INIT ACK chunk, which includes an encrypted cookie containing essential information to validate the association without prematurely compromising resources. Subsequently, the client returns the encrypted cookie to the server through the COOKIE ECHO chunk, thus demonstrating the authenticity of the request. Finally, the server confirms the association by sending a COOKIE ACK chunk.

Efficiently, the COOKIE ECHO chunk sent by the E2 node (i.e. O-DU, O-CU) also includes a DATA chunk that encapsulates the E2 Setup Request message of the E2AP protocol, as depicted in Figure 19. Although RFC 9260 prohibits sending data before receiving the COOKIE ACK, it allows a DATA chunk to accompany the COOKIE ECHO, provided that such data is not processed by the server until after correctly validating the cookie. This practice, common in SCTP implementations in the Linux kernel, reduces latency in the

establishment phase, thus optimizing E2 interface initialization. In this context, the E2 Setup Request message allows the gNB to communicate its identity (PLMN ID, Node ID) and the E2 functions it supports, enabling the RIC to build its control model over the node. Subsequently, once the association has been confirmed with the COOKIE ACK, the near-RT RIC responds with a DATA chunk containing the E2 Setup Response message, where acceptance (total or partial) of the proposed RAN functions is confirmed and the functional relationship between the controller and the node is formalized.

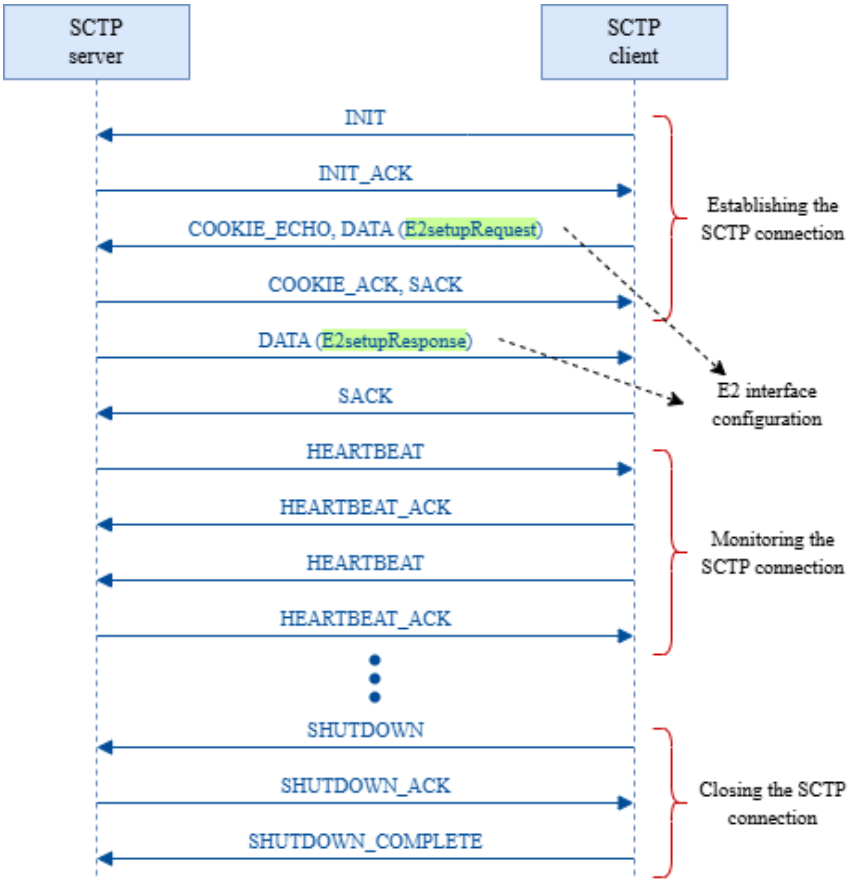

Figure 19 E2 SETUP messages over SCTP

6.4.2. O-RAN experimental platform

The testing platform is shown in Figure 20. It consists of two Virtual Machines (VMs) which play the role of O-RAN near-RT RIC and gNB entities (i.e. either DU or CU), and a third one which represents the malicious node. VM1 hosts the Near RT RIC component, which acts as the server node within the communication through the SCTP protocol. Although an xApp module is also deployed in this VM, this component was not actively used during the execution of the experiments presented in this work.

VM2 implements the SCTP client side through an E2 agent that simulates the functionalities of a gNB. Communication between VM1 and VM2 is established through their respective virtual network interfaces *enp0s3*, which enable the implementation of the E2 interface, responsible for message exchange between the near-RT RIC and the emulated gNB node.

Finally, it represents the attacking node within the testing environment. This node hosts the Python scripts developed to carry out different types of security attacks directed at the system. This component aims to simulate malicious behaviors in the system's control plane, in order to evaluate the robustness and possible vulnerabilities of the implemented O-RAN architecture against external attacks.

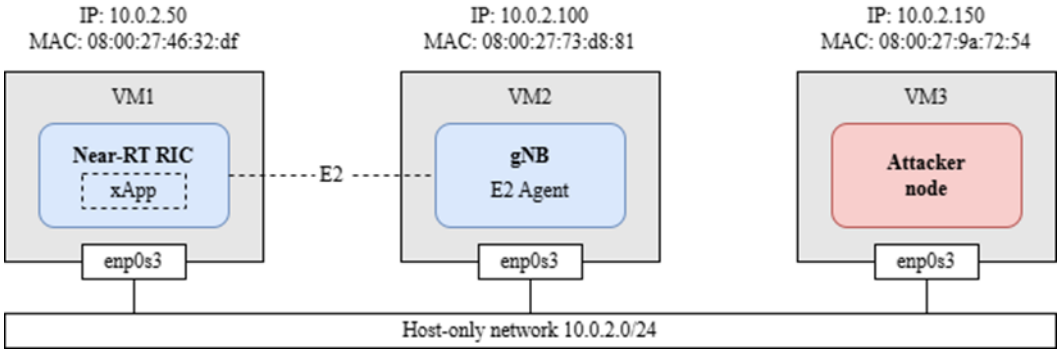

Figure 20 E2 interface testing platform

6.4.3. Spoofing attack

The closure of the SCTP connection between the Near RT-RIC and the gNB produced by the attack through the SHUTDOWN message, described in Section 7.2.2, does not significantly affect the operation of both components at the application level. The SHUTDOWN message induces the Near RT-RIC to remove the gNB identifier from its internal records, thus allowing the gNB to attempt to reestablish the connection subsequently. However, the ABORT message finally received by the gNB only has an effect at the SCTP layer level. At the application level, the gNB does not detect that the connection has been closed, suggesting a possible deficiency in the E2 node implementation. This limitation could be related to a lack of adequate notification to the upper layer, i.e., to the application (gNB), about the connection status. Such notification, which could manifest as an error or a specific event, depends on the implementation of the SCTP API used by the application, and its absence evidences a potential vulnerability in the system design.

The attack through the SHUTDOWN message has had three main effects: first, it has eliminated the SCTP connection between the Near RT-RIC and the gNB; second, the nearRT-RIC has deleted the gNB from its records and freed the E2 interface identifiers; and third, the gNB remains disconnected from the nearRT-RIC without realizing what has occurred.

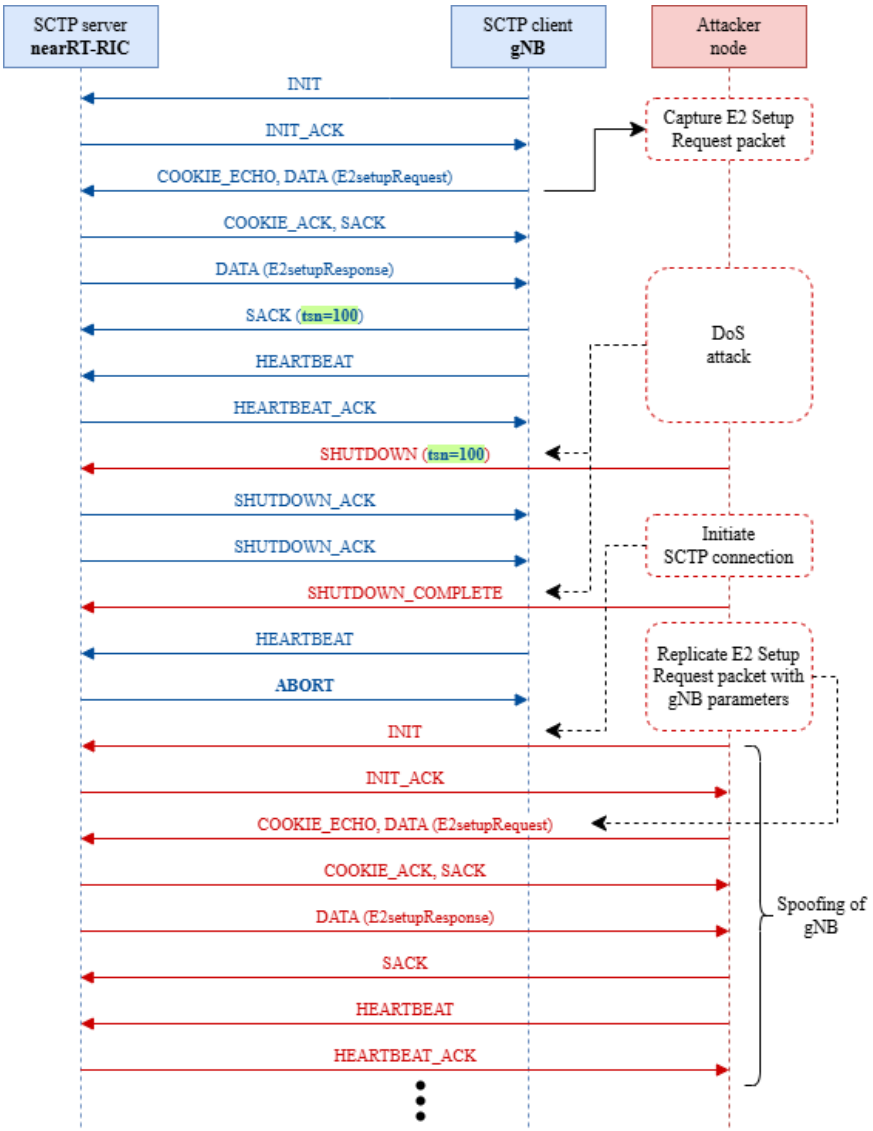

Figure 21 Spoofing attack over E2 interface

These circumstances can be exploited by the malicious node to execute a spoofing attack, as depicted in Figure 21. For this purpose, the malicious node performs continuous network monitoring from its initiation, with the objective of intercepting an SCTP packet containing the COOKIE_ECHO message. This packet incorporates a DATA chunk that includes an E2 Setup Request message from the E2AP protocol. The primary purpose of this message is to initiate the configuration of a connection between an E2 node, such as a gNB, and a Near RT-RIC, allowing the E2 node to notify its presence and capabilities to the controller, as well as establish an association intended for network management and control within the O-RAN architecture framework. Among the key parameters that identify the E2 node in this message, the following stand out:

- Global gNB ID: A globally unique identifier that defines the E2 node, composed of a public land mobile network identifier (PLMN ID) and a specific node identifier (gNB ID).
- E2 Node Component ID: A parameter that identifies individual components within the E2 node (for example, cells or radio units), allowing the nearRT-RIC to associate specific functions to each part of the node.

These parameters ensure that the nearRT-RIC can uniquely recognize and manage the E2 node, promoting coordinated and efficient communication in the O-RAN network. Figure 22 presents the E2 node (gNB) identifiers captured from an E2 Setup Request packet, obtained using the Wireshark tool.

Figure 22 E2 Setup frame dissection

Taking advantage of the complete disconnection between the Near RT-RIC and the gNB, the malicious node establishes a new SCTP connection with the Near RT-RIC by sending an INIT packet immediately after the DoS attack. Since this involves a new association, it is not essential to perform IP address spoofing. After receiving the INIT ACK, the malicious node sends a duplicate of the previously captured E2 Setup Request message, which contains the connection parameters associated with the gNB related to the E2 interface. In this way, the malicious node manages to spoof the gNB's identity for this new association. Subsequent packets, such as SACK and HEARTBEAT, are part of the usual SCTP connection management processes. However, both at the SCTP association level and at the E2 interface configuration level, the malicious node effectively operates as if it were the legitimate gNB.

Figure 23 presents a capture of the packets exchanged between the nearRT-RIC (IP 10.0.2.50), the gNB (IP 10.0.2.100), and the malicious node (IP 10.0.2.150), obtained using the Wireshark tool. Frames 3 to 15 correspond to the standard process of establishing the SCTP association between the nearRT-RIC and the gNB, as well as to the configuration of the E2 interface between both entities.

	Time	Source	Destination	Protocol	Length	Info	Src MAC	Dst MAC
3	1.800268	10.0.2.100	10.0.2.50	SCTP	82	INIT	08:00:27:73:d8:81	08:00:27:46:32:df
6	1.800692	10.0.2.50	10.0.2.100	SCTP	306	INIT_ACK	08:00:27:46:32:df	08:00:27:73:d8:81
7	1.800956	10.0.2.100	10.0.2.50	E2AP|NGAP	1314	COOKIE_ECHO , E2setupRequest	08:00:27:73:d8:81	08:00:27:46:32:df
8	1.801015	10.0.2.50	10.0.2.100	SCTP	66	COOKIE_ACK SACK (Ack=0, Arwnd…	08:00:27:46:32:df	08:00:27:73:d8:81
9	1.801386	10.0.2.50	10.0.2.100	E2AP	190	E2setupResponse	08:00:27:46:32:df	08:00:27:73:d8:81
10	1.801582	10.0.2.100	10.0.2.50	SCTP	62	SACK (Ack=0, Arwnd=106368)	08:00:27:73:d8:81	08:00:27:46:32:df
14	35.515988	10.0.2.100	10.0.2.50	SCTP	106	HEARTBEAT	08:00:27:73:d8:81	08:00:27:46:32:df
15	35.516044	10.0.2.50	10.0.2.100	SCTP	106	HEARTBEAT_ACK	08:00:27:46:32:df	08:00:27:73:d8:81
18	35.539807	10.0.2.100	10.0.2.50	SCTP	60	SHUTDOWN	08:00:27:9a:72:54	08:00:27:46:32:df
19	35.539832	10.0.2.50	10.0.2.100	SCTP	50	SHUTDOWN_ACK	08:00:27:46:32:df	08:00:27:73:d8:81
20	36.566025	10.0.2.50	10.0.2.100	SCTP	50	SHUTDOWN_ACK	08:00:27:46:32:df	08:00:27:73:d8:81
21	36.579055	10.0.2.100	10.0.2.50	SCTP	60	SHUTDOWN_COMPLETE	08:00:27:9a:72:54	08:00:27:46:32:df
28	68.283807	10.0.2.100	10.0.2.50	SCTP	106	HEARTBEAT	08:00:27:73:d8:81	08:00:27:46:32:df
29	68.283867	10.0.2.50	10.0.2.100	SCTP	50	ABORT	08:00:27:46:32:df	08:00:27:73:d8:81
35	139.726252	10.0.2.150	10.0.2.50	SCTP	82	INIT	08:00:27:9a:72:54	08:00:27:46:32:df
36	139.726350	10.0.2.50	10.0.2.150	SCTP	306	INIT_ACK	08:00:27:46:32:df	08:00:27:9a:72:54
37	139.726683	10.0.2.150	10.0.2.50	E2AP|NGAP	1314	COOKIE_ECHO , E2setupRequest	08:00:27:9a:72:54	08:00:27:46:32:df
38	139.726736	10.0.2.50	10.0.2.150	SCTP	66	COOKIE_ACK SACK (Ack=0, Arwnd…	08:00:27:46:32:df	08:00:27:9a:72:54
39	139.727251	10.0.2.50	10.0.2.150	E2AP	190	E2setupResponse	08:00:27:46:32:df	08:00:27:9a:72:54
40	139.727513	10.0.2.150	10.0.2.50	SCTP	62	SACK (Ack=0, Arwnd=106368)	08:00:27:9a:72:54	08:00:27:46:32:df
45	173.624143	10.0.2.150	10.0.2.50	SCTP	106	HEARTBEAT	08:00:27:9a:72:54	08:00:27:46:32:df
46	173.624195	10.0.2.50	10.0.2.150	SCTP	106	HEARTBEAT_ACK	08:00:27:46:32:df	08:00:27:9a:72:54
47	174.230103	10.0.2.50	10.0.2.150	SCTP	106	HEARTBEAT	08:00:27:46:32:df	08:00:27:9a:72:54
48	174.230995	10.0.2.150	10.0.2.50	SCTP	106	HEARTBEAT_ACK	08:00:27:9a:72:54	08:00:27:46:32:df

Figure 23 Wireshark capture of the spoofing attack over E2 interface

In frame 18, the malicious node initiates the attack by sending a SHUTDOWN packet to the Near RT-RIC, according to the description provided in previous sections. Although the malicious node possesses IP address 10.0.2.150, in this SCTP packet it spoofs the gNB's identity, using the value 10.0.2.100 as the source IP address. This spoofing can be confirmed by analyzing the associated MAC addresses. In previous frames, the gNB's MAC address is identified as 08:00:27:73:d8:81; however, in frame 18, the MAC address of the source node, which presents itself as the gNB, is 08:00:27:9a:72:54, corresponding to the malicious node.

In frames 19 and 20, the Near RT-RIC sends a SHUTDOWN ACK packet to the gNB (IP 10.0.2.100, MAC 08:00:27:73:d8:81). As previously explained, the gNB discards this message because it does not expect a SHUTDOWN ACK according to its internal state. Upon detecting the presence of the SHUTDOWN ACK packets, the malicious node responds by sending a SHUTDOWN COMPLETE packet, again spoofing the gNB's IP address (10.0.2.100), but using its own MAC address (08:00:27:9a:72:54), which evidences the persistence of the spoofing.

In frame 29, the Near RT-RIC completely terminates the association with the gNB by sending an ABORT packet. In subsequent frames, the malicious node initiates a new SCTP association with the Near RT-RIC using its own identity, without the need to spoof the gNB at the network level, since the spoofing has been consummated at the E2 interface level. Therefore, in later frames the real IP and MAC addresses of the malicious node are observed (IP 10.0.2.150, MAC 08:00:27:9a:72:54), confirming its active role in the new connection established with the Near RT-RIC.

7. 5GC security evaluation

This section describes the security evaluation performed over the 5G core according to the security architecture defined in [1] . The section will first describe the 5G Core Network implementations evaluated and the deployment options. Then, the security evaluation tools and techniques will be introduced, before outlining the security evaluation outcome. The security evaluation conducted in this section mainly focuses on those interfaces of the 5G core that interconnect with the RAN, and whose designs and protocol stacks are specific to the 5G Core. Furthermore, an analysis over the internal SBI will be also performed. Figure 24 shows the 5G core functional architecture indicating the interfaces involved in the evaluation; namely N2, N3, N4 and SBI.

