O-RAN.WG11.SecTestSpecs.0-R004-v09.00

Technical Specification

O-RAN Work Group 11 (Security Work Group)

O-RAN Security Test Specifications

O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany

Register of Associations, Bonn VR 11238, VAT ID DE321720189

Contents

List of figures

List of tables

Foreword

This Technical Specification (TS) has been produced by O-RAN Alliance.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the O-RAN Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in O-RAN deliverables except when used in direct citation.

1 Scope

The present document provides description of the Security Tests, which validate security functions and configurations per security and security protocols requirements and are based on the priority of the risk analysis for O-RAN systems.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long-term validity.

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies. In the case of a reference to a 3GPP document, a non-specific reference implicitly refers to the latest version of that document in Release 18.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, O-RAN cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] O-RAN ALLIANCE TS: "O-RAN Architecture Description"

[2] O-RAN ALLIANCE TS: "O-RAN Security Protocols Specifications"

[3] void

[4] O-RAN ALLIANCE TS: "O-RAN End-to-End Test Specification"

[5] O-RAN ALLIANCE TS: "O-RAN Security Requirements and Controls Specifications"

[6] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications"

[7] 3GPP TS 33.117: "Catalogue of General Security Assurance Requirements"

[8] 3GPP TS 33.511: "Security Assurance Specification (SCAS) for the next generation Node B (gNodeB) network product class"

[9] 3GPP TS 33.216: "Security Assurance Specification (SCAS) for the Evolved Node B (eNodeB) network product class"

[10] Openssh Security Vulnerabilities, https://www.cvedetails.com/vulnerability-list/vendor_id-97/product_id-585/Openbsd-Openssh.html

[11] "IEEE Standard for Local and Metropolitan Area Networks--Port-Based Network Access Control," in IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018), vol., no., pp.1-289, 28 Feb. 2020, doi: 10.1109/IEEESTD.2020.9018454

[12] "Generating Software Bills of Materials (SBOMs) with SPDX at Microsoft", https://devblogs.microsoft.com/engineering-at-microsoft/generating-software-bills-of-materials-sboms-with-spdx-at-microsoft/

[13] NTIA: "The Minimum Elements For a Software Bill of Materials (SBOM)", https://www.ntia.gov/sites/default/files/publications/sbom_minimum_elements_report_0.pdf

[14] IETF RFC 8341: "Network Configuration Access Control Model"

[15] O-RAN ALLIANCE TR: "O-RAN O-Cloud Security Analysis Report"

[16] O-RAN ALLIANCE TR: "O-RAN Near-RT RIC and xApp Security Analysis Report"

[17] O-RAN ALLIANCE TR: "O-RAN Non-RT RIC Security Analysis Report"

[18] IETF RFC 6749: "The OAuth 2.0 Authorization framework"

[19] IETF RFC 7519: "JSON Web Token (JWT)"

[20] IETF RFC 7515: "JSON Web Signature (JWS)"

[21] O-RAN ALLIANCE TS: "O-RAN WG4 Management Plane Specification"

[22] O-RAN ALLIANCE TS: "O-RAN Use Cases Detailed Specification"

[23] 3GPP TS 33.523: "5G Security Assurance Specification (SCAS); Split gNB product classes"

[24] void

[25] 3GPP TS 33.501: "Security architecture and procedures for 5G system"

[26] O-RAN ALLIANCE TS: "O-RAN WG4 Control, User and Synchronization Plane Specification"

[27] O-RAN ALLIANCE TS: "O-RAN Near-RT RIC Architecture"

[28] O-RAN ALLIANCE TS: "O-RAN E2 Application Protocol (E2AP)"

[29] O-RAN ALLIANCE TS: "O-RAN E2 Interface Test Specification"

[30] IETF RFC 9562: "Universally Unique IDentifiers (UUIDs)"

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies. In the case of a reference to a 3GPP document, a non-specific reference implicitly refers to the latest version of that document in Release 18.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i.1] Service Name and Transport Protocol Port Number Registry, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[i.2] SPDX, https://spdx.dev/

[i.3] CycloneDX, https://cyclonedx.org/

[i.4] NIST IR 8060: "Guidelines for the Creation of Interoperable Software Identification (SWID) Tags", http://dx.doi.org/10.6028/NIST.IR.8060

[i.5] O-RAN ALLIANCE TR: "O-RAN Security Threat Modeling and Risk Assessment"

3 Definition of terms, symbols and abbreviations

3.1 Terms

For definitions of the security terms Attack, Attack surface, Risk, Security control, Technical control, Threat, and Vulnerability used in this document, refer to [i.5].

For the purposes of the present document, the terms and definitions given in 3GPP TR21.905 [6], O-RAN Architecture Description [1], and the following in this clause apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR21.905 [6] and O-RAN Architecture Description [1].

A1: Interface between non-RT RIC and Near-RT RIC to enable policy-driven guidance of Near-RT RIC applications/functions, and support AI/ML workflow.

E2: Interface connecting the Near-RT RIC and one or more O-CU-CPs, one or more O-CU-UPs, and one or more O-DUs.

RAN: Generally referred as Radio Access Network. In terms of this document, any component below Near-RT RIC per O-RAN architecture, including O-CU/O-DU/O-RU.
Overcommitting resources: It refers to the practice of allocating or promising more resources than are physically available on a system. This concept is commonly used in virtualized and cloud environments. The idea behind overcommitment is to optimize resource utilization based on the observation that not all applications will use their allocated resources to the maximum at the same time. Here's a breakdown of overcommitment for different resources:

- CPU Overcommitment: More virtual CPUs (vCPUs) are allocated to VMs or Containers than there are physical CPU cores available on the host.
- Memory Overcommitment: The total memory allocated to VMs or Containers exceeds the physical RAM available on the host.
- Storage Overcommitment: More storage space is allocated to VMs or Containers than the actual available capacity on the storage device.
- Network Overcommitment: More bandwidth is promised to VMs or Containers than the physical network can provide.

Overcommit ratios: It defines the extent to which resources can be overallocated compared to the actual available physical resources.

- CPU Overcommit Ratio: If you have a CPU overcommit ratio of 2:1, it means you can allocate twice the number of virtual CPUs (vCPUs) as there are physical CPU cores on the host. For instance, if a server has 8 physical CPU cores, you could allocate 16 vCPUs across various VMs or Containers.
- Memory Overcommit Ratio: If you have a memory overcommit ratio of 1.5:1, it means you can allocate 1.5 times the amount of virtual RAM as there is physical RAM on the host. For a server with 64GB of physical RAM, you could allocate a total of 96GB of RAM across various VMs or Containers.