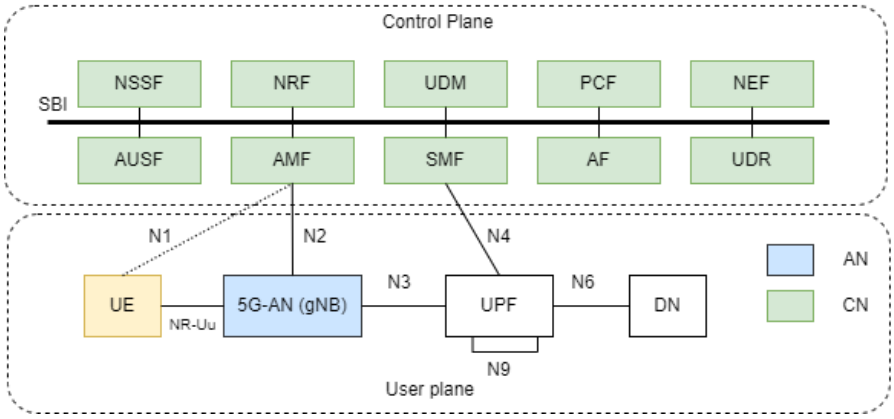

Figure 24. 5G core functional architecture and interfaces with the RAN

The N2 interface [15] connects the control plane of the AN and CN. Besides, it embeds the N1 interface, that allows the interaction between the UE and the CN. Specifically, the N2 interface appears between the gNB and the AMF entity of the CN. Figure 25 depicts the protocol stack of the N2 interface. As can be seen, it is an IP-based interface that uses SCTP [14] as transport protocol. In the following the main procedures brought by the N2 interface are defined to better understand the impact of attacks over it:

- NG setup: it is used to exchange application level data needed for the RAN and the AMF to correctly interoperate. This procedure erases any existing application level configuration data in the two nodes and replaces it with the one received
- Initial UE message: it is used when the AN node has received the first uplink NAS message from a UE, that is then forwarded to the AMF.
- Down/up-link NAS transport: it is used by the AMF to send NAS message transparently to the UE through the RAN (down) or when an associated UE communicates with the AMF (uplink).
- Initial context setup: it is sent from the AMF to request the setup of an associated UE, through the RAN.
- PDU session setup: this procedure is used to assign resources on the Uu interface (between the RAN and the UE) and the N3 (between the RAN and the UPF). It applies to one or several PDU sessions, and their corresponding QoS flows, and it sets up the corresponding Data Radio Bearers (DRB) for a given UE.

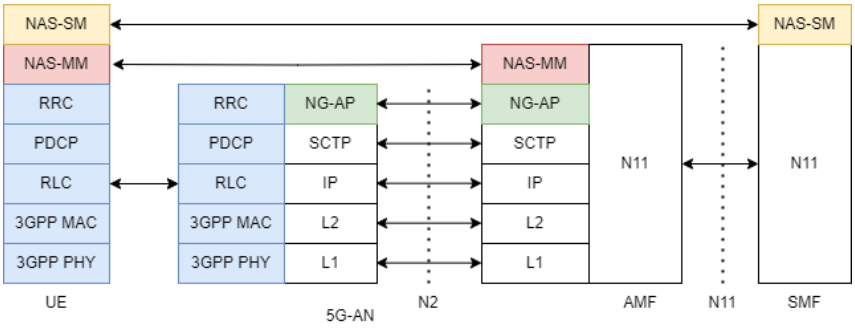

Figure 25 Protocol stack of the N2 interface

The N4 interface [15] allows the interaction between the SMF and the UPF, so connecting user and data planes within the 5G core. While it is an internal 5G core interface, it does not follow the service-based design existing between the rest of the entities in the core. As can be observed in Figure 26, the interface makes use of the Packet Forwarding Control Protocol (PFCP) that is directly transported over UPD messages. This interface provides the following procedures:

- Association Setup: it is used to create a PFCP association so that the SMF can use the resources of the UPF. The association can be initiated by any pair of the communication.

- Session establishment: it is sent by the SMF to establish a new session context at the UPF, so that it can handle incoming packets from UEs.
- Session modification: this procedure allows the dynamic reconfiguration of a session; for instance, by creating a new rule to handle user traffic, modifying an existing one, or deleting a rule of a session.
- Session report: it is sent by the UPF to report information related to a PFCP session to the SMF.
- Session deletion: it terminates a PFCP session and it is sent by the SMF, for instance when the session corresponding to a UE (e.g. PDU session) is deleted.

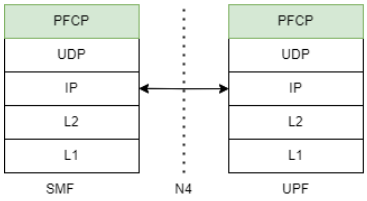

Figure 26 Protocol stack of the N4 interface

The protocol stack of the N3 interface [16] is shown in Figure 27. It allows the interaction between the AN and 5G core at the user plane, being responsible of transporting user data between the gNB and the UPF at the core. As can be observed, the interface adopts the GPRS Tunneling Protocol (GTP) which is transported over connection-less UDP messages. Differently to the previous cases, N3 interface uses GTP-U as a tunneling protocol that carries both user and signaling data. This protocol stack is also used to implement the N9 interface that allow the communication between different UPF instances.

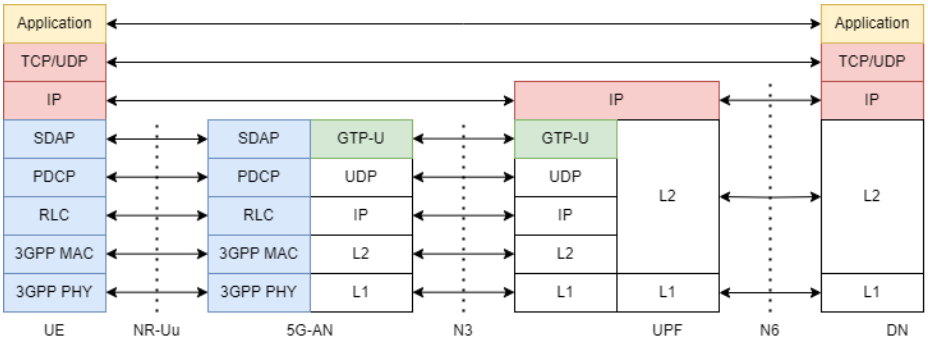

Figure 27 Protocol stack of the N3 interface

7.1. 5G core implementations

Over the past few years Open-source solutions for the 5G Core have gained popularity as low-cost alternatives to proprietary implementations, allowing flexible deployments on standard commercial hardware [17] . Among the most prominent platforms are Free5GC, Open5GS, and OpenAirInterface (OAI), each with distinctive features in terms of architecture, performance, and deployment options [18] [19] . These solutions also offer different implementation modalities: virtual machines, Docker containers, and Kubernetes orchestration [20] . Table 2 presents the main technical features of the implementations, which are summarized below:

- **Free5GC:** it is an academically developed 5G Core implementation that follows 3GPP Release 15 specifications and is written in Go [17] [19] . This platform stands out for its modularity and flexibility, offering a web interface for management and support for non-3GPP devices.
- **Open5GS:** this is a mature platform developed in C++ that implements 3GPP Release 17 specifications, standing out for its stability and efficiency in control plane procedures. It provides the best control plane latency among the evaluated solutions.

- **OpenAirInterface (OAI):** it is a comprehensive platform that covers both the 5G Core and radio access network functions, implementing 3GPP Release 16 specifications. It stands out for offering the highest data rates in the user plane [17] [19] .

Table 2 Summary of features of the 5G core implementations analyzed

Feature	Free5GC	Open5GS	OpenAirInterface (OAI)
Specifications	3GPP Release 15	3GPP Release 17 **Error! Reference source not found.**	3GPP Release 16
Language	Go	C++	C
Key Features	• Modular architecture • Lower resource consumption • Full 5G SA control plane support • Integrated web interface [5]	• Best control plane latency [1] • VoNR support • Network monitoring • DPDK optimization	• Highest user plane data rates [1][3] • MEC support • RAN integration • NEF support
Performance Focus	Resource efficiency	Control plane stability	User plane throughput
Feature	Free5GC	Open5GS	OpenAirInterface (OAI)
Specifications	3GPP Release 15	3GPP Release 17	3GPP Release 16
Language	Go	C++	C

Independently of the solution, all of the 5G Core options follow the SBA approach, as defined by the 3GPP standards. The architecture is organized into distinct functional domains, separating the control and user planes, and highlighting the modular and flexible nature of modern 5G networks. According to the architecture depicted in Figure 24, in the following the different entities and functions implemented re described:

1. **User Equipment (UE) and Access Network (RAN):** While these two elements are included in the generic definition of the architecture, they do not actually belong to the 5G Core, but rather are the elements that make use of the communication interfaces provided by the 5G Core.
   - UE (User Equipment): Represents the mobile device or terminal that connects to the 5G network.
     - Free5GC does not include a built-in UE simulator, but it is commonly used with UERANSIM, an open-source project that simulates both UE and gNB (RAN).
     - Open5GS, like Free5GC, does not include its own UE implementation. It is typically tested with UERANSIM or real devices.
     - OAI provides its own UE stack (OAI-UE), which can run on standard computers or SDR hardware, allowing for both simulated and real device testing.
   - (R)AN (Radio Access Network): Provides wireless connectivity between the UE and the core network. The interface between UE and (R)AN is labeled as N1, and between (R)AN and UPF as N3.
     - Free5GC does not provide a RAN implementation. It relies on external simulators like UERANSIM or real 5G gNB hardware for integration and testing.
     - Open5GS does not include a RAN implementation. It is compatible with UERANSIM (for simulation) or real gNB hardware.

- OAI stands out by providing a full 5G RAN implementation (OAI-gNB), which can be used with SDR (Software Defined Radio) hardware for real

2. User Plane Functions: they refer to the set of network entities responsible for the actual transport, routing, and delivery of user data packets between the mobile device (UE) and external services or networks. The two main components in this set are the User Plane Function (UPF) and the Data Network (DN).
    a. UPF (User Plane Function): Handles data forwarding between the RAN and the Data Network (DN). It is the anchor point for user data and implements packet routing, forwarding, and QoS enforcement. Interfaces: N3 which connects the RAN and UPF; N6: Connects UPF to Data Network (DN); N9 Connects multiple UPFs (for distributed architectures).
    b. DN (Data Network): Represents external networks such as the Internet or private enterprise networks.
3. Control Plane Functions: The control plane consists of several Network Functions (NFs), grouped into two main categories:
    a. Subscriber/Data Management NFs
        i. UDM (Unified Data Management): Manages subscriber data and authentication credentials.
        ii. AUSF (Authentication Server Function): Handles authentication for UEs.
        iii. UDR (Unified Data Repository): Centralized repository for structured data used by other NFs.
        iv. UDSF (Unstructured Data Storage Function): Stores unstructured data (marked as "under development" in the image).
    b. Signaling NFs
        i. NEF (Network Exposure Function): Exposes network capabilities and events to external applications (under development).
        ii. NSSF (Network Slice Selection Function): Selects appropriate network slices for UEs.
        iii. NRF (NF Repository Function): Allows NFs to register and discover each other.
        iv. SCP (Service Communication Proxy): Optimizes and manages service-based communication between NFs.
    c. Core Control NFs
        i. PCF (Policy Control Function): Manages policy rules for network behaviour (e.g., QoS, charging).
        ii. AMF (Access and Mobility Management Function): Handles connection and mobility management for UEs.
        iii. SMF (Session Management Function): Manages session establishment, modification, and release.
4. External and Future Functions: these functions are shown as "available in the future," indicating planned or optional features:
    a. AF (Application Function): Interacts with the core for application-level policy and QoS control.
    b. NWDAF (Network Data Analytics Function): Provides analytics and insights for network optimization.

Table 3 summarizes the network features included in each of the considered 5G Core Opensource solutions.

Table 3 Summary of functionalities provided by 5G core implementations

Network Function	Free5GC (3GPP **R15+)**	Open5GS , 3GPP R17 (Note 1)	OpenAirInterface, 3GPP R16+ (Note 2)

AMF	Yes	Yes	Yes
SMF	Yes	Yes	Yes
UPF	Yes (multiple, ULCL)	Yes	Yes
AUSF	Yes	Yes	Yes
UDM	Yes	Yes	Yes
UDR	Yes	Yes	Yes
PCF	Yes	Yes	Yes
NRF	Yes	Yes	Yes
NSSF	Yes	Yes	Yes
NEF (Note 3)	Yes (since v3.4.4)	Yes	Yes
CHF	Yes	Yes	Yes (limited)
N3IWF	Yes	Yes	Yes
TNGF	Yes	Yes	No (or limited)
AF (Note 4)	Yes (external)	Yes	Yes
UDSF (Note 5)	No	No	No
Webconsole (OAM)	Yes	Yes (integrated monitoring)	Yes (external tools)
Network Slicing	Yes	Yes	Yes
Service-Based Architecture (SBA)	Yes	Yes	Yes

Note 1. *Open5GS supports all main 5G Core functions according to 3GPP R17, although the documentation does not list all non-3GPP gateway variants in as much detail as Free5GC.*
Note 2. *OpenAirInterface supports the main 5G Core and RAN functions, but some optional or advanced gateway functions (such as TNGF) may not be as developed as in Free5GC or Open5GS. Always consult the specific component documentation.*
Note 3. *Free5GC supports NEF since recent versions (v3.4.4+), Open5GS and OAI also support NEF.*
Note 4. *Free5GC supports the interaction of external AFs (not always as an internal component), Open5GS and OAI also support AFs.*
Note 5. *None of the three solutions support UDSF in the current standard implementation.*

7.1.1. Deployment options for security evaluation

Modern open-source 5G Core solutions are designed to support flexible, scalable, and cloud-native deployments. Their architectures allow operators, researchers, and developers to choose the deployment model that best fits their environment, ranging from traditional virtual machines to advanced Kubernetes-based orchestration.

Virtual machine deployment: Each network function or the entire 5G Core can be installed on one or more virtual machines, typically running Linux distributions such as Ubuntu. Figure shows a generic example of this type of configuration, where each part of 5G Networks is implemented as an individual virtual machine, and operator only needs to select a concrete distribution. This is the simplest solution, ideal for small-scale labs, educational purposes, or initial testing where isolation and simplicity are priorities.

- Free5GC provides detailed guides for deployment on virtual machines, particularly optimized for VirtualBox [22] . The process includes cloning existing VMs, configuring host-only networks, and installing specific kernel modules such as gtp5g. The documentation provides specific examples for multi-VM configurations, where different network functions can be distributed among separate virtual machines.
- Open5GS supports deployments on multiple Linux distributions through virtual machines, including CentOS Stream 8 and Ubuntu [24] . The platform offers support for Vagrant, facilitating automated creation of virtualized environment[25] . Deployments can be configured with multiple VMs to separate control plane and user plane functions.
- OAI provides documentation for deployments on virtual machines for both core and radio access network components[26] . The platform supports cloud-based configurations where core components are deployed on OpenStack-based systems [27] . Deployments can include distributed configurations with separate VMs for HSS, MME, and SP-GW.

Container deployment: network functions are packaged as Docker containers, enabling lightweight, portable, and reproducible deployments. All three platforms offer Docker images and Compose files, allowing users to bring up the core and its dependencies in a single host or across multiple hosts. Figure shows a very simple deployment of NF's based on containers, individually managed. Operator decides the amount and characteristics of each deployed NF. In this case, proposed solutions are suitable for development, CI/CD pipelines, and environments where rapid deployment and teardown are needed.

- Free5GC offers a complete containerization solution through free5gc-compose, which uses Docker Compose to orchestrate all 5G Core components [28] . The solution requires privileged containers for tunnel interface creation and includes support for building images both locally and remotely from Docker Hub [29] . The deployment includes specific network configurations for communication between containers and support for different network profiles [30] .
- Open5GS provides multiple containerization options, including official implementations and third-party projects [31] . The Borjis131 solution offers Docker images for amd64 and arm64 architectures, with support for different Open5GS versions. Deployments can be configured for both single-host and multi-host scenarios, with support for control plane and user plane separation [32] .
- OAI supports containerized deployments with Docker Compose for all its 5G Core components [33] . The platform includes automation scripts for starting containers and verifying service status [34] . Deployments can include additional components such as IMS (IP Multimedia Subsystem) and RAN simulators.

Kubernetes-based deployment: The most advanced and scalable option, leveraging Kubernetes to orchestrate network functions as microservices (pods) across a cluster. It is recommended for production, large-scale testbeds, and research environments requiring high availability, auto scaling, and dynamic resource management. Basically, each NF runs as a separate pod, allowing independent scaling and upgrades, as figure shows. This solution provides the largest integration with monitoring (e.g., Monarch for Open5GS), persistent storage (MongoDB, MySQL), and network plugins (OVS, Multus), supporting multi-node and multi-slice deployments for advanced 5G scenarios.

- Free5GC offers multiple options for Kubernetes deployment, including projects such as free5gc-k8s and free5gc-helm [35] . The solution requires OVS bridge configuration for communication between nodes and installation of the gtp5g kernel module on nodes running UPF [36] . Deployments can include multi-node configurations with network functions distributed among different clusters [37] .
- Open5GS provides Helm charts for Kubernetes deployment, with support for microservices architectures where each network function runs as a separate pod [38] . The solution includes support for multiple network slices with dedicated SMF and UPF instances [39] . Deployments can be integrated with monitoring systems such as Monarch for advanced metrics.

- OAI supports Kubernetes deployments through Helm charts, including integration with SDR hardware via specific device plugins [40] . The solution can be deployed on multi-node clusters with support for different network configurations [41] . Deployments include support for end-to-end use cases with real radio equipment.

7.1.2. Performance and resource analysis

Table 4. Deployment support of 5G core implementations

Solution	Virtual Machines	Containers	Kubernetes
Free5GC	✓ VirtualBox/VMware support ✓ Multi-VM configuration ✓ Detailed documentation	✓ Native Docker Compose ✓ Images on Docker Hub ✓ Simplified configuration	✓ Available Helm charts ✓ Multi-node support ✓ OVS integration
Open5GS	✓ Multi-distribution support ✓ Vagrant automation ✓ CP/UP separation	✓ Multiple implementations ✓ Multi-architecture support ✓ Flexible configuration	✓ Native microservices ✓ Network slicing ✓ Integrated monitoring
OpenAirInterface	✓ Cloud-based support ✓ Distributed configuration ✓ RAN/Core integration	✓ Complete Docker Compose ✓ Automation scripts ✓ IMS components	✓ SDR device plugins ✓ End-to-end deployment ✓ Real hardware support

Table 4 summarizes the main aspects to be considered regarding the deployment options of each implementation. According to recent comparative evaluations, each platform presents distinctive performance characteristics [29] :

- Open5GS provides the best control plane procedure latencies
- OpenAirInterface offers the highest user plane data rates
- Free5GC shows the lowest system resource consumption

In terms of RAM consumption, Free5GC proved to be the most efficient, while Open5GS showed a RAM usage 165.42% higher than Free5GC [19] . For Kubernetes deployments, studies indicate that the overhead introduced by containerization and orchestration is minimal, resulting in only a 1.5% degradation in throughput and latency [20] . All in all, each deployment options could be used depending on concrete requirements for each scenario, but some recommendations could be considered.

- For Development and Testing Environments
  - Free5GC with Docker Compose: Ideal for rapid development and testing due to its simple configuration and lower resource consumption
  - Virtual machines: Recommended for lab environments with strict isolation requirements
- For Production Deployments
  - Open5GS on Kubernetes: Optimal for production environments requiring low control plane latency and automatic scaling capabilities
  - OpenAirInterface on Kubernetes: Recommended for deployments requiring maximum data throughput and integration with SDR hardware
- For Hybrid Deployments

- Multi-modal configurations: Combination of containers for development and Kubernetes for production, allowing gradual migration and continuous testing

The three analysed solutions offer viable options for deploying open source 5G cores, each with specific advantages depending on the use case. Free5GC stands out for its resource efficiency and ease of configuration, Open5GS for its control plane performance and stability, while OpenAirInterface excels in data throughput and integration capabilities. The choice of deployment modality should consider factors such as performance requirements, resource availability, scalability needs, and management complexity.

7.2. Security evaluation tools and techniques

7.2.1. Scanning tools for 5GC

Although Open RAN technologies and open source 5G Core implementations, described in this report, introduce new architectures and challenges, many tools previously used for security testing in IT or traditional telecoms environments also remain useful in the Open RAN/5G ecosystem. Tools for port scanning, vulnerability detection, configuration auditing or fuzzing can be successfully applied to security testing of open 5G solutions, since fundamental threats - such as unintended service exposure, misconfiguration or known software vulnerabilities - also remain valid in the new architectures. Additionally, the openness and modularity of Open RAN components and the reliance on standard protocols and operating systems (e.g. Linux, containerization) increase attack surface while making these tools often compatible without major modifications [42] . This chapter presents the common tools used for security analysis of Open RAN and 5G Core.

7.2.1.1. Port scanning

Port scanning is one of the fundamental steps in the security analysis of any network system, including Open RAN solutions and 5G Core components. It allows the identification of services and applications exposed to the external networks, the exposure of network services and the detection of potential attack vectors at an early stage of testing. Introducing such tools into the testing process not only allows the detection of unauthorised or unnecessarily open services, but also helps validate configuration settings and mitigate attack surface of the entire 5G infrastructure. In this way, attack entry points and anomalies can be identified early, before they are exploited by potential attackers.

The most popular tools are [43] :

- Nmap[5] – one of the most versatile network scanning tool, providing information not only on ports, but also on services, their versions and configuration details. However, its detailed analysis comes at the cost of lower scanning speed compared to the next two tools.
- Masscan[6]– the fastest of the listed port scanning tools, capable of scanning the entire IPv4 address space in minutes, but it provides significantly less information about services or configurations
- Zmap - the tool is optimized for high-speed scanning of a single port across large address spaces, making it ideal for Internet-wide surveys [44] .

7.2.1.2. Vulnerability scanning

Vulnerability scanning is one of the key steps in ensuring security in traditional IT systems as well as in Open RAN and 5G Core environments. These tools allow the automatic identification of known vulnerabilities

[5] https://nmap.org/

[6] https://www.kali.org/tools/masscan/

(CVEs), misconfigurations and potential weaknesses in operating systems, applications or network devices, including virtualized and containerized components commonly used in 5G infrastructures. Appropriately selected solutions enable both regular security audits and continuous monitoring of new threats, supporting risk management and compliance maintenance processes.

- OpenVAS[7] - Network vulnerability scanner, allows the scanning of hosts, services, networks and selected configuration items. Has integration with CVE databases (open source)
- Wuzah[8] - a SIEM platform with a module for vulnerability scanning (in continuous mode). It scans operating systems, applications, configuration files, services and ports, as well as system configuration for compliance with regulations and standards. Has integration with CVE databases (open source)
- Lynis[9] - a tool for auditing systems, mainly operating systems, focused on detailed configuration analysis.
- Trivy[10] - container vulnerability scanner (Docker, Kubernetes), detects vulnerabilities in images and application dependencies used in Open RAN/Core (if the environment is containerised).

7.2.1.3. Fuzzing

Fuzzing is a technique that involves automatically generating and sending unusual or random input to applications, services or protocols under test in order to detect bugs, vulnerabilities and unexpected behaviour like crashes, hangs, or unintended outputs. Widely used in IT environments, this method is also becoming increasingly important in the security analysis of Open RAN and 5G Core solutions - especially where web interfaces, APIs or custom communication protocols are available. The use of fuzzing tools makes it possible to detect vulnerabilities that traditional vulnerability scanners do not reveal, such as logic errors, data leaks or buffer overflow vulnerabilities, and thus significantly increases the effectiveness of security testing. Examples of tools include ZAP[11] (web fuzzing) or Peach Fuzzer[12] and boofuzz[13] (network fuzzing).

7.2.2. Vulnerabilities of SCTP based interfaces

Different interfaces, such as N2 of the 5G core or E2 of O-RAN, are based on the SCTP transport protocol, which provides the necessary transport functionalities for reliable exchange of control messages between these entities. From an implementation perspective, components employing the SCTP protocol adopt either a server or client role based on their position within the corresponding interface architecture. In the context of the N2 interface, the AM plays the role of a SCTP server, while the gNB node behaves as the client. In the following a detailed analysis of the communication flow established between these components is presented. Figure 28 shows the main procedures used by SCTP for both the establishment and maintenance of network associations.

[7] https://openvas.org/
[8] https://wazuh.com/
[9] https://cisofy.com/lynis/
[10] https://github.com/aquasecurity/trivy
[11] https://www.zaproxy.org/
[12] https://peachtech.gitlab.io/peach-fuzzer-community/
[13] https://boofuzz.readthedocs.io/en/stable/

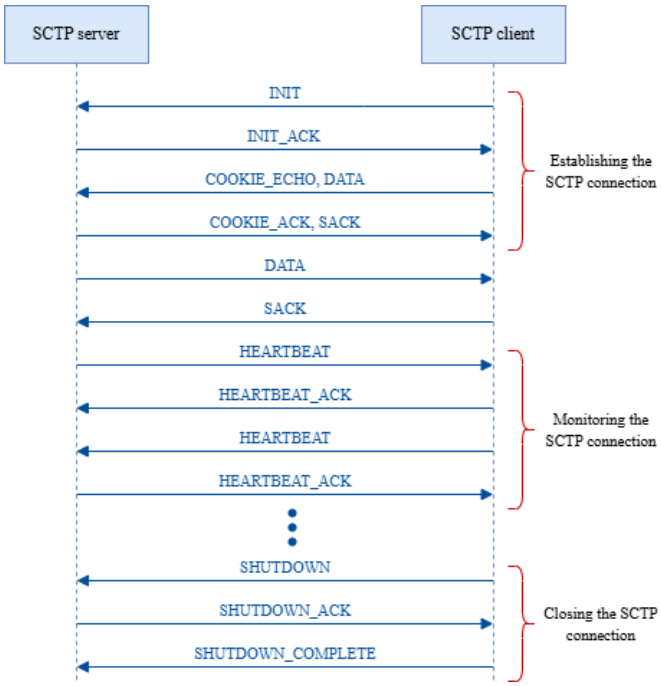

Figure 28 Main SCTP procedures

- SCTP Connection Establishment: The process begins with the initiation of an SCTP connection, which follows the four-way handshake mechanism. First, the client initiates the connection by sending an INIT message to the server. This message specifies parameters such as supported streams and an initial verification number. The server responds with an INIT ACK that includes a locally generated State Cookie to protect against SYN flooding attacks. Subsequently, the client responds with a COOKIE ECHO, which returns the cookie as proof that it correctly received the previous message and is willing to establish the association. Finally, the server sends a COOKIE ACK, confirming the validity of the cookie and thus completing the establishment of the SCTP association from the client's perspective, who can then begin sending DATA messages. If data has been sent through DATA chunks during this message exchange, SACK (Selective Acknowledgment) messages play a crucial role, as they allow each endpoint to precisely confirm the reception of DATA chunks. Unlike cumulative acknowledgment, SACK enables the identification of gaps in the sequence of received chunks, which improves the reliability and efficiency of the control channel, ensuring that data is delivered without duplication or loss.
- Connection Monitoring: During the lifetime of the association, both endpoints actively participate in connection maintenance through the use of HEARTBEAT messages. Each endpoint periodically sends HEARTBEAT packets to verify the accessibility of available IP routes, to which the counterpart responds with HEARTBEAT ACK. This mutual verification is essential, as SCTP can operate in multihoming mode, which allows traffic redirection to alternative routes in case of failures, thus guaranteeing service continuity.
- Connection Closure: the connection is concluded with the orderly termination of the SCTP association, initiated by the client through sending the SHUTDOWN message. This action indicates that the client node has finished data transmission, and request controlled session closure. The server responds with SHUTDOWN ACK, ensuring that all data has been received, and the client completes the procedure with the SHUTDOWN COMPLETE message. This orderly closure prevents information loss and ensures that all pending messages have been delivered before releasing network resources.

7.2.2.1. Attack Methodology

The methodology employed for security evaluation SCTP is based on a specific threat model that establishes the attacker's conditions and capabilities. The security analysis is conducted under the assumption that the adversary has previously gained access to the network where critical components operate, namely the O-RAN or 5G components. This premise, although restrictive, reflects a realistic scenario in the cybersecurity context where attackers can compromise perimeter security through various penetration techniques.

It is assumed that the attacker has achieved a privileged position within the network infrastructure, possibly through intrusion into an existing node via techniques such as reverse shell, exploitation of software vulnerabilities, or social engineering. This internal access capability allows the attacker to perform passive network traffic monitoring activities, as well as the generation and transmission of malicious packets specifically directed at the components of the involved architectures.

The attacks developed in this study fundamentally exploit vulnerabilities inherent to the SCTP protocol. The SCTP protocol, although robust in terms of reliability and connection management, presents certain characteristics that can be maliciously exploited when adequate security measures are not implemented. Specifically, the association control mechanisms, state management, and integrity verification of SCTP constitute attack vectors that enable the execution of denial-of-service attacks and identity spoofing.

For the practical implementation of attacks, specialized scripts have been developed using the Python programming language in combination with the Scapy library. This technological choice is justified by the flexibility that Scapy offers for low-level network packet construction and manipulation, allowing precise control over SCTP protocol header fields. The implemented scripts have the capability to intercept, analyze, and generate malicious SCTP packets, replicating the necessary parameters to spoof identities and manipulate the state of associations between components that implement SCTP.

The experiments were conducted using a test platform specifically designed for this purpose, as illustrated in Figure 29. This platform consists of three virtual machines (VMs) deployed on the VirtualBox hypervisor, all interconnected through a Host-only network with subnet address 10.0.2.0/24.

Each virtual machine runs an Ubuntu Linux-based operating system. VM1, with IP address 10.0.2.50 and MAC address 08:00:27:46:32:df, hosts the SCTP server, and VM2, with IP address 10.0.2.100 and MAC 08:00:27:73:d8:81, implements the SCTP client. Communication between VM1 and VM2 is established through their respective virtual network interfaces (enp0s3).

Finally, VM3, configured with IP 10.0.2.150 and MAC 08:00:27:9a:72:54, represents the attacker node within the test environment. This node hosts the Python-developed scripts that carry out different types of security attacks directed at the system. This component aims to simulate malicious behaviors in the system's control plane, in order to evaluate the robustness and possible vulnerabilities of ACTP based interfaces.

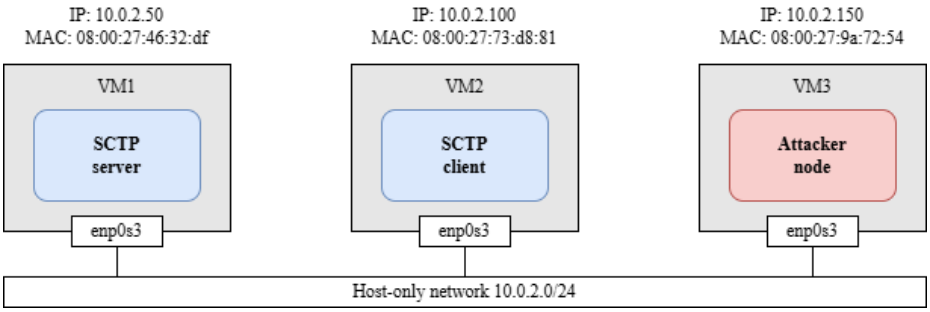

Figure 29 Evaluation setup

The security evaluation of critical telecommunications systems requires the identification and practical demonstration of potential vulnerabilities that may compromise the operational integrity of network infrastructure. In this context, this section describes the experimental implementation of two main

categories of attacks directed against the SCTP protocol: denial-of-service (DoS) attacks and identity spoofing attacks.

The implemented attacks exploit specific characteristics of the SCTP protocol to demonstrate how an adversary with network access can disrupt or manipulate communications between components that use the SCTP protocol. The first type of attack uses malicious ABORT messages to cause abrupt termination of legitimate associations, resulting in a denial of service that prevents subsequent reestablishment of communication. The second type of attack employs falsified SHUTDOWN messages to first destroy the existing connection and then execute identity spoofing using previously captured parameters.

Both attacks have been developed through scripts implemented in Python using the Scapy library, and have been experimentally validated on the test platform described previously. The detailed analysis of these attacks provides a fundamental basis for understanding the security limitations inherent to the SCTP protocol when implemented without additional protection mechanisms, thus establishing the necessary conceptual framework for subsequent evaluation of mitigation solutions.

7.2.2.2. Denial-of-Service Attack

A DoS (Denial of Service) attack is a type of cyberattack whose objective is to interrupt the availability of a system, service, or network, preventing legitimate users from accessing it. Typically, this is achieved by overloading the target's resources, such as bandwidth, processing capacity, or memory, through massive sending of malicious requests or data. In this study, a specific mechanism (the SCTP ABORT message) is used to carry out the DoS attack.

The ABORT message in the SCTP protocol has the primary function of abruptly terminating an existing SCTP association, releasing all associated resources without following the orderly closure procedure (as occurs with SHUTDOWN messages). This message is sent to notify the other endpoint that the association must terminate immediately due to an error condition or a situation that prevents continuing communication.

The ABORT message must be properly structured so that it is not rejected by the other endpoint, ensuring that it meets the protocol's expectations. One of the main parameters within the basic structure of the ABORT chunk is the T Bit and its relationship with the Verification Tag of the SCTP header.

The T Bit in the ABORT message plays a fundamental role in managing the authenticity and context of abrupt termination of an association. This bit, located in the ABORT chunk flags, indicates whether the message is sent as a response to a previously received chunk or if it originates autonomously from the sender endpoint. Its state determines how the Verification Tag, a key element of the SCTP header, should be interpreted and validated by the receiver.

When the T Bit is set to 1, the ABORT message is considered a reaction to a received chunk, such as an INIT, DATA, or any other that has generated a detectable error. In this case, the Verification Tag included in the SCTP header of the ABORT message must match the Verification Tag present in the chunk that triggered the response, allowing the receiver to verify that the message comes from the correct association and is related to the detected error context. This configuration ensures the integrity of the abort process by directly linking it to the previous interaction.

On the other hand, when the T Bit is set to 0, the ABORT is generated independently, typically due to an internal condition of the sender, such as a resource failure or a unilateral closure decision. In this situation, the Verification Tag of the SCTP header must correspond to the value that the receiver expects, that is, the tag that identifies the sender within the association. This relationship allows the receiver to validate the legitimacy of the message using the expected identifier of the connection, guaranteeing that the abort comes from an authorized endpoint within the association.

The interaction between the T Bit and the SCTP header is, therefore, essential for maintaining coherence and security in abrupt termination, as it ensures that the receiver can process the ABORT message without ambiguities, avoiding rejections or confusion in association handling. This dynamic reflects the robust design of SCTP, which prioritizes reliability and traceability even in critical error scenarios.

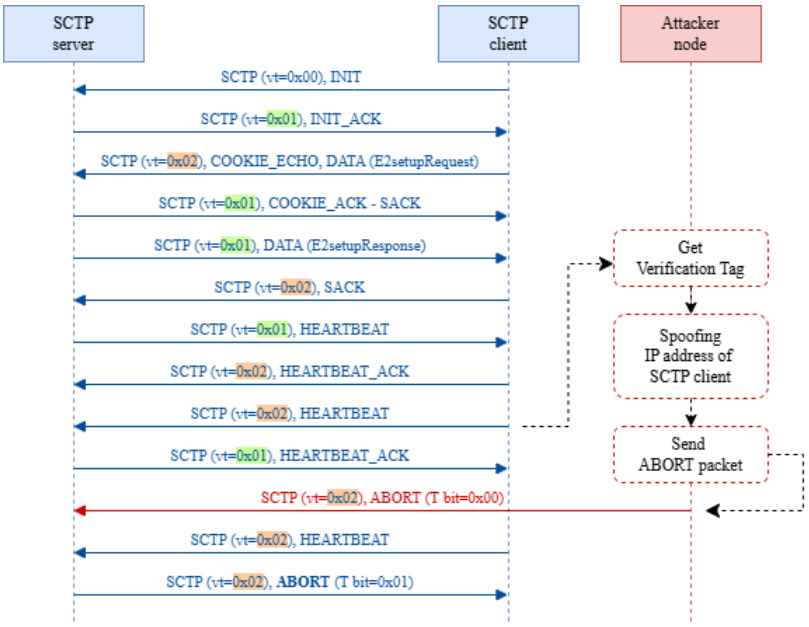

Figure 30 DoS attack through ABORT message

Figure 30 illustrates an attack executed by a malicious node through the use of the ABORT message. This node actively monitors the network, capturing packets with the objective of intercepting the HEARTBEAT message transmitted by the SCTP client. Once the desired packet is identified, the malicious node extracts the critical parameters from the SCTP header, including the source IP address, destination IP address, source port, destination port, and Verification Tag. With this information, the malicious node constructs an ABORT packet, setting the T Bit to 0 to signal a unilateral termination of the association. This packet is sent to the SCTP server, simulating being the client by employing the IP, port, and Verification Tag extracted from the latter. Upon receiving the ABORT packet, the server interprets that the client has requested the destruction of the SCTP association on the server side. Meanwhile, the client, unaware of the spoofing, continues sending HEARTBEAT packets to the server. Upon detecting the reception of a HEARTBEAT after having processed an ABORT presumably originated by the client, the server identifies an anomaly. In response, it sends an ABORT packet to the client, setting the T Bit to 1 and including in the SCTP header the Verification Tag of the message that caused the inconsistency. Finally, the client, upon receiving this second ABORT, proceeds to destroy the association on the client side.

7.2.2.3. Identity Spoofing Attack

An identity spoofing attack is a type of cyberattack in which an adversary impersonates a legitimate system entity, fraudulently adopting its credentials and identification parameters to establish unauthorized communications.

This section implements an identity spoofing attack that exploits vulnerabilities in the SCTP protocol. This attack uses the SHUTDOWN message as an initial vector to disconnect the legitimate node, subsequently creating an appropriate scenario for the attacker to establish a new association by spoofing the identity of some of the involved components.

One of the critical processes in SCTP is the orderly closure of an association, which ensures that all pending data in the send and receive queues are delivered before terminating the connection. According to RFC 9260, the closure procedure involves a sequence of three messages: SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE, each with specific roles in coordinating the closure.

The SHUTDOWN message is sent by the endpoint that initiates the association closure, indicating that it will not send more data and has completed the transmission of all pending data. One of the critical parameters

of SHUTDOWN is the Cumulative TSN Ack. The TSN (Transmission Sequence Number) is a unique identifier assigned to each data chunk in an SCTP association, used to guarantee ordered and reliable delivery. The Cumulative TSN Ack in the SHUTDOWN message confirms to the receiver that all data with TSN less than or equal to the specified one has been correctly received, allowing the receiver to release resources associated with that data and proceed with the closure.