3.2 Symbols

Void

3.3 Abbreviations

For the purposes of the present document, the abbreviations given 3GPP TR21.905[6], O-RAN Architecture Description[1], and the following in this clause apply. A abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR21.905[6] and O-RAN Architecture Description[1].

AI/ML Artificial Intelligence / Machine Learning

CMS/PKCS#7/CAdES Cryptographic Message Syntax/Public-Key Cryptography Standards/CMS Advanced Electronic Signatures

CNF Cloud Native Function

COT Chain of Trust

CSI Channel State Information

CSR Certificate Signing Request

DTLS Datagram Transport Layer Security

DUT Device Under Test

eCPRI Enhanced Common Public Radio Interface

FTP File Transfer Protocol

FTPS File Transfer Protocol Secure

gRPC a cross-platform remote procedure call (RPC) framework

IPSEC Internet Protocol Security

JSF JSON Signature Format

JSON JavaScript Object Notation

JWT JSON Web Token

JWS JSON Web Signature

KPI Key Performance Indicator

mTLS Mutual Trasport Layer Security

NACM Network Configuration Access Control Model

NETCONF Network Configuration Protocol

NFO Network Function Orchestration

NTIA National Telecommunications and Information Administration - United States Department of Commerce

OAuth Open Authentication

PDCP Packet Data Convergence Protocol

PNF Physical Network Function

PTP Precision Timing Protocol

RBAC Role-based Access Control

REST Representational state transfer

RDF Resource Description Format

RIC O-RAN RAN Intelligent Controller

RoT Root of Trust

SBOM Software Bill of Materials

SDLC Software Development Lifecycle

SSH Secure Shell

SUT System Under Test

TLS Transport Layer Security

VNF Virtualized Network Function

WAS Web Application Security

XML Extensible Markup Language

YAML YAML Ain’t Markup Language

4 Objectives and scope

This security test specification is focused on:

- Validating the implementation of security requirements and security protocols specified in [5] and [2].
- Emulating security attacks against the O-RAN component(s), interfaces, and the system to measure the robustness of the O-RAN system and the service impact(s).
- Validating the effectiveness of the security mitigation method(s) to protect the O-RAN system and the services it offers.

This security test specification is based on the priority of the risk assessment of the O-RAN security threats and security requirements of the O-RAN system.

5 Testing methodology and configuration

This clause describes the common testing methods and configurations used in the subsequent clauses. To ensure fair and comparable test results among various test campaigns, consistent test setups shall be utilized. This security test specification describes the test conditions, methodologies, and procedures, so that the test can be reproduced if needed, and the test results can be used for comparison or reference purposes.

5.1 DUT / SUT

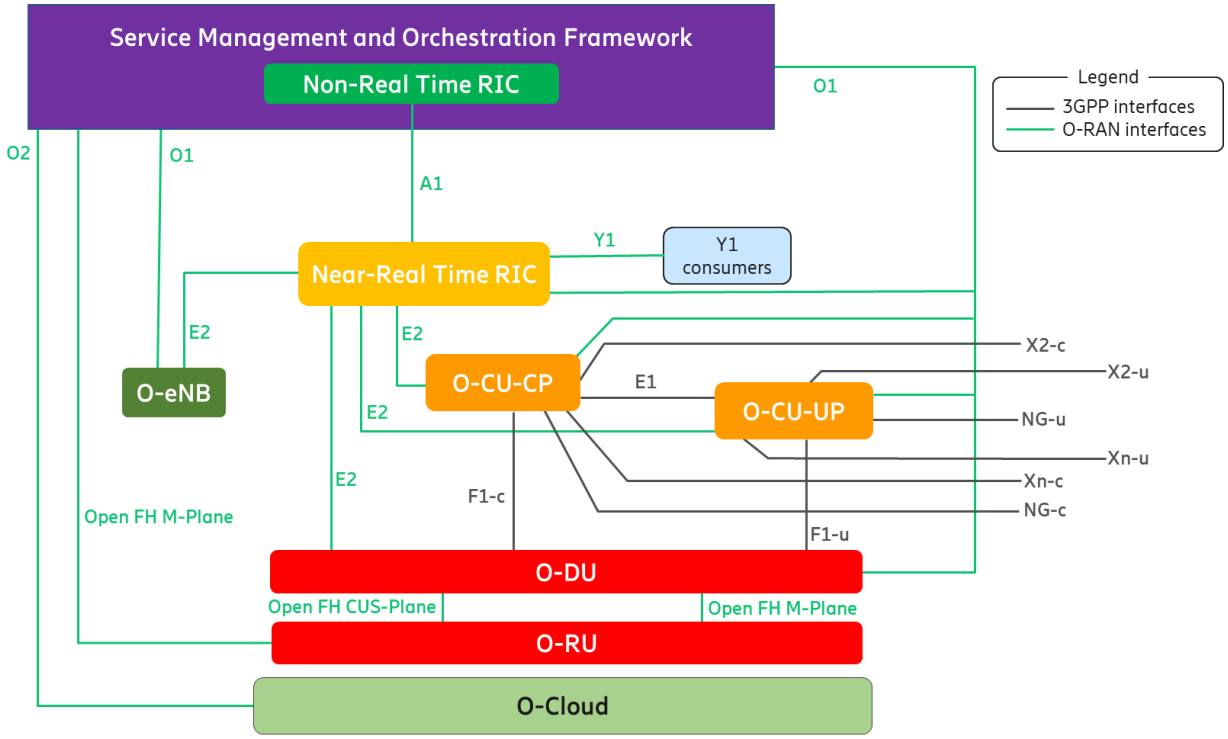

Figure 5-1: Logical Architecture of O-RAN system

Figure 5- illustrates the O-RAN architecture elements, interfaces, and overall system.

As specified in [1], the following O-RAN architecture elements and interfaces shall be the DUT or SUT addressed in this security test specification.

O-RAN architecture elements:

- Network functions and applications
  - Service Management and Orchestration (SMO)
  - Non-RT RIC and rApps
  - Near-RT RIC and xApps
  - O-CU-CP/UP
  - O-DU
  - O-RU
  - O-eNB

- Cloud computing platform
    - O-Cloud comprising a collection of physical infrastructure nodes that meet O-RAN requirements to host the relevant O-RAN functions (such as Near-RT RIC, O-CU-CP, O-CU-UP, and O-DU), the supporting software components (such as Operating System, Virtual Machine Monitor, Container Runtime, etc.) and the appropriate management and orchestration functions.