The SHUTDOWN ACK message contains no additional parameters beyond its header, as its main purpose is to confirm that the SHUTDOWN receiver has processed the message and is ready to terminate the association. This message is sent once the receiver has delivered all pending data in its send queue.

The SHUTDOWN COMPLETE message is the final step of the closure process. This message confirms that both endpoints have completed data delivery and that the association can be terminated. Like the SHUTDOWN ACK, it includes no additional parameters, as its function is merely confirmatory. Once this message is received, both endpoints release resources associated with the association.

The closure process in SCTP is robust against failures, as it includes retransmission mechanisms for SHUTDOWN and SHUTDOWN ACK messages in case of loss. Compared to abrupt closure (via the ABORT message), orderly closure with SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE guarantees data integrity and controlled resource release, which is especially valuable in applications requiring high reliability, such as telecommunications and messaging systems.

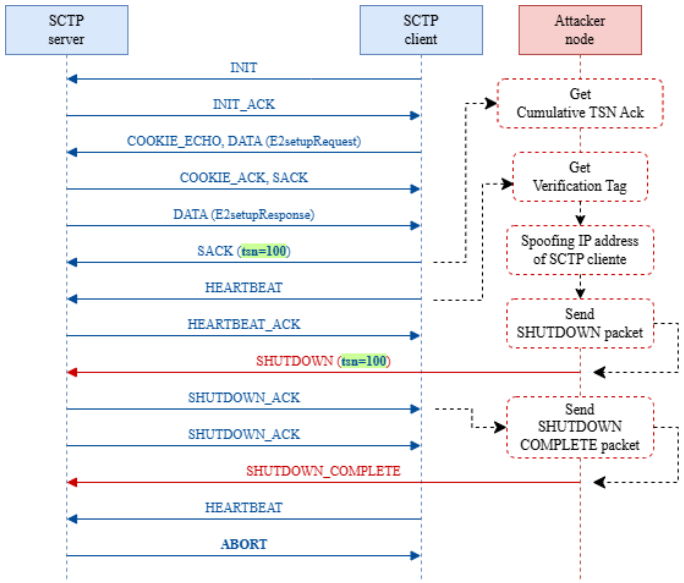

Figure 31 Identity spoofing attack through SHUTDOWN message

Figure 31 presents the procedure carried out by a malicious node to execute an attack using the SCTP SHUTDOWN message. Initially, the malicious node performs active network monitoring with the purpose of capturing an SCTP packet containing a SACK chunk. From this packet, it extracts the Cumulative TSN Ack value, an essential parameter for constructing the malicious message. Subsequently, the malicious node continues network supervision to intercept a HEARTBEAT packet sent by the SCTP client. Analogously to the procedure employed in the DoS attack described in the previous section, the malicious node extracts from this packet the critical parameters of the SCTP header, including source and destination IP addresses, source and destination ports, and the Verification Tag.

With this information, the malicious node constructs an SCTP packet that incorporates a SHUTDOWN chunk, using the previously obtained Cumulative TSN Ack value. In this packet, the malicious node does not use its own IP address as the source, but instead adopts the SCTP client's IP address, spoofing its identity. Thus, the SCTP server interprets that the SHUTDOWN packet legitimately comes from the client. Upon receiving the SHUTDOWN message, the server responds by sending a SHUTDOWN ACK packet to the client. However, the

client discards this packet, as, according to its internal state, it does not expect a SHUTDOWN ACK since it has not initiated an SCTP connection closure process.

Since the client does not respond to the SHUTDOWN ACK, the server, assuming the packet was lost, retransmits the SHUTDOWN ACK. Meanwhile, the malicious node, which has continued monitoring the network, detects the SHUTDOWN ACK packet sent by the server. In response, it generates a SHUTDOWN COMPLETE packet, again using the client's IP address as the source, perpetuating the spoofing. Since the client has no knowledge that the SCTP connection has been closed by the malicious node, it continues sending HEARTBEAT packets as part of its standard mechanism to keep the connection active. Upon receiving one of these HEARTBEAT packets, the server identifies an anomaly in the communication process with the client, as it considers that the connection had already been closed. Consequently, the server sends an ABORT packet to the client, abruptly terminating the SCTP connection and completing the attack's impact.

7.2.2.4. Spoofing Attack

The attack based on the SHUTDOWN message described above generates three significant consequences: i) the elimination of the SCTP association established between the involved components has occurred; ii) the SCTP server has removed the client from its internal records, releasing the identifiers associated with the corresponding O-RAN or 5G interface; and iii) at the application level, the client remains unaware of these events, resulting in its inadvertent disconnection from the SCTP server.

These circumstances can be exploited by a malicious node to execute a spoofing attack based on the availability of information from the components involved during the SCTP association establishment process.

Different standards propose the use of TLS to protect interfaces that use the SCTP protocol, but this is not sufficient. The TLS handshake is performed after the SCTP association phase is completed, that is, after the complete exchange of the four messages of the SCTP association (INIT, INIT ACK, COOKIE ECHO, COOKIE ACK). Even during the TLS handshake process, the SCTP association exists but TLS is not yet established, therefore SCTP and TLS messages travel unprotected until the handshake is completed. This opens a vulnerable time window where the attacker can carry out DoS and spoofing attacks

7.3. Vulnerabilities evaluation of the N2 interface

This section evaluates vulnerabilities in the three 5G Core Network implementations described in Section 7.1: Free5GC, Open5GS, and OpenAirInterface. Each implementation utilizes UERANSIM software as the Radio Access Network (RAN), which emulates gNB functionalities.

Among the 5G core network interfaces identified in Section 7, the security evaluation focuses on the N2 interface, as this interface employs the SCTP protocol. Specifically, the N2 interface operates between the gNB and the AMF entity within the core network.

Section 7.2.2 examined two attack methodologies targeting this protocol: Denial-of-Service (DoS) attacks and Identity Spoofing attacks. The Python scripts implementing these attacks, as described in Section 7.2.2, are employed in this section to evaluate SCTP protocol vulnerabilities within the N2 interface context of the 5G core networks under study. In this configuration, the AMF functions as the SCTP server, executing on a virtual machine (VM) with IP address 10.10.10.155, while the gNB node operates as the client on a VM with IP address 10.10.10.7.

The vulnerability assessment was performed on all three 5G core network implementations (Free5GC, Open5GS, and OpenAirInterface), with each evaluation producing identical outcomes regarding network behavior, frame exchange patterns, and attack impact on the core network entities. To avoid redundancy in presenting duplicate results, this section provides a comprehensive analysis using Free5GC as the representative implementation, with the understanding that the findings apply equally to Open5GS and OpenAirInterface.

7.3.1. Denial-of-Service Attack

The Denial-of-Service (DoS) attack exploits the SCTP ABORT packet to maliciously terminate established associations. The attacker node performs network sniffing to obtain the required SCTP association parameters, including IP addresses, ports, and verification tags, before executing the attack.

Figure 32 shows the Wireshark capture of frames exchanged between the AMF (Free5GC) and the gNB (UERANSIM) from the initial establishment of the SCTP association through the conclusion of the DoS attack.

	Time	Source	Destination	Protocol	Length	Info	Src MAC	Dst MAC
9	1.573690191	10.10.10.7	10.10.10.155	SCTP	82	INIT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
10	1.573752527	10.10.10.155	10.10.10.7	SCTP	306	INIT_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
11	1.574252482	10.10.10.7	10.10.10.155	SCTP	278	COOKIE_ECHO	08:00:27:0a:35:56	08:00:27:e6:e5:d1
12	1.574282191	10.10.10.155	10.10.10.7	SCTP	50	COOKIE_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
18	1.575636293	10.10.10.7	10.10.10.155	NGAP	134	NGSetupRequest	08:00:27:0a:35:56	08:00:27:e6:e5:d1
19	1.575667551	10.10.10.155	10.10.10.7	SCTP	62	SACK (Ack=0, Arwnd=106427)	08:00:27:e6:e5:d1	08:00:27:0a:35:56
20	1.593122176	10.10.10.155	10.10.10.7	NGAP	118	NGSetupResponse	08:00:27:e6:e5:d1	08:00:27:0a:35:56
21	1.593524746	10.10.10.7	10.10.10.155	SCTP	62	SACK (Ack=0, Arwnd=106442)	08:00:27:0a:35:56	08:00:27:e6:e5:d1
28	11.1183805…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
29	11.1191328…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
30	17.7743715…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
31	17.7750941…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
47	23.9210811…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
48	23.9219612…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
52	30.0623669…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
53	30.0631127…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
57	34.7876355…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
58	34.7876848…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
107	34.8054751…	10.10.10.7	10.10.10.155	SCTP	60	ABORT	08:00:27:b9:64:67	08:00:27:e6:e5:d1
122	67.5748987…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
123	67.5749617…	10.10.10.155	10.10.10.7	SCTP	50	ABORT	08:00:27:e6:e5:d1	08:00:27:0a:35:56

Figure 32 Wireshark capture during DoS attack

Frames 9 through 12 execute the SCTP association establishment procedure, while frames 18 through 21 perform the NG Setup procedure, which establishes the initial connection and signaling relationship between the gNB and the AMF.

Figure 33 displays the startup logs generated on both UERANSIM and Free5GC VMs, confirming successful completion of both the SCTP association and the NG Setup procedure.

EURANSIM

```
UERANSIM v3.2.6
[2025-06-19 08:48:00.626] [sctp] [info] Trying to establish SCTP connection... (10.10.10.155:38412)
[2025-06-19 08:48:00.633] [sctp] [info] SCTP connection established (10.10.10.155:38412)
[2025-06-19 08:48:00.633] [sctp] [debug] SCTP association setup ascId[39]
[2025-06-19 08:48:00.633] [ngap] [debug] Sending NG Setup Request
[2025-06-19 08:48:00.645] [ngap] [debug] NG Setup Response received
[2025-06-19 08:48:00.645] [ngap] [info] NG Setup procedure is successful
```

Free5GC

```
2025-06-19T08:48:00.917460505Z [INFO][AMF][Ngap] [AMF] SCTP Accept from: 10.10.10.7:60151
2025-06-19T08:48:00.927864113Z [INFO][AMF][Ngap] Create a new NG connection for: 10.10.10.7:60151
2025-06-19T08:48:00.929095972Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:60151] Handle NGSetupRequest
2025-06-19T08:48:00.929555917Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:60151] Send NG-Setup response
```

Figure 33 VMs logs during gNB and UERANSIM start-up

The SCTP HEARTBEAT packets appearing from frame 28 onward correspond to the standard SCTP protocol procedure for monitoring and verifying the availability and status of communication paths between SCTP association endpoints.

At frame 107, the malicious node transmits an ABORT packet to Free5GC while spoofing the IP address and port of the gNB. Figure 34 shows the Free5GC VM log, which demonstrates that Free5GC correctly receives the ABORT packet sent by the malicious node, displaying the message "Handle SCTP Notification[addr: 10.10.10.7:60151]". As a consequence of receiving this packet, Free5GC removes the gNB from its internal registry, indicated in the log by the message "Remove RAN Context[ID: <PlmnID: {Mcc:208 Mnc:93}, GNbID: 00000001>]".

Free5GC

```
2025-06-19T08:52:19.161166804Z [INFO][AMF][Ngap] Handle SCTP Notification[addr: 10.10.10.7:60151]

2025-06-19T08:52:19.161211811Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:60151] SCTP_ASSOC_CHANGE notification

2025-06-19T08:52:19.161223263Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:60151] SCTP state is SCTP_COMM_LOST,
close the connection

2025-06-19T08:52:19.161248874Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:60151] Remove RAN Context[ID: <PlmnID:
{Mcc:208 Mnc:93}, GNbID: 00000001>]

2025-06-19T08:52:19.161285324Z [ERRO][AMF][Ngap] Handle connection[addr: <nil>] error: connection reset by
peer

2025-06-19T08:52:19.161294642Z [INFO][AMF][Ngap] Handle SCTP Connection Error[addr: <nil>] - remove RAN

2025-06-19T08:52:19.161303392Z [WARN][AMF][Ngap] RAN context has been removed[addr: <nil>] SCTPConn:
SCTPWrite failed bad file descriptor
```

UERANSIM

```
[2025-06-19 08:52:51.632] [sctp] [debug] SCTP association shutdown (clientId: 2)

[2025-06-19 08:52:51.632] [sctp] [warning] Unhandled SCTP notification received terminate called after
throwing an instance of 'sctp::SctpError'

  what():  SCTP receive message failure: Transport endpoint is not connected

Aborted (core dumped)
```

Figure 34 VMs logs during ABORT attack

Since the gNB remains unaware that the malicious node sent the ABORT packet, it continues the standard procedure of transmitting HEARTBEAT packets to maintain SCTP communication with Free5GC, as shown in frame 122. Upon receiving the HEARTBEAT packet, Free5GC detects a communication anomaly with the gNB, having previously received an ABORT packet. This is reflected in the Free5GC logs with the message "Handle SCTP Connection Error[addr: <nil>] - remove RAN". To resolve this anomalous situation, Free5GC sends an ABORT packet to the gNB, as shown in frame 123, abruptly terminating communication at both SCTP and NGAP levels[13] . This action is also reflected in the UERANSIM logs with the message "SCTP association shutdown". Although the consequence of this attack is more severe for UERANSIM, resulting in an "Aborted (core dumped)" error, it should be noted that UERANSIM is merely an emulator and therefore not a robust RAN implementation.

7.3.2. Identity spoofing attack

The identity spoofing attack exploits the SCTP connection termination procedure through malicious SHUTDOWN packet injection. Figure 35 illustrates the frame exchange between the AMF (Free5GC) and the gNB (UERANSIM) from the initial establishment of the SCTP association through the conclusion of the spoofing attack.

	Time	Source	Destination	Protocol	Length	Info	Src MAC	Dst MAC
11	0.792163322	10.10.10.7	10.10.10.155	SCTP	82	INIT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
12	0.792242834	10.10.10.155	10.10.10.7	SCTP	306	INIT_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
13	0.792821890	10.10.10.7	10.10.10.155	SCTP	278	COOKIE_ECHO	08:00:27:0a:35:56	08:00:27:e6:e5:d1
14	0.792854248	10.10.10.155	10.10.10.7	SCTP	50	COOKIE_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
21	0.794043176	10.10.10.7	10.10.10.155	NGAP	134	NGSetupRequest	08:00:27:0a:35:56	08:00:27:e6:e5:d1
22	0.794066207	10.10.10.155	10.10.10.7	SCTP	62	SACK (Ack=0, Arwnd=106427)	08:00:27:e6:e5:d1	08:00:27:0a:35:56
23	0.810644887	10.10.10.155	10.10.10.7	NGAP	118	NGSetupResponse	08:00:27:e6:e5:d1	08:00:27:0a:35:56
24	0.811084852	10.10.10.7	10.10.10.155	SCTP	62	SACK (Ack=0, Arwnd=106442)	08:00:27:0a:35:56	08:00:27:e6:e5:d1
44	10.0709043…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
45	10.0717112…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
57	16.2149190…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
58	16.2160807…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
62	22.3589271…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
63	22.3596958…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
69	28.5029241…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT	08:00:27:e6:e5:d1	08:00:27:0a:35:56
70	28.5037870…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT_ACK	08:00:27:0a:35:56	08:00:27:e6:e5:d1
76	35.0890007…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
77	35.0890593…	10.10.10.155	10.10.10.7	SCTP	106	HEARTBEAT_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
125	35.1013894…	10.10.10.7	10.10.10.155	SCTP	60	SHUTDOWN	08:00:27:b9:64:67	08:00:27:e6:e5:d1
126	35.1014114…	10.10.10.155	10.10.10.7	SCTP	50	SHUTDOWN_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
135	36.1189094…	10.10.10.155	10.10.10.7	SCTP	50	SHUTDOWN_ACK	08:00:27:e6:e5:d1	08:00:27:0a:35:56
143	36.1318510…	10.10.10.7	10.10.10.155	SCTP	60	SHUTDOWN_COMPLETE	08:00:27:b9:64:67	08:00:27:e6:e5:d1
153	67.9149649…	10.10.10.7	10.10.10.155	SCTP	106	HEARTBEAT	08:00:27:0a:35:56	08:00:27:e6:e5:d1
154	67.9150199…	10.10.10.155	10.10.10.7	SCTP	50	ABORT	08:00:27:e6:e5:d1	08:00:27:0a:35:56

Figure 35 Wireshark capture during SHUTDOWN attack

Similar to the DoS attack, the initial frames through frame 24 correspond to SCTP association establishment and the NG Setup procedure. Subsequent frames through frame 77 execute the standard procedure for maintaining SCTP-level communication.

At frame 125, the malicious node sends a SHUTDOWN packet while impersonating the gNB. To accomplish this, the malicious node previously gathered information about SCTP association parameters, as well as the gNB's IP address and port. Figure 36 demonstrate correct reception of the SHUTDOWN packet by Free5GC, indicated by the message "SCTP_SHUTDOWN_EVENT notification, close the connection". As a consequence of receiving this packet, Free5GC confirms the shutdown request by sending a SHUTDOWN_ACK packet to the gNB (frame 126). Since the gNB is unaware that the malicious node previously initiated a shutdown process, it discards the SHUTDOWN_ACK packet, and UERANSIM indicates this in the logs with the message "Unhandled SCTP notification received".

Simultaneously, the malicious node maintains passive network monitoring to intercept the SHUTDOWN_ACK packet sent by Free5GC in order to complete the shutdown process. Upon detecting this packet, the malicious node sends a SHUTDOWN_COMPLETE packet while again spoofing the gNB's identity (frame 143). The Free5GC logs display the message "Remove RAN Context[ID: <PlmnID: {Mcc:208 Mnc:93}, GNbID: 00000001>]", confirming the action executed in Free5GC upon receiving the SHUTDOWN_COMPLETE packet.

The gNB continues the process of maintaining SCTP communication without knowledge of the previous actions carried out by the malicious node. For this reason, at frame 153, the gNB sends a HEARTBEAT packet to Free5GC. Reception of this packet at Free5GC is detected as an anomaly, with this event shown in the logs through the message "Handle SCTP Connection Error". To resolve this anomaly, Free5GC sends an ABORT packet to the gNB (frame 154), warning in the logs that it has again had to remove the gNB from its registries through the message "[WARN][AMF][Ngap] RAN context has been removed". Similarly, the UERANSIM logs indicate that communication with Free5GC has terminated abnormally, through the message "[error] Association terminated for AMF".

Free5GC

```
2025-06-19T09:07:01.853340320Z [INFO][AMF][Ngap] Handle SCTP Notification[addr: 10.10.10.7:36405]

2025-06-19T09:07:01.853474226Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:36405] SCTP_SHUTDOWN_EVENT
notification, close the connection

2025-06-19T09:07:01.853524611Z [INFO][AMF][Ngap][ran_addr:10.10.10.7:36405] Remove RAN Context[ID: <PlmnID:
{Mcc:208 Mnc:93}, GNbID: 00000001>]

2025-06-19T09:07:01.853589927Z [INFO][AMF][Ngap] Handle SCTP Connection Error[addr: 10.10.10.7:36405] -
remove RAN

2025-06-19T09:07:01.853613951Z [WARN][AMF][Ngap] RAN context has been removed[addr: 10.10.10.7:36405]

SCTPConn: SCTPWrite failed bad file descriptor
```

UERANSIM

```
[2025-06-19 09:07:32.274] [sctp] [debug] SCTP association shutdown (clientId: 2)

[2025-06-19 09:07:32.274] [sctp] [warning] Unhandled SCTP notification received

[2025-06-19 09:07:32.274] [ngap] [error] Association terminated for AMF[2]

[2025-06-19 09:07:32.274] [ngap] [debug] Removing AMF context[2]

terminate called recursively

Aborted (core dumped)
```

Figure 36 VMs logs during shutdown attack

7.3.3. Potential mitigation measurements

Basically, the main solution involves securing 5G interfaces through authentication and encryption, whether at the link layer (MACsec), network layer (IPsec), transport layer (SCTP), or application layer (Transport Layer Security (TLS)). However, perimeter solutions are also possible, such as implementing Intrusion Detection Systems (IDS) and intrusion prevention systems (IPS), which would allow real-time identification and mitigation of attacks, or using multi-factor authentication (MFA) methods to ensure that only authorized users and devices can access the network.

SCTP includes several important security options:

1. **Authentication**: SCTP can authenticate messages to ensure they come from a legitimate source. It uses a four-way handshake-based authentication mechanism. This process includes an authentication cookie that helps verify the identity of the communication participants.
2. **Packet Validation**: SCTP uses a verification tag in its header to uniquely identify association sessions, helping to protect against spoofing attacks.
3. **Encryption**: SCTP supports message encryption to protect sensitive information from eavesdropping and tampering. In a secure communication, data sent between two SCTP nodes can be encrypted using algorithms like AES (Advanced Encryption Standard), ensuring that only authorized nodes can decrypt and read the data.

These features make SCTP, at least in theory, a robust option for applications requiring secure and reliable data transport. However, 5G Core and O-RAN implementations often use the simpler version, which only includes the first two measures. RFC 4895 defines the SCTP AUTH chunk as a mechanism for the authentication and cryptographic integrity of SCTP messages, which is theoretically a logical option for protecting communications between 5G Core and O-RAN elements. Unfortunately, in practice, it is not used

in these environments for various technical reasons, such as the fact that not all commercial SCTP versions include it, making interoperability between equipment from different manufacturers difficult [45] .

On the other hand, 5G Core and O-RAN architectures are originally designed around higher-layer security models, prioritizing the use of widely standardized and supported protocols such as TLS (over SCTP) or IPsec (at the network level). These solutions offer more flexible and robust mutual authentication, end-to-end encryption, and key management mechanisms, adapted to distributed, virtualized, and multi-vendor architectures like those of 5G and O-RAN. Additionally, the use of HTTP/2 and REST APIs in the Service-Based Architecture (SBA) of the 5G Core promotes the integration of TLS as the de facto standard for protecting control and management communications [46] .

While the standards and guidelines of the O-RAN Alliance and 3GPP explicitly recommend the use of TLS (over SCTP or TCP) to protect control traffic between components such as the Near-RT RIC and E2 nodes, as well as IPsec to secure links between virtualized elements and distributed network functions, the most cited solution is MACsec. MACsec (Media Access Control Security, IEEE 802.1AE) is a layer 2 security standard that provides encryption, authenticity, and integrity for Ethernet frames. It uses the MKA (MACsec Key Agreement) protocol to negotiate session keys (SAK), derived from an association key (CAK) and a key name (CKN). Once the negotiation is complete, frames can be authenticated and encrypted using AES-GCM, and the flow is protected by sequencing the frames, thus ensuring protection against replay attacks. This approach is widely adopted in enterprise and operator environments to secure Ethernet infrastructure against eavesdropping, tampering, and internal attacks.

Thus, MACsec can be deployed between the Near-RT RIC and E2 nodes (O-DU, O-CU), if they are directly connected or on the same Ethernet segment (through a switch), between switches (hop-by-hop), or on direct links to routers, and if both ends support the standard. At each hop, the traffic is decrypted and re-encrypted, so the protection is local to each physical link, which may not be sufficient if the attacker could access the segment. However, there is an alternative that reduces the risk by using centralized credential management and access policies through 802.1X and RADIUS servers, which integrate authentication and key provisioning. In this scheme, a device requests access through 802.1X, the switch forwards the authentication to the RADIUS server, and after validation, the necessary keys are distributed to activate MACsec and protect the link.

If the traffic crosses a router (network segment change), MACsec protection ends and must be restarted in the next segment or complemented with solutions like IPsec for end-to-end protection.

7.4. Evaluating the Resilience of the 5G User Plane Function Under Targeted Network Attacks

UPF is central to 5G user plane operations, designed to meet the stringent performance demands of advanced 5G services. It connects with external IP networks hiding mobility related aspects from the external networks. Moreover, it performs different types of processing of the forwarded data, such as packet inspection, redirection of traffic and application of different data rate limitations. UPFs terminate various interfaces and protocols, executing key actions such as: (a) mapping traffic to tunnels via QoS Flow Identifier (QFI)-based Deep packet inspection (DPI) and protocol adaptation, (b) packet forwarding and steering, (c) traffic counting for charging and policy, (d) DPI for security and anomaly detection, and (e) queuing for service differentiation and delay assurance.

Leveraging 5G-Control and User Plane Separation (CUPS), multiple UPFs can be dynamically deployed to support edge capabilities and localized service delivery. The UPF supports several standardized interfaces to interact with both control and user plane entities:

- N4: Connects the UPF with the Session Manangement Function (SMF), enabling the SMF to control and configure the UPF.
- N3: Interfaces with the gNB-CU (Central Unit) to receive user plane data encapsulated in GPRS Tunnelling Protocol (GTP)-U. This is the main data path for user traffic entering the UPF from the RAN.
- N6: Connects the UPF to the external data network (e.g., the internet, enterprise networks, or Multiaccess Edge Computing (MEC) platforms. At this interface, the UPF performs traffic routing, Network address translation (NAT), firewalling, and enforces QoS and charging policies.
- N9: Enables communication between multiple UPFs in distributed deployments, particularly in CUPS architectures.
- DN-specific interfaces: In edge or MEC scenarios, the UPF may also interface directly with application functions via non-standard interfaces, allowing for optimized routing and low-latency delivery.

Given its central role in handling user traffic and enforcing policies, the UPF is a high-value target for adversaries, introducing an expanded attack surface for malicious entities. Its exposure to both internal and external interfaces, including N3 (from the RAN), N6 (towards external networks), and N4 (from the control plane), makes it susceptible to a variety of attack vectors such as packet flooding, malformed GTP-U messages, or control-plane manipulation. A successful attack on the UPF could result in traffic disruption, policy bypass, degraded quality of service, or even denial of service across multiple sessions. Therefore, securing the UPF and its interfaces is essential to ensure service continuity and network integrity. In this study, we investigate how the UPF performs under stress conditions caused by targeted attacks on its interfaces. Our goal is to assess its resilience and observe the impact on user traffic, with a focus on performance degradation, packet loss, and potential service interruptions.

7.4.1. Experimental Testbed

The physical infrastructure that we used for our experiments comprises a set of servers, optoelectronic switches, routers and physical links. All the physical resources are clustered into a (openstack) Private Cloud platform which is used to deploy and host all the 5G RAN & Core functions. The 5G Core Network is deployed through the *free5gc* open source platform while for the RAN side, UERANSIM was used (see Figure 37). A monitoring and data collection framework is in place to track system performance and resource utilization. Metrics from all compute and network resources are collected and exposed from agents to specific ports and are then retrieved and stored to a Prometheus database. Additionally, energy consumption metrics are retrieved from energy metering devices. All the metrics are visualized through Grafana. Finally, OSM was used as a Management and Network Orchestration (MANO) platform for the automated deployment of Control Plane and User Plane Core functions. To generate user traffic, a simple **iPerf** connection is established from the UE to a physical PC located outside the cloud infrastructure. Within this environment, a dedicated virtual machine (VM) acts as the attacker, using **hping** to launch DoS and SYN flood attacks targeting the UPF interfaces in order to assess resilience and performance under stress.

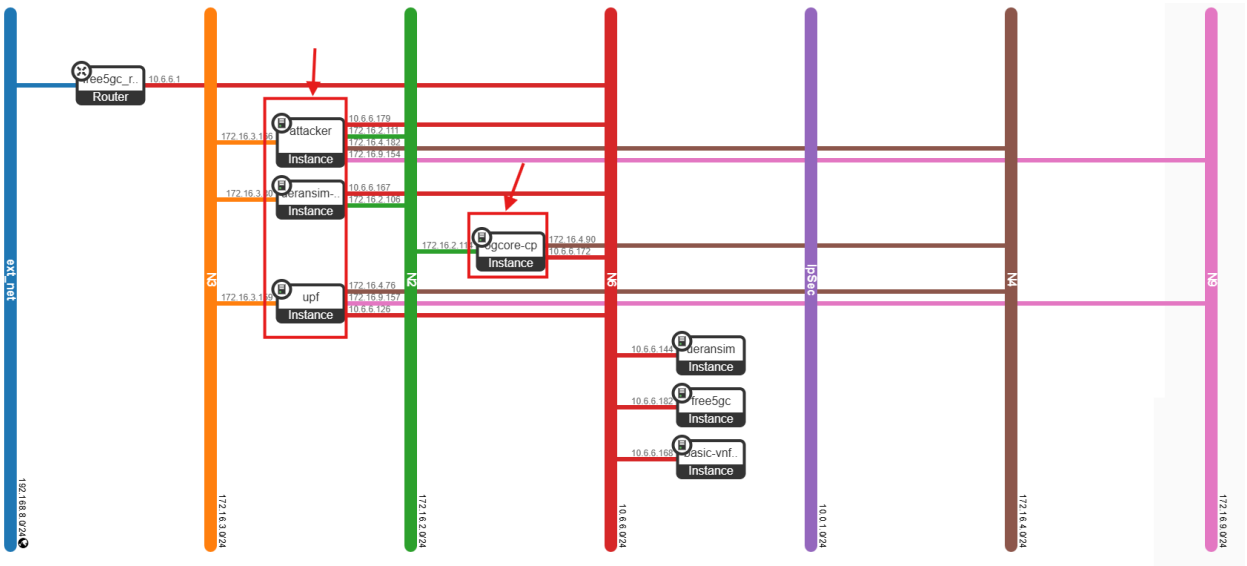

Figure 37: 5G Network Components in Private cloud

The successful deployment and operation of the testbed are illustrated through a series of figures. In Figure 38, we see the successful implementation of the 5GCN CUPS architecture, where the UPF is running in a dedicated VM, clearly separated from the other core network functions. In Figure 38 also shows the simulated gNB from UERANSIM that is successfully connected to the AMF. In Figure 39 the attached UE is visible, confirming that the RAN and core network components are properly communicating and that the UE is registered in the network. Figure 39 also shows a simple iPerf session initiated from the UE, generating traffic toward a PC located outside the cloud. Finally, Figure 40 displays Grafana dashboards illustrating real-time system monitoring, including network throughput and CPU usage during the iperf connection. We can see that the maximum throughput for the UE is 213 MB/s

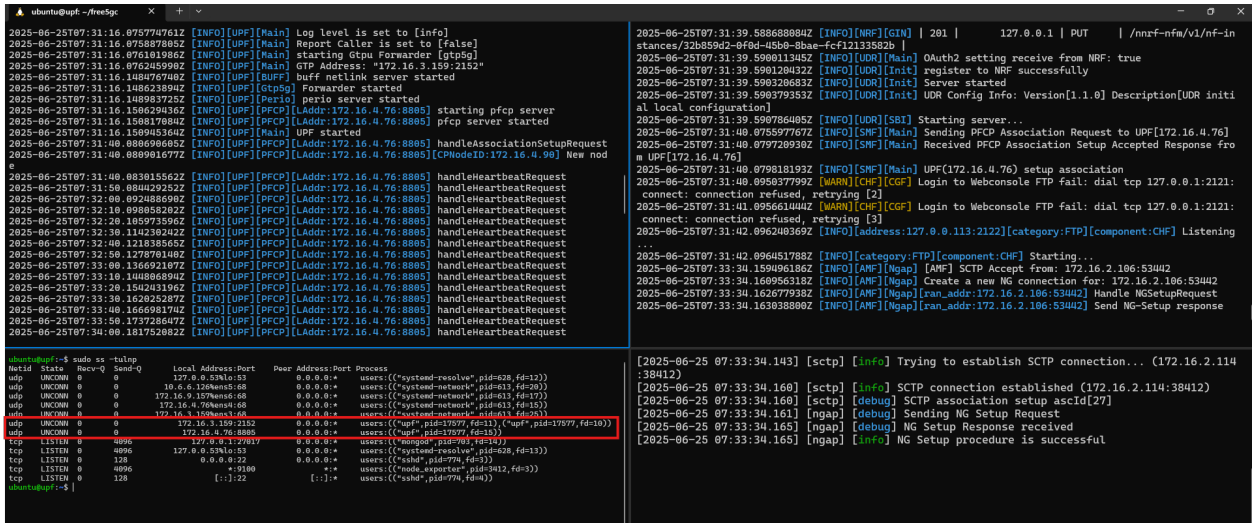

Figure 38: 5GC-CUPS instantiation and successful connection established between gNB and 5G-Core

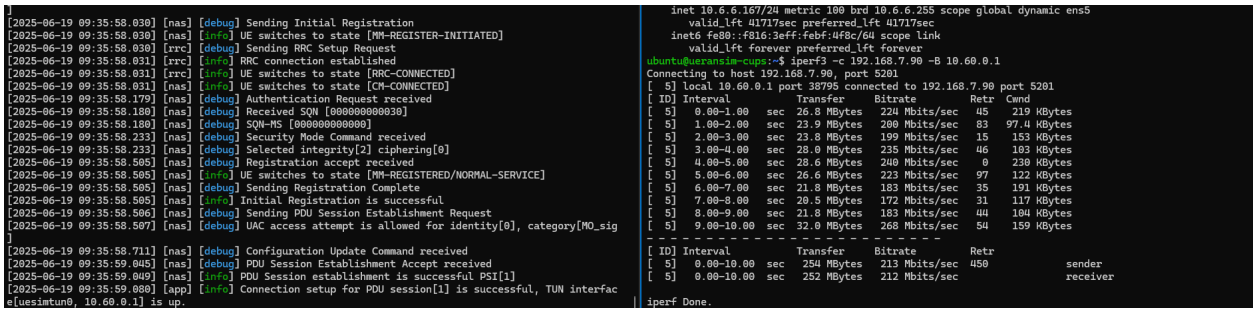

Figure 39: (a) UE registration and (b) traffic generation

Figure 40: Network throughput and CPU usage during a simple iperf connection

7.4.2. Interface Stress Testing

To evaluate the resilience of UPF under interface-level stress, we begin with the N3 interface, which handles GTP-U encapsulated user traffic from the gNB to the UPF. This interface is vital for user data delivery, and disruptions can directly impact the quality and continuity of service. In this experiment, we simulate a Denial of Service (DoS) attack on the N3 interface using the hping3 tool. This tool allows us to craft and send large numbers of packets with customizable parameters, enabling us to simulate different types of traffic attack. In this experiment we will send a rapid stream of TCP SYN packets to port 80 of the UPF's N3 IP address, overwhelming the target with half-open TCP connections. Although the UPF may not be running services on port 80, the flood aims to saturate the network stack and CPU with unsolicited traffic, simulating a real-world DoS scenario.

System behavior and user data forwarding are monitored in real time through Grafana, focusing on metrics such as CPU load, throughput, and packet loss. Figure 41 illustrates the impact of two SYN flood attacks on user traffic: the blue line representing throughput shows severe degradation and instability during the attack window. Figure 41 presents an iperf snapshot from the UE, confirming a sharp drop in data rate, approaching zero under load. Lastly, Figure 42 shows CPU usage before and during the attack, where a noticeable spike—almost a doubling—highlights the strain imposed on the UPF's processing resources.

Figure 41: Impact of two SYN attacks on the user data traffic

```
[  5]  32.00-33.00  sec  23.9 MBytes   200 Mbits/sec   28    121 KBytes
[  5]  33.00-34.00  sec  23.6 MBytes   198 Mbits/sec    8    132 KBytes
[  5]  34.00-35.00  sec  24.0 MBytes   201 Mbits/sec   36   94.8 KBytes
[  5]  35.00-36.00  sec  23.9 MBytes   200 Mbits/sec   11    126 KBytes
[  5]  36.00-37.00  sec  22.8 MBytes   191 Mbits/sec   53    136 KBytes
[  5]  37.00-38.00  sec  24.9 MBytes   209 Mbits/sec   18   98.7 KBytes
[  5]  38.00-39.00  sec  23.9 MBytes   200 Mbits/sec   22    132 KBytes
[  5]  39.00-40.00  sec  21.5 MBytes   180 Mbits/sec   77   47.4 KBytes
[  5]  40.00-41.00  sec  1.12 MBytes  9.44 Mbits/sec   67   3.95 KBytes
[  5]  41.00-42.00  sec  6.88 MBytes  57.7 Mbits/sec   25   93.5 KBytes
[  5]  42.00-43.02  sec  22.4 MBytes   185 Mbits/sec   93   43.4 KBytes
[  5]  43.02-44.00  sec  1.62 MBytes  13.8 Mbits/sec   51   35.5 KBytes
[  5]  44.00-45.00  sec  22.4 MBytes   188 Mbits/sec   61   77.7 KBytes
[  5]  45.00-46.00  sec  23.1 MBytes   194 Mbits/sec   16    133 KBytes
[  5]  46.00-47.00  sec  22.2 MBytes   187 Mbits/sec   23    137 KBytes
[  5]  47.00-48.00  sec  7.25 MBytes  60.8 Mbits/sec   97   1.32 KBytes
[  5]  48.00-49.00  sec  0.00 Bytes  0.00 bits/sec    1   1.32 KBytes
[  5]  49.00-50.00  sec  23.5 MBytes   197 Mbits/sec   25    549 KBytes
[  5]  50.00-51.00  sec  24.0 MBytes   201 Mbits/sec  302    255 KBytes
[  5]  51.00-52.00  sec  16.4 MBytes   137 Mbits/sec  165   86.9 KBytes
[  5]  52.00-53.00  sec  22.1 MBytes   186 Mbits/sec   49   90.8 KBytes
[  5]  53.00-54.00  sec  26.6 MBytes   223 Mbits/sec   61    165 KBytes
```

Figure 42: Snapshot from the UE performing an iperf connection during the attack.