Maintained interfaces by O-RAN:

- A1 Interface between Non-RT RIC and Near-RT RIC to enable policy-driven guidance of Near-RT RIC applications/functions, and support AI/ML workflow.
- O1 Interface connecting the SMO to the Near-RT RIC, one or more O-CU-CPs, one or more O-CU-UPs, and one or more O-DUs.
- O2 Interface between the SMO and the O-Cloud
- E2 Interface connecting the Near-RT RIC and one or more O-CU-CPs, one or more O-CU-UPs, one or more O-DUs, and one or more O-eNBs.
- Open Fronthaul CUS-Plane Interface between O-RU and O-DU
- Open Fronthaul M-Plane Interface between O-RU and O-DU as well as between O-RU and SMO

During the test execution, only one DUT or SUT shall be tested at the same time. The rest of elements involved in the test setup should be simulated or real, according to the test preconditions, but only the DUT or SUT shall be considered under evaluation.

5.2 Test Setup

Please refer to the security test cases listed in the following clauses for their specific test setups.

5.3 Test and measurement equipment and tools

The following table lists test and measurement equipment required for the security tests in this document.

Table 5-1 Test and measurement equipment list

Test tool	Description
Commercial UE and/or UE emulator	A commercial UE or UE emulator shall be used to establish stateful end-to-end connection and to generate or receive data traffic. The commercial UE used in this context as a test tool is typically a UE which is designed for commercial or testing applications with certain test and diagnostic functions enabled for test and measurements purposes. Such test and diagnostic functions should not affect the performance. This commercial UE requires an (emulated) SIM card which is pre-provisioned with subscriber profiles. A UE emulator or multiple commercial UEs can be used in multi-UE test scenarios requiring multiple UEs sessions. The UE shall connect to the SUT either via RF cables or via an over the air (OTA) connection. In a lab environment, the UE shall be placed inside an RF shielded box/room to avoid interference from external signals. A logging tool connected to the UE shall be used to capture measurements and KPI logs for test validation and reporting.
4G/5G Core or Core emulator	A 4G/5G core or core emulator shall be used to terminate 4G/5G NAS sessions, and to support core network procedures required for RAN (SUT) testing. 4G/5G core or core emulator shall support end-to-end connection and data transfer between Application server and commercial UE/UE emulator.
Application (traffic) server	An application (traffic) server shall be used as an endpoint for generation and/or termination of data traffic streams to/from commercial UE(s)/UE emulator. The application server shall be capable of generating data traffic for the services under test.
Network impairment emulator	A network impairment emulator shall be used for tests which require insertion of impairment (packet delay and/or jitter) at the network interface (e.g. OpenFH).
Packet generation tool / DoS emulator	A packet generation tool / Denial of Service (DoS) emulator shall be used for DoS traffic generation of security tests. The tool shall support crafting network traffic over the following network protocols: Ethernet, IP, UDP, TCP, PTP, eCPRI, TLS, HTTP/HTTPS.
Packet capture tool	A packet capture tool shall be used to capture samples of data traffic for validation, analysis, and troubleshooting. It may be used to capture samples of legitimate traffic, which then may be used as templates for fuzzing attacks. The tool shall support capturing network traffic over the following network protocols: Ethernet, IP, UDP, TCP, PTP, eCPRI, TLS, QUIC, HTTP/HTTPS.
Network tap	A network tap shall be a hardware or software device which provides access and visibility to the data flowing across a computer network.
Port scanner	A protocol scanner shall be used for probing network protocols and services. It shall be able to detect open ports. It shall be able to detect what service is exposed as active on the open port. Port scanners commonly come with built-in database of services. Service detection can use numerous built-in probes for querying various services. In practice, port scanners are often used for service detection.
Fuzzing tool	A protocol fuzzing tool shall be used for unexpected protocol input generation of security tests. The tool shall support mutating and replaying of captured network traffic over the following network protocols: Ethernet, IP, UDP, TCP, PTP, eCPRI, TLS, HTTP/HTTPS.
Vulnerability scanning tool	A vulnerability scanning tool shall be used for blind exploitation of well-known vulnerabilities during security tests. The tool may rely on cyclically updated database of known vulnerabilities based on Common Vulnerabilities and Exposures (CVE) and should support scanning network services running on TCP/IP stack of protocols.

NFV benchmarking and resource exhaustion tool	A Network Function Virtualization (NFV) tool shall be used for O-Cloud system performance measurement and resource exhaustion type of DoS attack generation. This tool shall be capable of supporting any types of O-Cloud environment (public or private) with testing VNF(s) and/or CNF(s).
SSH audit tool	An SSH audit tool shall be used to verify the following properties: version of protocol, cipher suites, and known vulnerabilities in server and client SSH software.
TLS scanning tool	A TLS scanning tool shall be used to verify the following properties: version of protocol, cipher suites, and known vulnerabilities in server TLS software.
DTLS scanning tool	A DTLS scanning tool shall be used to verify the following properties: version of protocol, cipher suites, and known vulnerabilities in server DTLS software.
IKE scanning tool	An IKE scanning tool shall be used to verify the following properties: version of protocol, cipher suites, and known vulnerabilities in server IPsec software.
Software image signing tool	A Software image signing tool shall be used to digitally sign and verify the software image, e.g. xApps or O-RAN component delivered by a software producer/provider.

5.4 Test report

Tests should be described in the test report with sufficient detail to allow the tests to be reproducible by different parties and to enable comparison. A template for a complete test report is found in Annex B and may be used. Photos and screenshots should also be taken as part of the test report to illustrate the test environment. Additional parameters are specified in the description of each test in the subsequent clauses.

5.5 Assumptions

All threat IDs in the present document are referenced from O-RAN Security Threat Modeling and Risk Assessment [i.5].

5.6 Testing tools

The tools outlined in this clause represent a selection of commonly used resources for testing processes. It's important to emphasize that this list is not exhaustive. Testers are encouraged to use additional tools as needed for comprehensive and effective testing, ensuring they meet the standards and requirements set forth in this test plan.

1. Packet capture and traffic analysis tools:
    - Wireshark: Wireshark is a widely used open-source network protocol analyser that can capture and analyse network traffic. It enables you to inspect packets for confidentiality, integrity, and replay-related issues. You can also verify authentication mechanisms and analyse access control measures.
    - tcpdump: tcpdump is a command-line packet analyser available on various operating systems. It captures network traffic and can save it to a file for later analysis. tcpdump offers powerful filtering capabilities to capture specific traffic based on criteria such as source/destination IP addresses, protocols, or ports.
    - Netscout Sniffer: Netscout Sniffer is a commercial network analysis tool that offers real-time packet capture and analysis capabilities. It provides comprehensive visibility into network traffic and offers advanced features for troubleshooting and performance analysis.