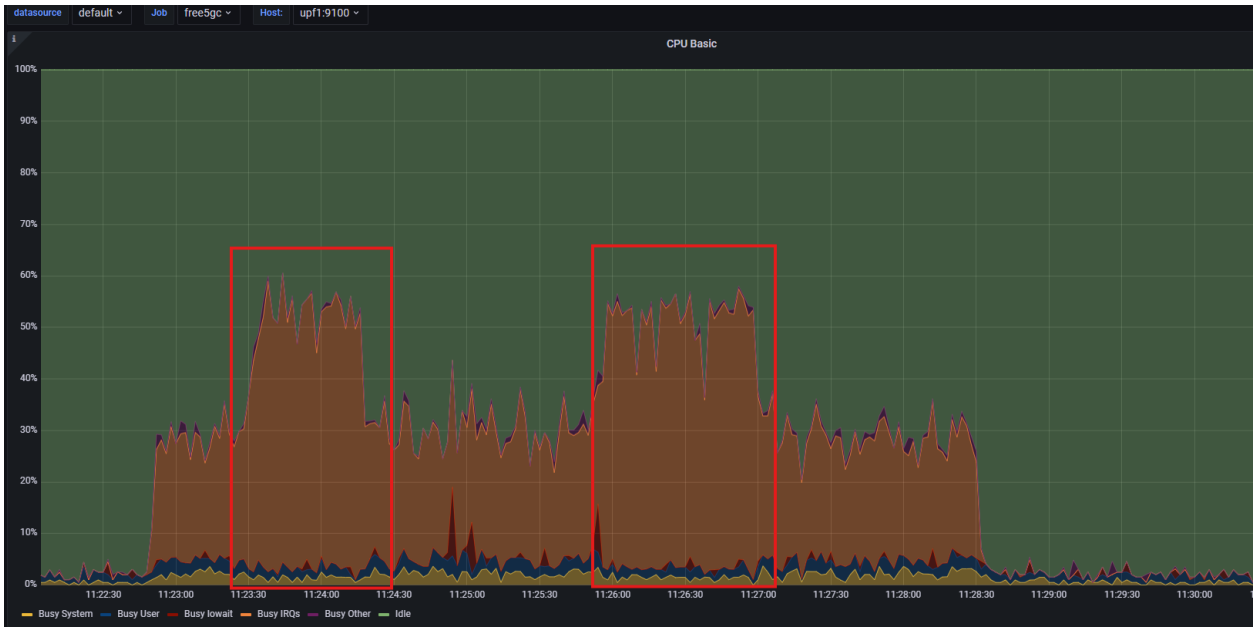

Figure 43: Impact on the CPU usage during the SYN attacks

We apply the same stress methodology to the N6 interface, which connects the UPF to the external data network, such as the internet or MEC platforms. Since this interface manages outbound traffic and enforces policies like NAT, QoS, and routing, it is a critical point for service delivery and a potential target for external attackers. By reusing the hping3 SYN flood attack, we emulate a situation where an adversary floods the N6 interface with TCP connection attempts, testing the UPF's ability to maintain its functionality under external pressure.

The impact of an attack performed on the N6 interface of the UPF node can be seen in Figure 44. As shown in the highlighted region of Figure 44 (left), the attack leads to a significant increase in CPU utilization, rising from 25% to 50%, representing a 25% escalation. Additionally, our attack results in anotable network throughput degradation, which decreasing by approximately 20% on average—from 250 Mbps to 200 Mbps.

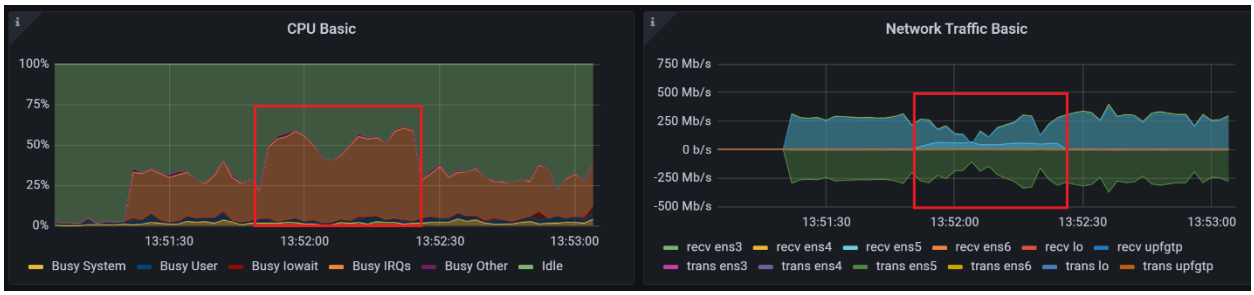

Figure 44: The impact on CPU and network resources of an attack performed on the N6 interface of the UPF node

7.4.3. Impact of an attack on UE registration

Finally, we examined the impact of a targeted attack on the N2 interface during the registration phase of the UEs. The N2 interface is used for control signaling between the gNB and the AMF, and its availability is essential for successful UE onboarding. In this experiment, we launched a SYN flood attack—again using hping3—on the N2 interface of the VM hosting the 5G Core. We then attempted to register 5 new UEs while the attack was in progress, in order to observe any degradation in control plane responsiveness or resource consumption.

Figure 45 (top) presents the baseline system performance during a standard UE registration, showing normal CPU load and network utilization. Figure 45(bottom) shows the system metrics when the registration occurs under attack. We observed a slight increase in registration time—approximately 2 seconds longer than

normal. Notably, the overall CPU busy time decreased, but the number of CPU interrupt requests (IRQs) increased significantly. This indicates that while the CPU was less occupied with higher-layer processing, it was burdened with handling large volumes of low-level interrupts caused by the flood of incoming SYN packets. These results demonstrate how even signaling-plane attacks can subtly degrade system responsiveness and consume processing resources in ways that may not be visible through traditional CPU usage metrics.

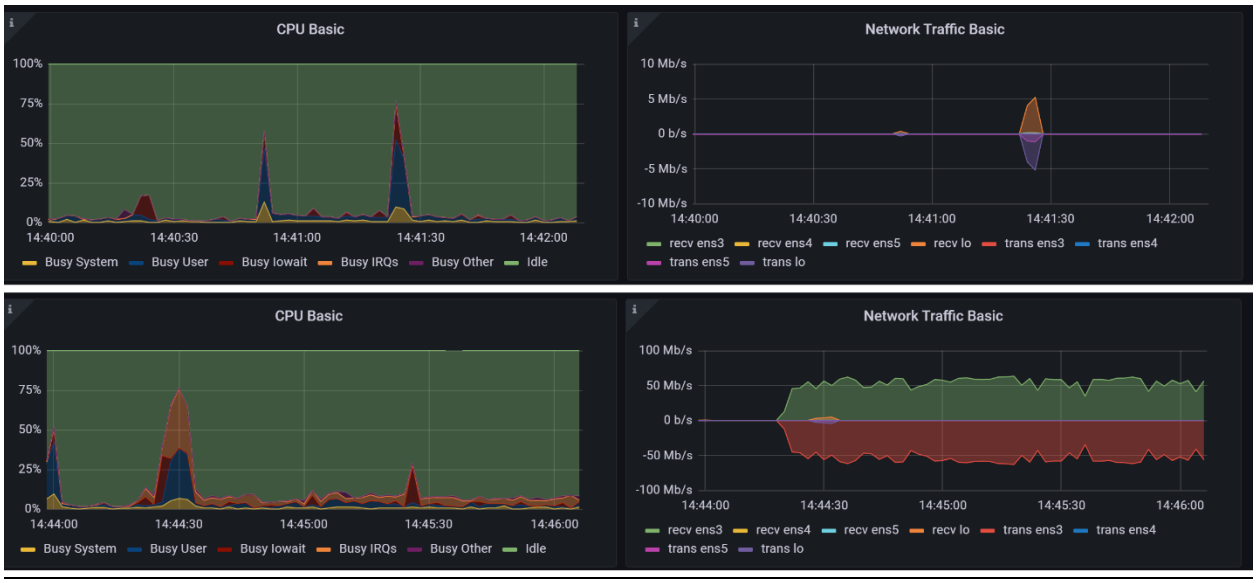

Figure 45: (top) Baseline system performance during a standard registration of 5 UEs (bottom) System performance during a registration of 5 UEs under an attack.

7.4.4. Mitigation Strategies

Advanced cybersecurity strategies must be tightly integrated into the network architecture to ensure robust and resilient service delivery [47] . Among the most promising approaches for dynamic network protection is the Moving Target Defense (MTD) [48] , which aims to increase uncertainty and complexity for potential attackers by continuously altering the system's attack surface. MTD-based mechanisms, particularly in cloud environments, focus on resource adaptation strategies that can act proactively or reactively to attacks. Existing MTD techniques generally fall into three main categories: network address shuffling, proxy-based approaches, and live migration strategies.

In this work we focus on VM migration-based MTD [49] , where critical services - like the UPF - are shifted to different VMs without interrupting ongoing user interactions (see Figure 46). Compared to network address shuffling and proxy-based approaches, VM live migration offers stronger protection by relocating the actual execution environment rather than simply obscuring network paths. This means that even if an attacker succeeds in identifying a target's IP address or proxy, the underlying service instance may have already been moved to a different physical or virtual host. This increases the attacker's cost, making it significantly more difficult to locate and compromise specific targets.

However, one question that arises regarding this technique is how often to migrate instances across sites [50] . On one hand, VMs hosting critical and NFs and entry points, must be migrated at regular intervals in order to efficiently avoid malicious acts. One the other hand, VM Migration requires large amounts of compute and network resources to be performed, and thus may have a huge impact to the underlying transport network, leading to congestion and degraded performance if not carefully planned. The challenge lies in balancing the security benefits of Live Migration MTD while not compromising network performance.

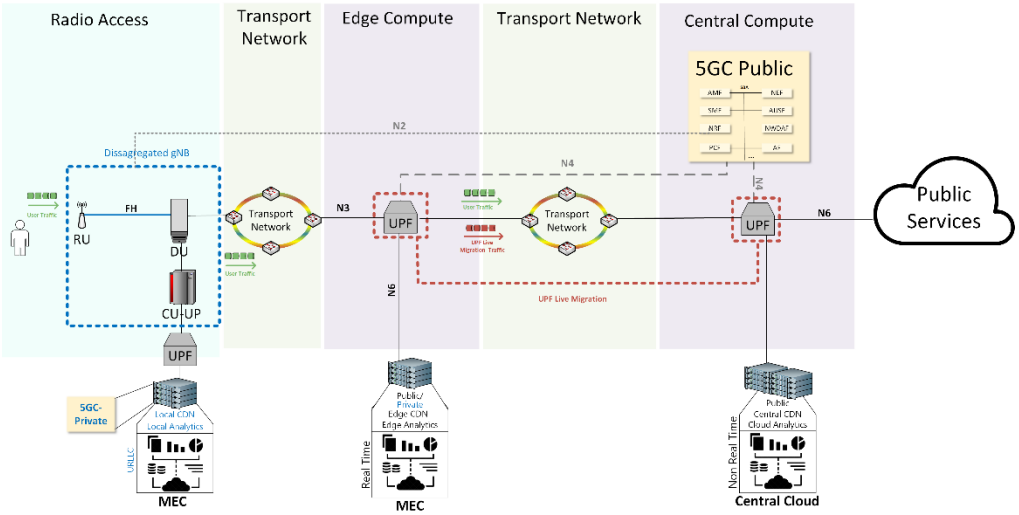

Figure 46: Network architecture with proposed VM migration-based MTD security mechanism

To this end, we have tried to identify an optimal trade-off between two competing objectives: minimizing the resource overhead and network strain introduced by frequent VM migrations, and maximizing the security gains by reducing the probability of a successful attack on the host OS, which could lead to the compromise or tampering of critical network functions such as the UPF.

7.4.5. Proposed Framework

The proposed framework is executed in the following steps. First, the desired network topology is initiated through OSM. Network traffic is generated from UEs through iperf connections and based on real user traffic statistics. In this setup, the Automated MTD (AMTD) method is applied, where active VMs hosting 5G VNFs are periodically relocated to different physical hosts. To ensure service continuity provided from the 5G VNFs, we exploit the feature of Live Virtual Machine Migration (LVMM) offered by openstack. To quantify the LVMM network impact on the transport network we perform block migration, which uses a pre-copy mechanism to initially transfer the Virtual Machines (VMs) storage, and then in a series of synchronization iterations, transfers the memory state of the host. After successful execution of a migration, the TN-induced overhead and migration duration can be retrieved from the monitoring platform.

7.4.6. Numerical Results and Discussion

We evaluated the impact of LVMM on network performance and the effect of a SYN flood attack on the UPF. To measure the impact of VM migration, we used the monitoring platform to capture the migration process, as illustrated in Figure 47. This allowed us to determine the volume of data transferred and the duration of each migration event. We also used the previous analysis to quantify the impact of a SYN flood attack on the physical node hosting the UPF VNF.

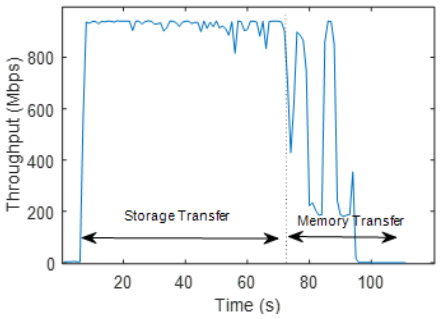

Figure 47: Live Migration of VM that hosts UPF.

The key objective of our approach was to identify a tradeoff between the frequency of migrations and network performance. In order to optimize the consumption of TN resources, we have developed a forecasting model based on LSTM neural networks, which is written in python using the Keras library. The forecasting model is executed as follows:

- Data Retrieval: TN traffic data are retrieved from the monitoring platform
- Data preprocessing: Involves the normalization of raw data and then splitting them into a training set and a test set
- Model Training: based on Keras library
- Prediction: The model predicts a future value related to the consumption of available resources

By leveraging the Long Short-Term Memory (LSTM) neural network model, we can predict forthcoming traffic patterns and only trigger migrations when the network load is below a defined threshold. This approach ensures that the security benefits of reducing attack exposure are achieved without negatively affecting the network's performance. Additionally, the prediction time horizon is aligned with the time required for migration, allowing sufficient lead time to optimize migration scheduling.

Furthermore, the threshold for migration takes into account the additional data volume introduced by the migration process. For example, if multiple VMs are migrated simultaneously, the migration impact on the network is larger, and the threshold adjusts accordingly to prevent congestion. The output of the LSTM forecasting model is shown in Figure 48.

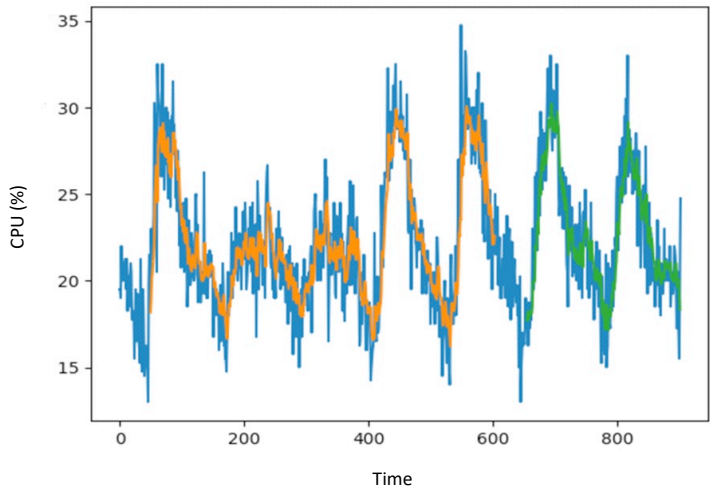

Figure 48: Forecasting model output. Blue: raw data, yellow data: training set, green data: test set

7.5. 5G CN functions vulnerability assessment with XDP

7.5.1. Introduction

The transition to a fully cloud-native, Service-Based Architecture (SBA) in 5G Standalone (SA) networks has transformed what was once a single EPC monolith into a distributed set of microservices interconnected by HTTP/2 APIs and high-speed GTP-U tunnels. While 3GPP Release 18 introduces security mechanisms such as mutual authentication and SUCI-based privacy for the signaling plane, it delegates responsibility for large-scale flood mitigation entirely to operators. As a result, the User Plane Function (UPF) and other SBA Network Functions remain vulnerable to off-the-shelf Layer 2/3/4 floods, as well as reflection and amplification attacks.

Recent data illustrates how significant this threat has become. In 2024, Cloudflare's autonomous mitigation systems blocked over 21.3 million DDoS attacks—an increase of 53% compared to the previous year—with an average rate of 4,870 mitigations per hour. During Q4 alone, more than 420 attacks exceeded 1 billion packets per second and 1 Tbps in volume, while >1 Tbps events surged nearly twenty-fold. A volumetric attack of that scale targeting the N3 or N6 interface could overwhelm Linux queuing mechanisms, exhaust CPU resources in SBA containers, and cause widespread subscriber detachment or even forced core restarts.

Conventional defences—such as virtual firewalls, VM-based deep packet inspection, or sidecar gateways—introduce tens of microseconds of latency and only take effect after traffic has traversed the kernel's network stack. This delay allows high-volume floods to consume valuable CPU resources, even if packets are ultimately dropped. In contrast, technologies like eBPF and the eXpress Data Path (XDP) enable small, verifier-safe programs to run inside the NIC driver itself. This allows for immediate packet filtering or redirection at line rate, with sub-microsecond latency, and without involving socket buffers.

However, current XDP deployments are typically based on static ACLs or basic heuristics, leaving them ill-equipped to detect rapidly evolving attack patterns that require dynamic, context-aware decisions. Meanwhile, advances in streaming analytics and lightweight machine learning models have made flow-based anomaly detection feasible in user space, with response latencies under a millisecond. What remains lacking is a seamless, end-to-end architecture that connects ML-driven classification with in-kernel enforcement and subscriber-aware feedback mechanisms.

Our proposed solution addresses this gap by deploying an XDP program at the UPF ingress. This program parses complete L2/L3/L4 tuples and uses a fixed-point decision-tree classifier to evaluate traffic directly in the fast path. Suspicious IPs are first subjected to a one-minute token-bucket rate limiter. If they trigger three violations within a five-minute sliding window, they are escalated to a five-minute hard block. Both enforcement actions are executed entirely within the NIC driver, at microsecond-scale latency.

Compact event summaries are passed to user space via a ring buffer. There, a control-plane daemon handles logging, visualizes state transitions in real time, writes full packet captures to disk, and populates live dashboards using data from pinned BPF maps. This daemon also maps throttled or blocked IPs back to subscriber IMSIs using UPF logs, enabling optional policy adjustments via SMF or AMF—without generating extra control-plane traffic. Enriched and deduplicated indicators are published to a local MISP instance when novel attack patterns, first-time rate limits, or escalations are observed.

By combining a high-speed, in-kernel ML-based enforcement layer with an intelligent, subscriber-aware user-space orchestrator, our framework delivers real-time, microsecond-scale protection aligned with the performance, programmability, and reliability requirements of modern 5G SA cores.

7.5.2. BACKGROUND AND RELATED WORK

7.5.2.1. Exposure Surface of a 5G SA Core

5G SA control-plane functions such as AMF, SMF, NRF, NEF and several others—communicate over the Service-Based Interface (SBI) using HTTP/2. On the user plane, the UPF bridges high-speed GTP-U traffic arriving on N3 to external IP networks using the N6 interface. Although 3GPP TS 33.501 mandates mutual authentication, integrity protection and encryption, it explicitly leaves volumetric DoS protection to the operator. In practice, the UPF is often the primary network function responsible for classifying incoming traffic in a 5G network—discarding malicious or non-compliant packets before they reach other core components. To achieve this, UPF elements can be integrated with eBPF and XDP allowing operators to classify traffic and detect anomalies directly within the kernel space, ensuring sub-microsecond response to user-plane threats

7.5.2.2. Extended Berkeley Packet Filter (eBPF) and the eXpress Data Path (XDP)

eBPF [51] (Extended Berkeley Packet Filter) is a powerful technology built into the Linux kernel that lets small, safe programs run in response to events like network packets, system calls, or performance traces. It was introduced in Linux 4.x and has become a key tool for networking, security, and observability.Developers write eBPF programs in a simplified version of C. These are compiled iand loaded into the kernel using the bpf() system call.