- Colasoft Capsa: Colasoft Capsa is a network analyser designed for network monitoring and troubleshooting. It captures and analyses network traffic, providing insights into protocols, applications, and potential secusrity issues. Capsa offers both real-time and post-capture analysis.
- Tcpreplay: Tcpreplay is an open-source tool used for replaying captured network traffic. It allows you to replay network packets from a previously captured pcap file, simulating real-world traffic scenarios. While its primary purpose is not security testing, tcpreplay can be utilized as a tool in security testing efforts, particularly for testing the replay and handling of network packets.

2. Traffic Generation Tools:
   - Scapy: Scapy is a powerful Python-based tool that can create, manipulate, and send custom network packets. It allows to generate and replay packets on an interface to test for replay vulnerabilities.
   - Hping: Hping is a command-line tool that can send custom packets and perform various network-related activities. It can be used to generate replayed packets on an interface for testing purposes.
3. Scripting and Automation Tools to develop custom test scripts:
   - Python: Python scripting language provides libraries (e.g., socket, scapy) that allow you to write custom scripts to generate and replay packets on an interface.
   - Bash scripting: Bash scripting can be utilized to automate the process of capturing packets and replaying them on an interface.
4. Network Emulation Tools:
   - GNS3: GNS3 is a network emulation tool that allows you to simulate complex network topologies. It can be used to create a virtual environment with RAN E1 interfaces, generate traffic, and simulate replay attacks for testing purposes.
5. Network performance tools:
   - iperf3: iPerf3 is an open-source tool for network performance testing and measurement. While it is primarily focused on network performance evaluation, it can also be utilized as a tool to indirectly assess certain aspects of security, such as bandwidth availability and network congestion.
6. Traffic Manipulation Tools:
   - Burp Suite: Burp Suite is a web application security testing tool that can intercept, modify, and replay network traffic. While it is primarily designed for web applications, it allows to test the integrity, confidentiality, and authenticity of data transmitted over an interface.
7. Vulnerability assessment tools
   - Nessus: Nessus is a popular vulnerability assessment tool that can scan an interface for known security vulnerabilities and misconfigurations. It can help identify potential weaknesses related to confidentiality, integrity, replay attacks, and access control.
   - OpenVAS: OpenVAS (Open Vulnerability Assessment System) is an open-source vulnerability scanner that can perform security audits on security protocols implementations. It can detect vulnerabilities, misconfigurations, and compliance issues, helping ensure that an interface adheres to security best practices and standards.
8. Security Information and Event Management (SIEM) Tools:
   - SIEM tools like Splunk or ELK (Elasticsearch, Logstash, Kibana) can help collect and analyse security events and logs related to the O-RAN components and interfaces. They can assist in identifying potential security incidents, monitoring access control, and detecting anomalies.
9. IPsec tool

- OpenSwan is an open-source implementation of the IPsec (Internet Protocol Security) protocols suite. It provides tools and libraries for setting up and managing IPsec connections, which can be used to test the confidentiality, integrity, replay, authenticity, and access control of an interface. Here's how OpenSwan can be used for testing:
  - Confidentiality and Integrity:
    - OpenSwan allows to configure IPsec tunnels with encryption algorithms (e.g., AES) and integrity algorithms (e.g., HMAC-SHA256). By setting up IPsec connections using OpenSwan, it is possible to verify the confidentiality and integrity of data transmitted over an interface.
  - Replay Attack:
    - OpenSwan supports anti-replay mechanisms, which protect against replay attacks by assigning sequence numbers to IPsec packets. These mechanisms can be tested to ensure that replayed packets are detected and rejected.
  - Authenticity:
    - OpenSwan supports authentication mechanisms such as pre-shared keys or digital certificates, which ensure the authenticity of IPsec connections. Testing can be performed to verify the proper authentication of an interface.
  - Access Control:
    - OpenSwan allows to configure IPsec security policies, including source/destination IP address filtering, protocol filtering, and port filtering. These policies can be tested to ensure that only authorized traffic is allowed through an interface.
- StrongSwan: Another open-source IPsec-based VPN solution that includes testing capabilities. It allows you to configure and simulate IPsec connections, test authentication methods, and perform security checks.

10. Cryptographic operations testing tools
    - Hashing Tools: Hashing tools such as sha256sum, can be used to calculate hash values of transmitted data. By comparing the computed hash values at the source and destination, you can verify the integrity of the data.

Cryptographic Libraries: Cryptographic libraries, such as Bouncy Castle, provide APIs and tools for implementing and testing integrity protection mechanisms. These libraries offer functions to generate integrity checks (e.g., MAC) and validate the integrity of received data.

6 Security Protocol & APIs Validation

6.1 Overview

This clause contains test cases to validate implementation of security protocols against O-RAN security requirements in [2] and [5].

6.2 SSH Server & Client

Requirement Name: Network Security Protocol - SSH

Requirement Reference: Clause 4.1, O-RAN Security Protocols Specifications [2]

Requirement Description: Robust protocol implementation with adequately strong cipher suites is being required for SSH

Threat References: T-O-RAN-05

DUT/s: SMO, O-DU, O-RU

Test Name: TC_SSH_Server_and_Client_Protocol

Purpose: To verify implementation of the secure communication protocol SSH as specified in [2].

Procedure and execution steps

Preconditions

- Tool: SSH audit tool with capabilities as defined in clause 5.3
- Testing of server configuration (DUT in the role of server): Network access to SSH server
- Testing of client configuration (DUT in the role of client): Access to configuration of SSH client

Execution steps

Testing of server configuration (DUT in the role of server):

- Run SSH audit tool in server audit mode against target SSH server
- Compare the tool's output with the list of approved SSH protocol versions and algorithms (for host key, symmetric encryption, key exchange, and MACs) as defined by clause 4.1, O-RAN Security Protocols Specifications [2].

Testing of client configuration (DUT in the role of client):

- Run SSH audit tool on target SSH client in client audit mode
- Compare the tool's output with the list of approved SSH protocol versions and algorithms (for host key, symmetric encryption, key exchange, and MACs) as defined by clause 4.1, O-RAN Security Protocols Specifications [2].

Expected results

- All detected SSH protocol versions are explicitly allowed by [2], clause 4.1.
- All detected SSH algorithms (for host key, symmetric encryption, key exchange, and MACs) are explicitly allowed by [2], clause 4.1.

Expected format of evidence: Report files produced by SSH audit tool and/or screenshots

6.3 TLS

Requirement Name: Network Security Protocol - TLS

Requirement Reference: Clause 4.2, O-RAN Security Protocols Specifications [2]

Requirement Description: Support TLS with protocol profiles

Threat References: T-O-RAN-05

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_TLS_Protocol

Purpose: To verify implementation of the secure communication protocol TLS as specified in [2], clause 4.2.

Procedure and execution steps

Preconditions

- Tool: TLS scanning tool with capabilities as defined in clause 5.3 with client certificate(s) installed;
- DUT equipped with CA certificate that is a trust anchor for the client certificate(s)
- Network access to DUT with exposed TLS server

Execution steps

- Protocol scanning
    - Run TLS scanning tool against DUT for detection of:
        - TLS version
        - Cipher suites
        - Elliptic curves
        - Certificate type
        - Diffie-Hellman groups
        - Compression methods
    - Compare the test result/report with the list of approved TLS versions and profiles as defined by O-RAN Security Protocols Specifications [2], clause 4.2.
- Mutual Authentication
    - Run TLS scanning tool with TLS v1.2 and valid client certificate against DUT with mutual authentication enabled to verify the establishment of the TLS session after successful authentication
    - Run TLS scanning tool with TLS v1.2 and invalid client certificate (including but not limited to expired certificate, missing field certificate, untrusted CA signed certificate, …) against DUT with mutual authentication enabled to verify the failed attempt of the TLS session establishment due to certificate validation
    - Run TLS scanning tool with TLS v1.3 and valid client certificate against DUT with mutual authentication enabled to verify the establishment of the TLS session after successful authentication
    - Run TLS scanning tool with TLS v1.3 and invalid client certificate (including but not limited to expired certificate, missing field certificate, untrusted CA signed certificate, …) against DUT with mutual authentication enabled to verify the failed attempt of the TLS session establishment due to certificate validation

Expected results

- All supported TLS protocol versions are explicitly allowed by [2], clause 4.2.
- All detected TLS cipher suites, elliptic curves, Diffie-Hellman groups and compression methods are explicitly allowed by [2], clause 4.2.
- Mutual authentication support works with client certificates

Expected format of evidence: Report files produced by TLS scanning tool and/or screenshots

6.4 DTLS

Requirement Name: Network Security Protocol - DTLS

Requirement Reference: Clause 4.4, O-RAN Security Protocols Specifications [2]

Requirement Description: Support DTLS

Threat References: T-O-RAN-01

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_DTLS_Protocol

Purpose: To verify implementation of the secure communication protocol DTLS as specified in [2], clause 4.4.

Procedure and execution steps

Preconditions

- Tool: DTLS scanning tool with capabilities as defined in clause 5.3
- Network access to DUT

Execution steps

- Run DTLS scanning tool against DUT for detection of:
  - DTLS version
  - Cipher suites
  - Elliptic curves
  - Certificate type
  - Diffie-Hellman groups
  - Compression methods
- Compare the test result/report with the list of approved DTLS versions and profiles as defined by Security Protocols Specification [2], clause 4.4.

Expected results

- All supported DTLS protocol versions are explicitly allowed by [2], clause 4.4.
- All detected DTLS cipher suites, elliptic curves, Diffie-Hellman groups and compression methods are explicitly allowed by [2], clause 4.4.

Expected format of evidence: Report files produced by DTLS scanning tool and/or screenshots.

6.5 IPsec

6.5.1 IPsec security

Requirement Name: Network Security Protocol - IPsec

Requirement Reference: Clause 4.5, O-RAN Security Protocols Specifications [2]

Requirement Description: Support IPsec tunnel mode with confidentiality, integrity, authentication, and anti-replay protection.

Threat References: T-O-RAN-05

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_IPsec_Security

Purpose: To verify implementation of the secure communication protocol IPsec.

Procedure and execution steps

Preconditions

- Tool: IKE scanning tool with capabilities as defined in clause 5.3
- Network access to DUT

Execution steps

- Run IKE scanning tool against DUT for detection of:
  - ESP Encryption Transforms
  - ESP Authentication Transforms
  - Diffie-Hellman groups
  - Certificate type
- Compare the test result/report with the list of approved IPsec IKE versions, ESP Encryption Transforms, ESP Authentication Transforms and Diffie-Hellman groups as defined by O-RAN Security Protocols Specification [2], clause 4.5.

Expected results

- All detected IPsec IKE versions are explicitly allowed by [2], clause 4.5.
- All detected ESP Encryption Transforms, ESP Authentication Transforms and Diffie-Hellman groups are explicitly allowed by [2], clause 4.5.IKE version (v2) support with no older version(s) enabled.
- If certificates are used, their format is X.509v3

Expected format of evidence:

- .pcap files capturing the IKE negotiations between the tool and the DUT.
- Report or output from the IKE scanning tool, specifically highlighting:
  - Detected ESP Encryption Transforms.
  - Detected ESP Authentication Transforms.

- Detected Diffie-Hellman groups.
- Detected Certificate type.

- Screenshots from the IKE scanning tool showing scan results, especially the supported IKE version detected.
- If certificates are used, a sample or screenshot verifying the X.509v3 format.

6.5.2 IKE Header Flags Fuzzing

Requirement Name: Network Security Protocol - IPsec

Requirement Reference: Clause 4.5, O-RAN Security Protocols Specifications [2]

Requirement Description: Support IPsec tunnel mode with confidentiality, integrity, authentication, and anti-replay protection.

Threat References: T-O-RAN-01

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_IKE_HEADER_FLAGS_FUZZING

Purpose: The purpose of this test is to verify the robustness of the IKEv2 server when faced with malformed IKE headers. Flags within the IKE header are intended to provide specific instructions or information about the message. By fuzzing these flags, we can identify potential vulnerabilities or flaws in the server's processing logic.

Procedure and execution steps

Preconditions

- A controlled environment with an IKEv2 server and a test client.
- Packet capture tool (e.g., Wireshark) for monitoring the traffic.
- Fuzzing tool or script to generate malformed IKE header flags.

Execution steps

1. Begin by starting the packet capture tool to record the test session.
2. Use the fuzzing tool or script to generate IKEv2 messages with the following malformed flags in the IKE header:
   - **Initiator flag**: Flip this flag to see if the server can identify a message that isn't from an initiator.
   - **Version flag**: Introduce an unsupported version.
   - **Response flag**: Send messages that have this flag inappropriately set.
   - **Combination of multiple flags**: Mix flags to generate completely unexpected combinations.
3. Send each of these malformed messages to the IKEv2 server individually, waiting for a response before sending the next.
4. Observe server reactions, looking specifically for any unhandled exceptions, crashes, or irregular behaviours.

Expected Results

- The IKEv2 server handles the malformed flags gracefully, either by rejecting the message or by ignoring the unexpected flag values.
- There is no crashes, hangs, or undefined behaviours.

Expected format of evidence

- Packet capture files (.pcap) showing the malformed flags sent and the server's responses.
- Server logs indicating the handling (or rejection) of the malformed messages.

6.5.3 IKE Key Exchange Payload Fuzzing

Requirement Name: Network Security Protocol - IPsec

Requirement Reference: Clause 4.5, O-RAN Security Protocols Specifications [2]

Requirement Description: Support IPsec tunnel mode with confidentiality, integrity, authentication, and anti-replay protection.