Before running, each program is checked by a verifier to make sure it:

- Won't crash the system (memory-safe)
- Won't run forever (no infinite loops)
- Won't use too much stack memory

Once verified, the kernel's Just In Time (JIT) compiler converts the bytecode into fast machine instructions, stored in executable memory. These programs can run at line rate, and can be updated live without restarting the system.

XDP (eXpress Data Path) is the fastest execution point for eBPF programs. It runs inside the network driver, right after the NIC transfers a packet via DMA, but before the Linux kernel allocates socket buffers (SKBs) or applies any queuing logic. This early position allows XDP programs to quickly pass, drop, or redirect packets with minimal overhead. On modern hardware, XDP can process massive traffic volumes. A single 3 GHz CPU core can handle over 25 million packets per second (Mpps). With Receive Side Scaling (RSS) and multiple dedicated cores, even standard servers can achieve 100–400 Gbps throughput purely in software.

High-speed processing offered eBPF and XDP [52] brings two major advantages to telecom environments:

- DDoS Mitigation at the Earliest Point: Large-scale floods can be blocked before they hit kernel memory or trigger user-space services, protecting the 5G core from resource exhaustion or cascading failures.
- Agile, Real-Time Policy Updates: Since eBPF programs can be loaded dynamically, operators can deploy new rules—like updated signatures or ML-based classifiers—in milliseconds without packet loss or service disruption. This enables continuous delivery of packet policies, something not possible with traditional kernel modules.

To take advantage of eBPF and XDP, 5G-TACTIC has developed packet monitoring and classification tools that offer the following functionalities:

(i) full-line-rate parsing and ML-based flow classification directly inside the XDP fast path with microsecond processing ovehead; allowing malicious traffic to be promptly discarded.

(ii) a user-space controller that consumes ring-buffer telemetry, feeds updated rules back through BPF maps and prints only state transitions to keep consoles quiet;

(iii) subscriber-aware correlation that maps offending IPs to IMSIs or slices without hammering the control plane; Through this approach IMSIs registered to UDR can be directly isolated from the mobile network

(iv) automatic cyberthreat information sharing

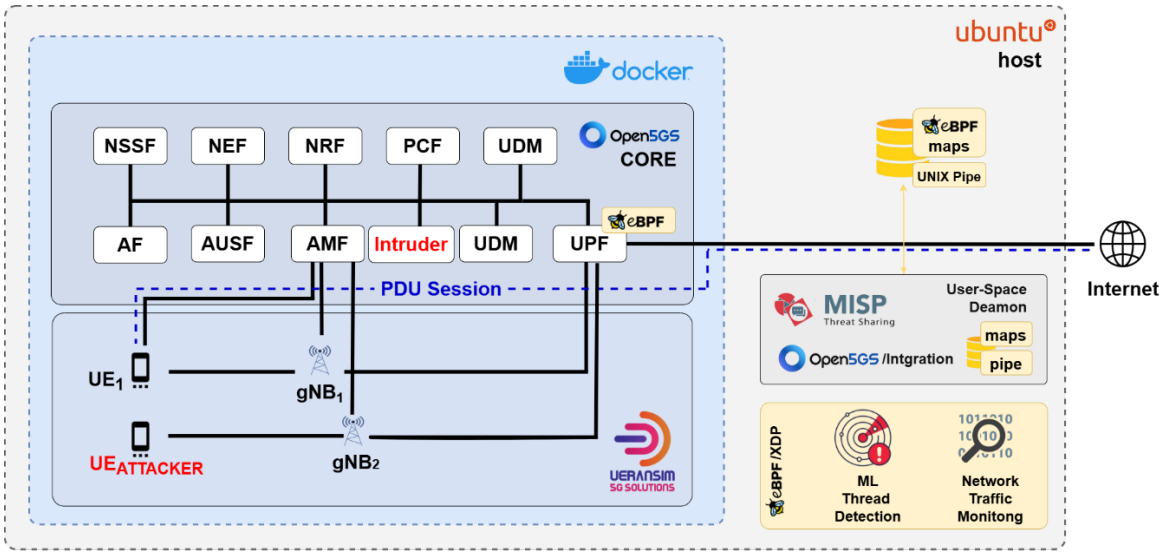

Figure 49. Malicious traffic detection with eBPF-based anomaly detection running in UPF nodes and exposure to CTI platforms..

7.5.3. SYSTEM ARCHITECTURE AND THREAT MODEL

Figure 49, depicts the proposed defense stack: an XDP monitor embedded in the UPF's NIC driver performs fast parsing, decision-tree classification, while a lightweight user-space controller ingests ring-buffer telemetry, correlates offending IPs with UPF session state, updates BPF maps, and streams enriched events to both a local MISP instance and the SMF/AMF for optional subscriber-level policies. Shared BPF maps form the low-latency control channel between kernel and user space, and pinned map persistence allows hot reloads without losing enforcement state.

7.5.3.1. User-space daemon

Running as an ordinary process outside the kernel, the proposed controller subscribes to the ring buffer emitted by the XDP layer, keeps lightweight per-IP caches, and pushes enforcement updates back through shared BPF maps. Each incoming traffic event first updates counters, then checks whether it signals a new state for the source address (benign → attack, limited → blocked, and so on). Only genuine transitions hit the console, preventing alarm fatigue, whereas every record is time-stamped to disk and forwarded as JSON over a named pipe for SIEMs. When an IP moves into its one-minute limit or five-minute block, the daemon can consult UPF session logs to look up the corresponding IMSI and, if configured, send a north-bound gRPC call to the AMF/SMF for subscriber-level action. It also builds deduplicated, context-rich MISP payloads, complete with attack label, severity, and remediation note, and ships them to the local CTI, falling back to a spool file if the API is unreachable to avoid data loss. Periodic scans of pinned maps drive live dashboards (top flows, active limits, block timers) and garbage collection; on exit, handlers detach the XDP link, close the FIFO, flush logs, and print a summary, making this component both the operations console and the orchestration brain of the defense chain.

7.5.3.2. XDP Monitor / Policy Enforcer

The in-kernel module attaches at the earliest ingress hook of the UPF NIC, preceding the IP stack, tc, so each frame is examined at line-rate inside the RX queue. It parses Ethernet and IPv4 headers, extracts the five-tuple, then consults two hash maps: one storing per-flow counters (packets, bytes, timestamps, flag tallies)

and one counting packets per source IP for hot-talker spotting. New flows enjoy a short grace (≤ 10 packets or 5 ms) to spare TCP handshakes; afterwards nine integer features (protocol, packet count, byte total, duration, mean size, bytes-per-second, SYN/ACK/RST counts) enter an in-lined depth-8 decision tree derived from offline training. With no loops or allocations, the tree returns one of twenty verdicts (Benign, 18 attack classes, SYN-flood) in a handful of cycles, and the outcome plus metadata are written to the ring buffer without a system call.

If the verdict is malicious, a two-tier response kicks in. Tier 1: a token-bucket limiter grants each bad IP 1 MB of tokens with a 10 kB µps refill; for 60 s packets spend tokens or are dropped, every tenth drop being let through for liveness. Tier 2: three such periods within a rolling five-minute window cause the IP to be stamped into Blocked IPs map, silencing it for five minutes. Both maps are updated and aged on every packet, so expired limits and blocks vanish automatically. Extra events (BANDWIDTH_LIMITED and BLOCKED) keep the daemon fully informed. Because all work precedes SKB allocation and Netfilter, per-packet latency stays in low microseconds at 100 Gbps. Maps are pre-sized, counters are atomic, and helper usage respects verifier limits, enabling unprivileged loading on modern LTS kernels. The tree is plain C code, so operators can retrain offline, re-compile, and hot-swap without touching policy logic; extra checks (TTL, QUIC heuristics, signatures) can be added as new leaves at negligible cost.

7.5.3.3. Threat Detection

The embedded tree is generated from the CIC-DDoS-2019 flow set, whose CSV affords labelled benign and 17 DDoS/DrDoS classes. A training script normalizes column names, confirms the nine required features, derives a fixed-point Flow Bytes/s, and under-samples so each class has equal rows, then shuffles to avoid order bias. The resulting balanced frame contains:

- Protocol Type
- Total Forward Packets
- Forward Packet Length Total
- Flow Duration
- Packet Length Mean
- Flow Bytes/s
- SYN Flag Count
- ACK Flag Count
- RST Flag Count

A scikit-learn Decision Tree Classifier is trained. Continuous fields are multiplied by 1000 so the final C uses only 32-bit integers. Depth-8 fits comfortably under verifier limits while retaining discrimination; lab runs show perfect in-sample precision/recall and sub-µs inference at 100 Gbps. The script then auto-emits C: each internal node becomes an `if (Feature <= Threshold)` on integers, and leaves return attack constants.

7.5.3.4. Threat-Intelligence Sharing

CTI is treated as an equal-priority outcome of defense. Whenever an IP is limited, blocked, or seen using a new tactic, the daemon forges a MISP event containing threat level, analysis stage, scope, IP

source/destination, port, protocol, packet size, taxonomy tag, and remediation text. Consistent tag names (`automated-response`, `IP-blocked`, etc.) let downstream playbooks filter by phase, while the tree's label appears in `Network activity` for easy pivots. If the HTTPS push fails, JSON is queued locally and retried, ensuring nothing is lost. To avoid flooding CTI peers, three in-memory caches discard duplicates: one-hour for blocked IPs, ten-minute for first-time limits, thirty-minute for per-IP & attack-type combinations, reducing duplicate objects by 95%. Outbound events intentionally omit IMSI or subscriber IDs; those stay internal for SMF hooks, though operators may extend the template (slice, RAT, region) when policy allows. Besides MISP, the exporter can output STIX 2.1 for TAXII 2 or push to SIEM/SOAR over REST or MQ.

7.5.3.5. Open5GS Integration

The daemon shares a host with the Open5GS UPF, granting read access to it append-only log. Each new suspicious event triggers a lookup in logs mapping GTP-U tunnel IPs to IMSIs and slices. Thus, after a limit or block, the daemon can issue a Policy Association Update via gRPC to the SMF's admin port, setting a throttled rate or inserting a redirect rule. Repeat offences may prompt a PFCP Session Deletion, forcing the UE to reattach without harming others on the same cell. If the SMF is down, requests queue with exponential back-off.

7.6. 5G core network functions vulnerability assessment with System Call Approach

7.6.1. Introduction

In 5G networks—where ultra-fast connectivity and extremely low latency are mandatory—intrusion-detection systems that rely only on network-traffic inspection run into major obstacles, particularly when hostile actions are artfully disguised as legitimate flows. Extended Berkeley Packet Filter (eBPF) offers a fresh option by tracking kernel-level events with negligible overhead, making it exceptionally well-suited to 5G's high-performance setting .

Most current Intrusion Detection Systems (IDS) solutions still zero in on traffic—packets, flow records, and payload exchanges—to pinpoint anomalies. Although that method exposes certain threats, it falls short against today's sophisticated adversaries. Attackers now routinely mask their moves inside what appears to be normal traffic, enabling them to slip past conventional detectors. Encrypted payloads further obscure what packet-based monitors can see. For that reason, traffic analysis alone no longer supplies the protection modern infrastructures demand; rising attack complexity calls for fresh perspectives. We therefore present an alternate strategy for safeguarding 5G environments. Rather than parsing packets, we observe system calls—the low-level requests programs issue to the operating system, such as opening files or creating sockets. These calls faithfully reflect a program's genuine behavior, providing a richer lens on host activity.

Our implementation employs eBPF to collect and stream system-call traces in real time while imposing only a minimal load on CPU cycles and memory footprints. By learning and evaluating call sequences, the system can rapidly decide whether operations match normal workload patterns or signal a brewing compromise or privilege-escalation attempt.

This technique brings several advantages:

- It stays effective even when attackers hide within benign-looking traffic streams.
- It delivers deep insight into internal process behavior that packets cannot reveal.
- It remains lightweight, conserving the tight resource budgets of 5G nodes.

Combining accuracy with efficiency, this pivot from traffic inspection to behavior-centric analysis strengthens 5G security and surfaces threats traditional network-focused IDS would likely overlook, helping operators maintain reliable and trustworthy services.

7.6.2. Methodology

Our intrusion-detection strategy gathers system-call activity through eBPF probes anchored in the kernel, allowing immediate recognition of abnormal behavior inside 5G environments by concentrating on host dynamics instead of packet signatures. To accomplish this, we deploy an eBPF program that continuously records the exact execution sequence of every system call.

To reproduce lifelike threat situations, we scripted 18 distinct attack categories with hping3. Each scenario ran for three separate intervals—5 min, 15 min, and 60 min. The traces produced during the 15- and 60-minute runs formed our training corpora, whereas the 5-minute traces served exclusively as the evaluation set. We trained and tested four learning engines—Long Short-Term Memory (LSTM), Dense Neural Network (DNN), Random Forests, and XGBoost—measuring accuracy, precision, recall, and F1-score so we could judge the system's ability to spot both known and novel attack patterns.

For the 5G sandbox we adopted the Open5GS framework orchestrated with Docker Compose, following the topology in Figure 49. All essential core components were deployed, together with a User Equipment (UE) acting as the adversary. This hostile UE, armed with hping3, reached the User Plane Function (UPF) over a dedicated virtual NIC. Through that interface it drove malicious traffic directly at the UPF, capitalising on the high-throughput data path. A second, legitimate UE generated baseline flows with iperf; once the assault began, its bandwidth collapsed to 0 bitps, illustrating how the attack crippled normal service. Because the UPF mediates traffic between the 5G core and external networks, the exercise showed how attackers can disrupt delivery, erode quality of service, and possibly expose user data—underscoring the need for strong defences at the UPF.

7.6.3. Data Collection and Analysis

System-call sequences—such as [systemcall_23, systemcall_4, systemcall_43, …]—were harvested with eBPF. Each trace was divided with multiple window and step configurations to pinpoint the combination that delivered the highest predictive accuracy, then enhanced with supplementary attributes so the models could separate ordinary from hostile behaviour.

Alongside host telemetry, we captured packets with tcpdump and wrote them to PCAP files. From these captures we distilled flow descriptors: flow duration, inter-arrival gaps, byte and packet totals, TCP-flag distributions, plus entropy-based metrics. Combining host-level and network-level attributes furnished a richer depiction of every episode.

During preprocessing we normalised values and computed descriptive statistics—frequency counts, transition probabilities, repetition tallies, and aggregate measures (mean, standard deviation, minimum, maximum, median). We also engineered higher-order n-gram counts (bigram, trigram, four-gram) to encode co-occurrence patterns inside the call stream.

Feature-correlation analysis exposed redundancy; strongly correlated columns were dropped, trimming complexity yet preserving predictive power. We then excluded attributes with very low mutual-information scores, limited variance, or excessive overlap. For XGBoost and Random Forest we retained the ten most informative variables; LSTM and DNN consumed the full feature set to exploit their representation capacity. Eliminating low-value attributes streamlined computation and reduced over-fitting. The discarded low-impact features—shown in Figure—illustrate their scant relevance to the prediction target.

7.6.4. Machine-Learning Models

After preprocessing, the refined sequences fed four classifiers suited to temporal and feature-rich data: XGBoost, Random Forest, LSTM, and DNN. Each model learned to label traces as either

- Normal – a system-call pattern emblematic of routine, benign operation, or
- Attack – a pattern signalling malicious intent.

Training relied on clearly annotated samples, enabling the supervised algorithms to internalise the distinctions between safe execution and security breaches. Their combined evaluation across accuracy, precision, recall, and F1-score demonstrates that host-centred monitoring with eBPF, coupled with carefully engineered features and diverse learners, can robustly uncover both overt and covert threats within high-speed 5G infrastructures.

7.6.5. Results

Our experiments were designed to measure how well system-call–sequence monitoring with eBPF uncovers sophisticated intrusions and to benchmark that method against conventional traffic-flow analysis. We gathered host-level traces as well as packet captures (tcpdump PCAPs) and distilled pertinent features from both. Three learning engines—XGBoost, Random Forest, and a deep-learning model—were trained and tested on data collected over 5-, 15-, and 60-minute windows. Performance was judged with Precision, Recall, F1-score, and Accuracy.

Overall, the eBPF-based, system-call approach surpassed traffic analysis when spotting stealthy, low-level compromises. Because system-call attributes expose fine-grained execution details, they delivered noticeably higher recall and F1. Packet-flow features still excelled at flagging high-volume anomalies but struggled to expose subtle behaviours that barely disturb aggregate traffic.

Performance in Network Traffic Analysis

Traffic-only results underscored those limitations. For the 15- and 60-minute dataset (3,062 and 11,882 rows, respectively), every algorithm underachieved:

- XGBoost showed low performance with Precision, Recall, F1, and Accuracy consistently around 0.15–0.17.
- Random Forest marginally improved upon these results, but its metrics still hovered between 0.16 and 0.17.
- Deep Learning exhibited the highest performance within this category, achieving Precision, Recall, and F1-scores of 0.17–0.20.

Performance in System-Call Sequence Analysis

Host-level inspection told a radically different story. Leveraging eBPF, every model—XGBoost, Random Forest, and the DNN—hit 0.99 for Precision, Recall, F1, and Accuracy on both the 15-minute (621,000 rows) and 60-minute (1,175,900 rows) sets. Figure 51 confusion matrix for XGBoost shows almost flawless separation. Figure 50, contrasting call distributions in normal versus attack states, explains why: hostile activity reshapes call frequencies and types so dramatically that patterns become unmistakable.

The stark performance gap proves that system-centric monitoring exposes behavioural shifts invisible to flow-centric tools, allowing pinpoint detection of advanced threats.

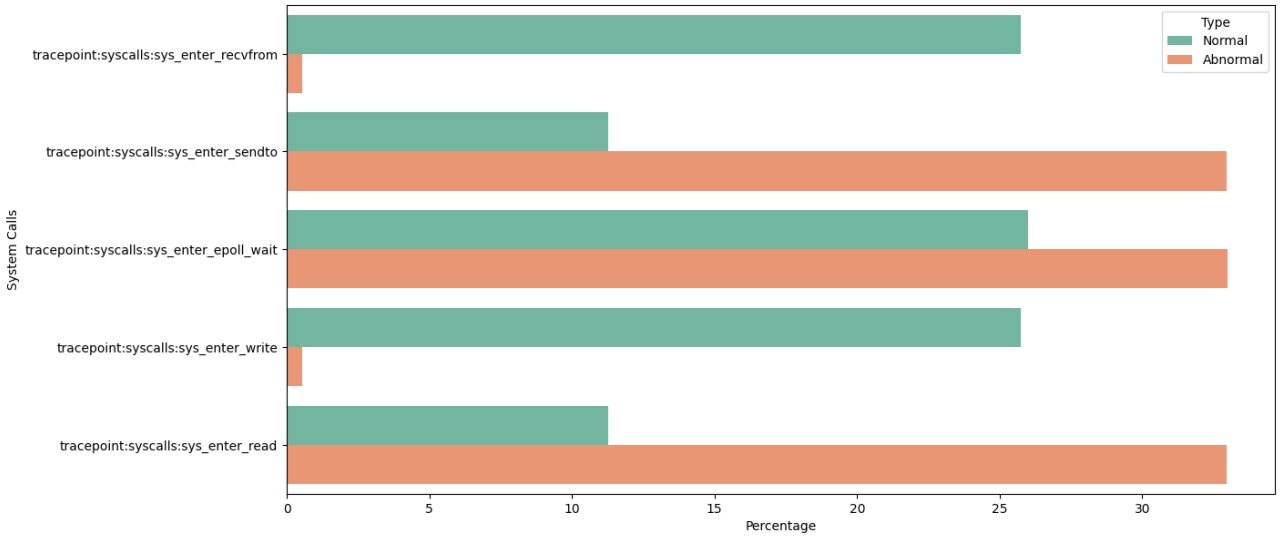

Figure 50. System call distribution.