Threat References: T-O-RAN-01

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_IKE_KEY_EXCHANGE_PAYLOAD_FUZZING

Purpose: The purpose of this test is to examine the IKEv2 server's ability to manage corrupted or unexpected data within the Key Exchange (KE) payload. The KE payload carries the Diffie-Hellman public value. If the server is unable to handle malformed KE payloads, it might be susceptible to attacks or crashes.

Procedure and execution steps

Preconditions

- A controlled environment with an IKEv2 server and a test client.
- Packet capture tool (e.g., Wireshark) for monitoring the traffic.
- Fuzzing tool or script capable of generating malformed KE payloads.

Execution steps

1. Start the packet capture tool to ensure you record every detail of the test session.
2. Use your fuzzing tool or script to generate IKEv2 messages with the following specific manipulations in the KE payloads:
    - Unexpected length: Prepare 10 distinct messages where the KE payload's declared length is longer or shorter than the actual payload.
    - Corrupted data: Generate 10 messages introducing random bytes into the KE payload to see how the server handles non-standard values.
    - Unsupported Diffie-Hellman groups: Create 5 messages attempting to initiate a key exchange using a DH group that is either deprecated or not supported by the server.
    - Empty KE payload: Formulate 5 messages with an empty KE payload.
3. Sequentially send these 30 malformed messages to the IKEv2 server. After sending each message, wait for the server's response to avoid overloading it. Ensure the following sequence:
    - Send the 10 "Unexpected Length" messages.
    - Follow with the 10 "Corrupted Data" messages.
    - Continue with the 5 "Unsupported Diffie-Hellman Groups" messages.

- Conclude with the 5 "Empty KE Payload" messages.

NOTE: Monitor the server's reactions closely. The server ideally handles errors gracefully, either ignoring them or responding with an appropriate error message, without any crashes or hangs.

Expected Results

- The IKEv2 server gracefully handles the malformed KE payloads, either by ignoring them, responding with an error, or requesting a valid KE payload.
- No crashes, hangs, or undefined behaviours occur.

Expected format of evidence

- Packet capture files (.pcap) highlighting the malformed KE payloads and the server's corresponding responses.
- Server logs detailing the handling (or rejection) of the malformed KE payloads.

6.5.4 IKE Malformed Certificate Payload

Requirement Name: Network Security Protocol - IPsec

Requirement Reference: Clause 4.5, O-RAN Security Protocols Specifications [2]

Requirement Description: Support IPsec tunnel mode with confidentiality, integrity, authentication, and anti-replay protection.

Threat References: T-O-RAN-01

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_IKE_MALFORMED_CERTIFICATE_PAYLOAD

Purpose: This test aims to verify the IKEv2 server's capability to properly validate certificate payloads. Certificate payloads are essential in the IKEv2 authentication phase. A server vulnerable to malformed certificate payloads could be susceptible to impersonation or man-in-the-middle attacks.

Procedure and execution steps

Preconditions

- A controlled environment with an IKEv2 server and a test client.
- Packet capture tool (e.g., Wireshark) to monitor and capture traffic.
- A set of both valid and deliberately malformed certificates.

Execution steps

1. **Valid Certificate Test:**
    1. Initiate an IKEv2 session using a valid certificate to ensure baseline functionality.
    2. Confirm successful authentication and session establishment.
2. **Expired Certificate:**
    1. Use a previously valid certificate that has now expired.
    2. Attempt to initiate an IKEv2 session.
    3. Observe the server's rejection of this certificate.

3. **Certificate with Invalid Signature:**
    1. Modify a valid certificate's content slightly (e.g., change an attribute) without re-signing it. This will invalidate its signature.
    2. Attempt to initiate an IKEv2 session using this certificate.
    3. The server detects the invalid signature and rejects the connection.
4. **Certificate from Untrusted Authority:**
    1. Generate a new certificate signed by a Certificate Authority (CA) that the IKEv2 server doesn't trust or recognize.
    2. Attempt to initiate a connection using this certificate.
    3. Observe the server rejecting the certificate due to the untrusted CA.
5. **Certificate with Modified Subject/Issuer Fields:**
    1. Modify the subject or issuer fields of a certificate to contain irregular or unexpected values (e.g., overly long strings, special characters).
    2. Use this certificate to initiate an IKEv2 session.
    3. The server validates these fields, notices the irregularities, and potentially rejects the connection.
6. **Certificate with Invalid Key Usage:**
    1. Use a certificate that doesn't have "key encipherment" or "digital signature" as its key usage, which are typically needed for IKEv2 operations.
    2. Attempt to initiate a session.
    3. The server detects the inappropriate key usage and declines the connection.

Expected Results:

- For the valid certificate, the IKEv2 server authenticates successfully and establish a session.
- For all other scenarios, the IKEv2 server detects the certificate anomalies and rejects the connection attempts. Specific error messages or logs relating to certificate validation failure are generated.

Expected format of evidence:

- Packet capture files (.pcap) capturing the entire exchange, showing the certificate exchange and the server's response.
- Server logs detailing the acceptance or rejection of each certificate, with corresponding reasons or error messages for rejections.

6.6 OAuth 2.0

Requirement Name: Authorization based on OAuth 2.0

Requirement Reference: Clause 4.7, O-RAN Security Protocols Specifications [2]

Requirement Description: O-RAN OAuth 2.0 based authorization including resource registration, access token request and service access request based on token verification process as defined in [2]

Threat References: T-O-RAN-05

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_OAuth2.0_Protocol

Purpose: To verify implementation of the authorization for API consumer requests received by API producer based on OAuth 2.0 as specified in [2]

Procedure and execution steps

Preconditions

- Supported version of OAuth 2.0
- Access token request process
    - Mutual TLS authentication is required
- Service access request based on token verification process as defined in [2]
    - Mutual TLS authentication is required
- API consumer – O-RAN architecture element as referred in DUT
- Resource owner/server – Service Producer as DUT
- Authorization server – OAuth 2.0 Authorization Server (real or emulated)
    - API consumer registration can be a manual process with client profile pre-provisioned on the Authorization server based on client certificate's subject alternative name field
    - Resource owner/server registered with the Authorization server for its supported API service(s)
        - This process can be a manual or automatic process preceding with Resource owner/server authentication
        - The API service profile(s) follows the definition in O-RAN specifications
- TLS service enabled on the Authorization Server, Resource owner/server and API consumer with all the required keys, root and/or immediate (if necessary) CA certificates required for mutual TLS authentication procedure
- Network access to Authorization Server, Resource owner/server and API consumer

Execution steps

Access token request process validation

1. Access token request process with valid client certificate and parameters

    API consumer makes the access token request towards Authorization server over secured TLS communication session with mutual TLS authentication;

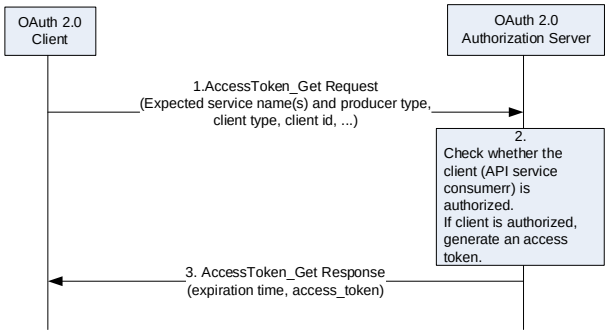

Figure 6-1: Access Token request

Verify the session is established between the API consumer and Authorization server, and the access token request is processed with a successful response with digitally signed JSON Web Signature (JWS) as described in RFC 7515 [20] by the Authorization server.

2. Access token request process with wrong client certificate

   API consumer sends the access token request towards Authorization server over secured TLS communication session with mutual TLS authentication;

   Verify the session establishment in between the API consumer and Authorization server is not possible.

3. Access token request process with incorrect parameters

   API consumer sends the access token request with incorrect parameters towards Authorization server over secured TLS communication session with mutual TLS authentication;

   Verify the session is established between the API consumer and Authorization server, and the access token request is processed with a failed response by the Authorization server with error code defined in RFC 6749 [18].

Service access request based on token verification process as defined in [2]

1. Service access request based on token verification process with valid access token

   API consumer sends an API service request towards Resource owner/server (API producer) using the access token obtained as a response to access token request over a secured TLS communication session with mutual TLS authentication;

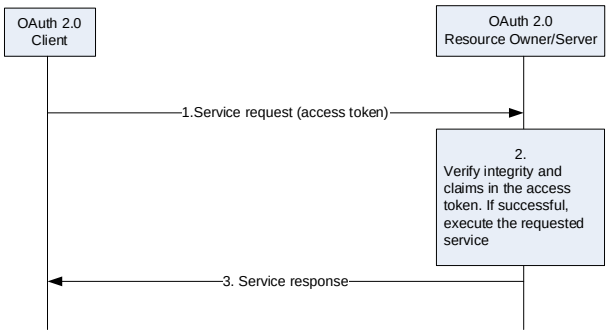

Figure 6-2: Service request

Verify the session is established between the API consumer and Resource owner/server (API producer), and the service request is processed with a response by the Resource owner/server (API producer).

2. Service access request based on token verification process with incorrect access token

   API consumer sends an API service request towards Resource owner/server (API producer) using an incorrect access token over a secured TLS communication session with mutual TLS authentication;

   Verify the session is established between the API consumer and Resource owner/server (API producer), and the service request is processed with a failed response (401) by the Resource owner/server (API producer).

Expected results

The API consumers are able to execute API service(s) call with OAuth 2.0 based authorization.

Expected format of evidence: Log files, traffic captures and/or screenshots.

6.7 NACM

6.7.1 NACM RBAC Configuration

Requirement Name: NACM security

Requirement Reference: REQ-NAC-FUN-1 to REQ-NAC-FUN-10, clause 5.2.2.1.3 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Netconf/NACM support

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_NACM_RBAC_CONFIGURATION

Purpose: The purpose of this test is to verify the RBAC configuration for secure access control on the TLS-based NACM with NETCONF.

Procedure and execution steps

Preconditions

- The NACM and NETCONF services are properly configured and operational.
- The RBAC feature is supported and enabled in the NACM system.
- RBAC roles, access control rules, and denied resources or operations are properly defined.

Execution steps

1. Verify RBAC role definitions.
    - Check that RBAC roles are properly defined for access control.
        a) Review the RBAC role definitions.
            - EXAMPLE: Command "“**show nacm rbac roles**"
        b) Validate that the defined roles match the intended access control requirements.
2. Verify RBAC role assignment.
    - Test the assignment of RBAC roles to users or user groups.
        a) Assign roles to users or user groups.
            - EXAMPLE: "Command: **configure nacm rbac role-assignment**"
        b) Verify that the assigned roles are reflected in the configuration.
3. Verify unauthorized access denial.
    - Test access to resources or operations that are not permitted for a specific RBAC role.
        a) Identify a resource or operation that is denied for a specific role.
            - EXAMPLE: "Command **show nacm rbac role-permissions <role_name>**"
        b) Attempt to access the denied resource or operation with a user assigned to the role.

- EXAMPLE: "Command: **execute netconf operation <operation_name>**"

Expected Results

1. For step 1), Roles are defined with their associated permissions and restrictions.
2. For step 2), Roles are assigned to the appropriate users or user groups.
3. For step 3)-a, The denied resource or operation is listed for the specified role.
4. For step 3)-b, Access to the denied resource or operation is denied, and an appropriate error message is displayed.

Expected format of evidence

1. For step 1), The output of the **show nacm rbac roles** command, showing the defined roles and their associated permissions and restrictions.
2. For step 2), Confirmation that the roles have been successfully assigned to the appropriate users or user groups, as reflected in the configuration.
3. For step 3), An appropriate error message indicating access denial when attempting to access a denied resource or operation with a user assigned to a specific role.

6.7.2 NACM Logging Monitoring

Requirement Name: NACM security

Requirement Reference: REQ-NAC-FUN-1 to REQ-NAC-FUN-10, clause 5.2.2.1.3 in O-RAN Security Requirements and Controls Specifications [5], REQ-SEC-SLM-AAI-EVT-1 to REQ-SEC-SLM-AAI-EVT-10, clause 5.3.8.11.6 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Netconf/NACM support

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_NACM_LOGGING_MONITORING

Purpose: The purpose of this test is to verify the logging and monitoring configuration for the TLS-based NACM with NETCONF.

Procedure and execution steps

Preconditions

- The NACM and NETCONF services are properly configured and operational.
- Logging and monitoring systems are in place, integrated and configured with the NACM system.

Execution steps

1. Verify logging configuration.
    - Check that logging is properly configured to capture relevant security-related events.
        a) Review the logging configuration settings.
            - EXAMPLE: "Command: **show nacm logging configuration**"
        b) Trigger security-related events (e.g., access violations, failed authentication attempts) and validate that the events are logged.