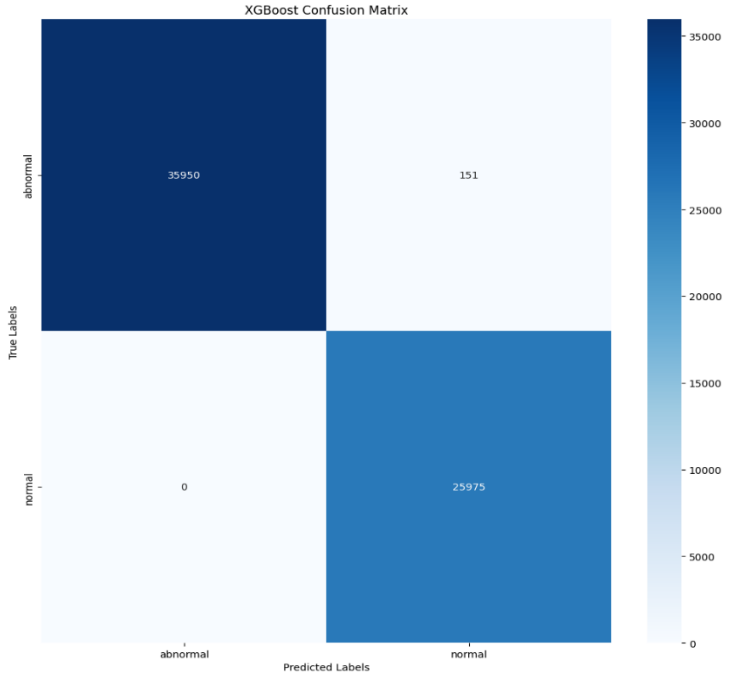

Figure 51. Confusion matrix.

Comparing both strategies confirms eBPF-driven IDS is markedly superior for advanced-threat hunting. Real-time call tracing yields deeper behavioural insight, whereas packet analysis suffered from low Precision, Recall, and F1. The eBPF method also scaled effortlessly to million-row datasets—vital for busy 5G cores. Key takeaways are:

- Effectiveness of eBPF Probes – System-call monitoring decisively outperformed traffic inspection, validating host-level visibility.
- Scalability – High throughput with no slowdown makes the approach ideal for 5G workloads.
- Sophisticated-Threat Coverage – Even SYN floods—often masked inside normal traffic—were reliably exposed.

- Algorithm Insights – While deep learning edged out others on packet data, all models achieved perfect scores on call data.

The IDS excelled when attacks were severe enough to drop every legitimate UE, achieving near-perfect metrics. In lighter assaults—only some UEs disconnected—accuracy slipped to 60–69 %, suggesting subtler disruptions are harder to tease apart from routine variance. Future work should boost sensitivity to low-intensity campaigns, perhaps by fusing additional host and network features or adopting richer anomaly-detection schemes, to guard against persistent, hard-to-spot threats.

8. Conclusions

End-to-end (E2E) security testing within the 5G-TACTIC project embodies a thorough and systematic approach to tackling the security challenges inherent in modern 5G deployments. By validating integrated security mechanisms across all network segments and ensuring compliance with EU regulatory frameworks and 3GPP standards, the project aims to establish a secure, resilient, and trustworthy 5G infrastructure capable of supporting the evolving needs of digital economies and societies.

In this context, 5G-TACTIC focused on the development of secure O-RAN solutions that addressed key challenges such as security, integration, interoperability, and disaggregation. By enabling the integration of multi-vendor Open 5G RAN platforms into commercial environments, the project promoted advancements in the EU supply chain and supported the strategic goal of reducing reliance on single or limited vendors. This approach helped MNOs avoid vendor lock-in, thereby increasing flexibility and resilience in network deployments.

To achieve these objectives, the deliverable used the SCAS/SECAM outlined in Deliverable D3.1 to analyze, test, and assess cybersecurity aspects of Open RAN solutions. These solutions were provided by ACC and BBRAN and included open-source implementations such as OpenAirInterface (OAI) for the RAN and core network components like OAI, free5GC, and Open5GS. The assessment work was divided into two main parts: Part I focused on the RAN, while Part II addressed the Core Network (CN).

In the RAN segment, the initial testing aimed to identify vulnerabilities and validate interoperable operations across multi-vendor RAN and CN functions. This involved deploying a combination of commercial and open-source platforms to evaluate the performance of service slices across heterogeneous network environments. Particular attention was given to the availability and integrity of gNB nodes under varying traffic loads and their behavior in response to unexpected or malformed data inputs. Furthermore, the security of control plane data was analyzed, with emphasis on the O-RAN E2 protocol that exposed interfaces to Near RT-RIC and Non-RT-RIC controllers. Additional evaluations targeted vulnerabilities in the authentication and authorization mechanisms of the gNB, potential weaknesses in the N2 interface, unauthorized access to RIC controllers, and improper management of authentication policies.

To protect RAN components from physical-layer threats such as jamming attacks, spectrum monitoring mechanisms were developed and combined with specialized defense applications running in RIC controllers. Compute domain vulnerabilities—including those associated with virtualization, web services, and hardware—were addressed through the deployment of DPI systems that monitored RAN function behavior in real time.

For the core network, the SECAM methodology was applied to identify vulnerabilities within various functional elements. In particular, AMF was evaluated for issues in the implementation of authentication and key agreement procedures. These assessments focused on the N2 interface, which operated over the SCTP protocol and connected the gNB with the AMF. Vulnerability assessments were performed on all three targeted 5G core implementations—Free5GC, Open5GS, and OpenAirInterface—and all exhibited consistent behavior in terms of network response, message exchange patterns, and susceptibility to attack.

UPF was evaluated for vulnerabilities in user data and signaling protocols. An experimental framework was developed to simulate Do) and SYN flood attacks against UPF interfaces, enabling the evaluation of system resilience and performance under stress. Other core functions, such as the NEF and NRF, were monitored using DPI techniques that continuously analyzed data flows within service-based interfaces. When anomalies were detected, AI-driven models assessed the risk and applied mitigation strategies, such as removing suspicious users from the UDR.

Finally, the project emphasized the importance of CTI by implementing systems capable of collecting, storing, distributing, and sharing cybersecurity indicators and incident information across both RAN and CN components. These efforts supported a proactive security posture, enabling faster detection and response to emerging threats across the 5G-TACTIC ecosystem.

9. References

[1] Shatnawi, M., Altaleb, H., & Zoltán, R. (2022). The Digital Revolution with NESAS Assessment and Evaluation. 2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC), 000099-000104. https://www.semanticscholar.org/paper/The-Digital-Revolution-with-NESAS-Assessment-and-Shatnawi-Altaleb/b057c116b4d0f072adeb9e611ae78d7f4f016b44

[2] Mongay Batalla, J., de la Cruz Llopis, L.J., Gómez, G.P., Andrukiewicz, E., Krawiec, P., Mavromoustakis, C.X., & Song, H.H. (2024). Multi-Layer Security Assurance of the 5G Automotive System Based on Multi-Criteria Decision Making. IEEE Transactions on Intelligent Transportation Systems, 25, 3496-3512. https://www.semanticscholar.org/paper/Multi-Layer-Security-Assurance-of-the-5G-Automotive-Batalla-Llopis/41b29af5d350623a6f996eb260a3c528175d84cd

[3] Yan, Z., Zhang, P., & Vasilakos, A.V. (2016). A security and trust framework for virtualized networks and software-defined networking. Secur. Commun. Networks, 9, 3059-3069. https://www.semanticscholar.org/paper/A-security-and-trust-framework-for-virtualized-and-Yan-Zhang/f6ed8a9a8f8bee47b3974bb86e0ae8fc03fd8343

[4] https://www.ericsson.com/en/blog/2024/2/5g-sba-security-release-16-updates

[5] https://go.abiresearch.com/hubfs/Marketing/Whitepapers/Privacy%20and%20Data%20Security%20in%205G%20Networks/ABI_Research_Privacy_and_Data_Security_in_5G_Networks.pdf

[6] https://www.nokia.com/sites/default/files/2021-05/Whitepaper-5G-security-Nokia-STC-March-31-2021.pdf

[7] Wang, Y., Xu, J., & Jiang, L. (2014). Challenges of System-Level Simulations and Performance Evaluation for 5G Wireless Networks. IEEE Access, 2, 1553-1561. https://www.semanticscholar.org/paper/Challenges-of-System-Level-Simulations-and-for-5G-Wang-Xu/1b19c34439c8d179c58c3d96a1e857b72f84ba3e

[8] https://www.nokia.com/sites/default/files/2021-05/Whitepaper-5G-security-Nokia-STC-March-31-2021.pdf

[9] https://www.standict.eu/sites/default/files/2021-02/ENISA%205G%20Threat%20Landscape%20-%20Update.pdf

[10] https://www.spirent.com/blogs/testing-times-automation-is-the-key-to-a-secure-5g-future

[11] Estévez, D.M., Bulakci, Ö., Ericson, M., Prasad, A., Pateromichelakis, E., Belschner, J., Arnold, P., & Calochira, G. (2017). RAN enablers for 5G radio resource management. 2017 IEEE Conference on Standards for Communications and Networking (CSCN), 1-6. https://www.semanticscholar.org/paper/RAN-enablers-for-5G-radio-resource-management-Est%C3%A9vez-Bulakci/a0a69aa96a9c8691c199fa6f45fdb60f88a43180

[12] Deliverable D2.3. 5G-TACTIC End-to-End Security Architecture

[13] 3GPP TS 38.413. NG Application Protocol (NGAP). Version 18.5.0. Release 18.

[14] R. Stewart et al., "Security Attacks Found Against the Stream Control Transmission Protocol (SCTP) and Current Countermeasures", RFC 5062, September 2007, doi 10.17487/RFC5062.

[15] 3GPP TS 29.244. Interfaces between the Control Plane and User Plane Nodes. Version 18.9.0. Release 18.

[16] 3GPP TS 29.281. Universal Mobile Telecommunications System (UMTS). Version 18.3.0. Release 18.

[17] M. Barbosa, M. Silva, E. Cavalcanti, and K. Dias, “Open-Source 5G Core Platforms: A Low-Cost Solution and Performance Evaluation,” arXiv:2412.21162 [cs.NI], Dec. 2024. [Online]. Available: https://arxiv.org/html/2412.21162v1

[18] J. Wickboldt et al., “Analysis of Open Source 5G Core Deployments for Network Automation and Management,” in Proc. IEEE/IFIP NOMS 2023, 2023. [Online]. Available: https://www.inf.ufrgs.br/~jwickboldt/wp-content/uploads/NOMS_2023-Submitted.pdf

[19] Open5GS Project, "Open5GS IPsec Tunnel Deployment Guide," 2023. [Online]. Available: https://open5gs.org/open5gs/docs/guide/06-ipsec/

[20] M. Vijayalakshmi, “Deployment and Evaluation of 5G Core on Kubernetes,” Technical Report, Indian Institute of Technology Bombay, 2025. [Online]. Available: https://www.cse.iitb.ac.in/~mythili/research/papers/2025-5gcore-k8s.pdf

[21] S. Hosseinishamoushaki, “Cloud-native 5G Core Network: Deployment and Performance Analysis,” Master’s Thesis, Università degli Studi di Padova, 2023. [Online]. Available: https://thesis.unipd.it/retrieve/9c877408-954d-488d-ab86-4e8bdfb4131c/Hosseinishamoushaki_Seyedali.pdf

[22] Free5GC, “Free5GC Virtual Machine Configuration Guide,” Free5GC.org, 2025. [Online]. Available: https://free5gc.org/guide/2-config-vm-en/. Accessed: Jun. 25, 2025.

[23] Free5GC, “Free5GC Blog: Latest Updates,” Free5GC.org, Apr. 16, 2025. [Online]. Available: https://free5gc.org/blog/20250416/20250416/. Accessed: Jun. 25, 2025.

[24] E. Ernie, “Step-by-Step Guide to Configuring a 5G SA Network with Open5GS and MEC on Virtual Machines,” GitHub Pages, Mar. 2, 2023. [Online]. Available: http://ernie55ernie.github.io/python/2023/03/02/step-by-step-guide-to-configuring-a-5g-sa-network-with-open5gs-and-mec-on-virtual-machines.html. Accessed: Jun. 25, 2025.

[25] Open5GS, “Open5GS Documentation: CentOS Platform Guide,” Open5GS.org, 2025. [Online]. Available: https://open5gs.org/open5gs/docs/platform/02-centos/. Accessed: Jun. 25, 2025

[26] OpenAirInterface, “OpenAirInterface 5G Core Network Lab Guide,” OpenAirInterface.org, Dec. 2021. [Online]. Available: https://openairinterface.org/wp-content/uploads/2021/12/Workshop-CN-Lab-1.pdf

[27] G. García, “OpenAirInterface5G Cloud Core Deployment,” GitHub Repository, 2023. [Online]. Available: https://github.com/GinesGarcia/OpenAirInterface5G_CloudCore

[28] Free5GC, "free5GC compose documentation," Free5GC.org, 2025. [Online]. Available: https://free5gc.org/guide/0-compose/. Accessed: Jun. 25, 2025.

[29] Chen et al., "free5gc-compose: Docker Compose deployment for Free5GC," GitHub Repository, 2025. [Online]. Available: https://github.com/free5gc/free5gc-compose. Accessed: Jun. 25, 2025.

[30] J. User, "Free5GC-Compose network configuration," Free5GC Forum, 2023. [Online]. Available: https://forum.free5gc.org/t/free5gc-compose-network-configuration/1132. Accessed: Jun. 25, 2025.

[31] Borjis131, "docker-open5gs: Open5GS Docker deployment," GitHub Repository, 2023. [Online]. Available: https://github.com/Borjis131/docker-open5gs. Accessed: Jun. 25, 2025.

[32] H. Supreeth, "docker_open5gs: Alternative Open5GS Docker implementation," GitHub Repository, 2023. [Online]. Available: https://github.com/herlesupreeth/docker_open5gs. Accessed: Jun. 25, 2025.

[33] 5G-Prototype Team, "OAI 5G Core Network Deployment," 5G-Prototype Documentation, 2023. [Online]. Available: https://5g-prototype.readthedocs.io/en/latest/module_5_OAI_5G_technologies/wikipages/wikipage_3.html. Accessed: Jun. 25, 2025.

[34] ORBIT Lab, "OAI Docker Deployment Tutorial," ORBIT-Wiki, 2023. [Online]. Available: https://www.orbit-lab.org/wiki/Tutorials/oai-docker. Accessed: Jun. 25, 2025.

[35] 19. N. Shrestha, "free5gc-k8s: Kubernetes deployment for Free5GC," GitHub Repository, 2023. [Online]. Available: https://github.com/niloysh/free5gc-k8s. Accessed: Jun. 25, 2025.

[36] Nephio Project, "Exercise 1: Deploying free5GC," Nephio R3 Documentation, 2023. [Online]. Available: https://r3.docs.nephio.org/docs/guides/user-guides/exercise-1-free5gc/. Accessed: Jun. 25, 2025.

[37] H. Tuan, "Open5gs with UERANSIM Deployment Guide," HackMD, 2023. [Online]. Available: https://hackmd.io/@haidinhtuan/ryRuKdEI3. Accessed: Jun. 25, 2025.

[38] Datree.io, "open5gs-cgiraldo Helm Chart," Datree Helm Registry, 2023. [Online]. Available: https://www.datree.io/helm-chart/open5gs-cgiraldo. Accessed: Jun. 25, 2025.

[39] N. Shrestha, "open5gs-k8s: Kubernetes deployment for Open5GS," GitHub Repository, 2023. [Online]. Available: https://github.com/niloysh/open5gs-k8s. Accessed: Jun. 25, 2025.

[40] Gradiant, "open5gs-oaignb Helm Chart Documentation," 5G-Charts, 2023. [Online]. Available: https://gradiant.github.io/5g-charts/open5gs-oaignb.html. Accessed: Jun. 25, 2025.

[41] AIDY-F2N Team, "OAI-UERANSIM Integration," GitHub Repository, 2023. [Online]. Available: https://github.com/AIDY-F2N/OAI-UERANSIM. Accessed: Jun. 25, 2025.

[42] Giambartolomei, Filippo, et al. "Penetration testing of 5g core network web technologies." ICC 2024-IEEE International Conference on Communications. IEEE, 2024.

[43] Pittman, Jason M. "A Comparative Analysis of Port Scanning Tool Efficacy." arXiv preprint arXiv:2303.11282 (2023).

[44] Durumeric, Zakir, Eric Wustrow, and J. Alex Halderman. "ZMap: Fast internet-wide scanning and its security applications." 22nd USENIX Security Symposium (USENIX Security 13). 2013.

[45] Amazon Web Services, "5G Network Evolution with AWS," Whitepaper, 2023. [Online]. Available: https://d1.awsstatic.com/whitepapers/5g-network-evolution-with-aws.pdf

[46] UERANSIM Project, "UERANSIM Documentation: Integration with IPsec," 2023. [Online]. Available: https://github.com/aligungr/UERANSIM/wiki

[47] Scalise, P.; Boeding, M.; Hempel, M.; Sharif, H.; Delloiacovo, J.; Reed, J. A Systematic Survey on 5G and 6G Security Considerations, Challenges, Trends, and Research Areas. Future Internet 2024, 16, 67. https://doi.org/10.3390/fi16030067

[48] Nguyen, Minh, Debroy, Saptarshi, Moving Target Defense-Based Denial-of-Service Mitigation in Cloud Environments: A Survey, Security and Communication Networks, 2022, 2223050, 24 pages, 2022.

[49] M. Torquato, P. Maciel, and M. Vieira, "Evaluation of time-based virtual machine migration as moving target defense against host-based attacks," J. Syst. Softw., vol. 219, p. 112222, 2025, doi: 10.1016/j.jss.2024.112222.

[50] S. Alhozaimy and D. A. Menascé, "A formal analysis of performance-security tradeoffs under frequent task reconfigurations," Future Gener. Comput. Syst., vol. 127, pp. 252–262, 2022, doi: 10.1016/j.future.2021.09.005.

[51] M. A. M. Vieira, M. S. Castanho, R. D. G. Pacífico, E. R. S. Santos, E. P. M. C. Júnior, and L. F. M. Vieira, “Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications,” vol. 53, no. 1, pp. 1–36, doi: 10.1145/3371038.

[52] U. K. Dayalan, Z. Wu, G. Gautam, F. Tian and Z. -L. Zhang, "Towards an eBPF+XDP Based Framework for Open, Programmable and Scalable NextG RANs," 2023 IEEE Future Networks World Forum (FNWF), Baltimore, MD, USA, 2023, pp. 1-6, doi: 10.1109/FNWF58287.2023.10520475.

[53] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, “MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform,” in Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security (WISCS '16). Association for Computing Machinery, New York, NY, USA, 49–56., doi: 10.1145/2994539.2994542