2. Verify monitoring configuration.
    - Test the monitoring configuration to ensure that security-related events and performance metrics are monitored.
        a) Review the monitoring configuration settings.
            - EXAMPLE: "Command: **show nacm monitoring configuration**"
        b) Trigger security-related events or exceed performance thresholds and verify that the monitoring system captures and reports these events or metrics.
3. Verify audit log review.
    - Test the ability to review audit logs for security-related events.
        a) Retrieve the audit logs.
            - EXAMPLE: "Command: **show nacm audit-logs**"
        b) Review the audit logs to ensure that they contain the expected information and provide a detailed record of security-related activities.

Expected Results

1. For step 1), Logging is enabled with appropriate log levels, log destinations, and log retention policies.
2. For step 2), Monitoring is enabled with appropriate metrics, thresholds, and alerting mechanisms.
3. For step 3), Audit logs containing security-related events are available.

Expected format of evidence

1. Confirmation that logging is enabled with the expected log levels, log destinations, and log retention policies. Additionally, evidence of captured security-related events in the logs.
2. Confirmation that monitoring is enabled with the configured metrics, thresholds, and alerting mechanisms. Evidence of captured security-related events or performance metrics exceeding thresholds.
3. The audit logs containing security-related events, demonstrating that they contain the expected information and provide a detailed record of security-related activities.

6.7.3 Void

6.8 802.1X

Void

6.9 X.509

6.9.1 X.509 Certificate Structure Verification for TLS

Requirement Name: X.509 security

Requirement Reference: SEC-CTL-O1-1, clause 5.2.2. 2, SEC-CTL-O2, clause 5.2.3.1, REQ-SEC-OCLOUD-O2dms-1 to REQ-SEC-OCLOUD-O2dms-3, clause 5.1.8.9.1.1, REQ-SEC-OCLOUD-O2ims-1 to REQ-SEC-OCLOUD-O2ims-3, clause 5.1.8.9.1.2, REQ-SEC-O-CLOUD-NotifAPI-1 to REQ-SEC-O-CLOUD-NotifAPI-2, clause

5.1.8.9.1.3, SEC-CTL-A1-1, SEC-CTL-A1, clause 5.2.1.1 [5], SEC-CTL-R1-1, SEC-CTL-R1, clause 5.2.6.1, REQ-SEC-Y1-1 to REQ-SEC-Y1-3, clause 5.2.7.2, O-RAN Security Requirements and Controls Specifications [5], clause 4.2.3 in O-RAN Security Protocols Specifications [2].

Requirement Description: To verify the X.509 certificate structure used by TLS components in all the ORAN systems.

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_X509_CERT_STRUCTURE_VERIFICATION

Purpose: The purpose of this test is to ensure that the X.509 certificate follows the correct structure and format for TLS. This test is applicable to X.509 certificates used in TLS modules in the O-RAN system.

Procedure and execution steps

Preconditions:

- End entity X.509 certificate in ASN.1 format.
- CA certificate that signed the end entity certificate is available to the test suite.

Execution steps

Following properties of the certificate are verified.

- Certificate Version field, Subject field, Issuer, Validity of certificate, Key Usage.

As defined in clause 4.2.3 in O-RAN Security Protocols Specifications [2], the following certificate fields are present.

1. **Certificate Fields Examination:**
    - Validate the certificate's version field. The version number is set to v3.
    - Verify the Subject field is present in the certificate. Verify that the Subject field conforms to the format defined in clause 4.2.3 in O-RAN Security Protocols Specifications [2].
    - Verify the Validity field to ensure the "Not Before" date is earlier than the "Not After" date, indicating a valid time range for the certificate's use.
    - Validate that the certificate’s signature field is based on the algorithm used by the CA to sign the certificate.
2. **Key Usage Extension:**
    - Validate the presence of the Key Usage Extension. Verify that it conforms with the mandatory critical Key Usage extension as specified in the clause 4.2.3 in O-RAN Security Protocols Specifications [2].
3. **CRL Distribution Point:**
    - Validate that the certificate contains the cRLDistributionPoint extension.
4. **Subject Alternative Name:**
    - Validate that the certificate contains the subjectAltName extension and conforms to the requirements as specified in the clause 4.2.3 in O-RAN Security Protocols Specifications [2].

Expected Results

The certificate adheres to the X.509 standard structure as defined in clause 4.2.3 in O-RAN Security Protocols Specifications [2].

Expected format of evidence

Log or report indicating successful certificate structure validation.

6.9.2 X.509 Certificate Validity Period Verification

Requirement Name: X.509 security

Requirement Reference: SEC-CTL-O1-1, clause 5.2.2.1.2, REQ-SEC-O2-1, clause 5.2.3.1, REQ-SEC-OCLOUD-O2dms-1 to REQ-SEC-OCLOUD-O2dms-3, clause 5.1.8.9.1.1, REQ-SEC-OCLOUD-O2ims-1 to REQ-SEC-OCLOUD-O2ims-3, clause 5.1.8.9.1.2, REQ-SEC-O-CLOUD-NotifAPI-1 to REQ-SEC-O-CLOUD-NotifAPI-2, clause 5.1.8.9.1.3, REQ-SEC-A1-1, REQ-SEC-A1-2, clause 5.2.1.1, REQ-SEC-E2-1, clause 5.2.4.1, REQ-SEC-R1-1, REQ-SEC-R1-2, clause 5.2.6.1, REQ-SEC-Y1-1 to REQ-SEC-Y1-3, clause 5.2.7.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Correctness of X.509 certificate

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_X509_CERT_VALIDITY_PERIOD_VERIFICATION

Purpose: The purpose of this test is to ensure that the certificate's validity dates are accurate and within an acceptable range. This test is relevant for all X.509 certificates within the O-RAN system.

Procedure and execution steps

Preconditions

- Prepare certificates with different validity periods (valid, expired, not yet valid).

Execution steps

1. **Verify a Valid Certificate:**
    - Set up a valid certificate with appropriate "Not Before" and "Not After" dates.
    - Verify that the certificate is accepted when used for its intended purpose.
2. **Verify an Expired Certificate:**
    - Set up a certificate with a past expiration date.
    - Attempt to use the expired certificate for its intended purpose.
    - Verify that the certificate is rejected due to expiration.
3. **Verify a Not Yet Valid Certificate:**
    - Set up a certificate with a "Not Before" date in the future.
    - Attempt to use the certificate before the valid start date.
    - Verify that the certificate is rejected due to being not yet valid.

Expected Results

Valid certificates are accepted, while expired and not-yet-valid certificates are rejected.

Expected format of evidence

Log or report showing successful validation and rejection for different validity periods.

6.9.3 X.509 Certificate Key Usage Verification

Requirement Name: X.509 security

Requirement Reference: SEC-CTL-O1-1, clause 5.2.2.1.2, REQ-SEC-O2-1, clause 5.2.3.1, REQ-SEC-OCLOUD-O2dms-1 to REQ-SEC-OCLOUD-O2dms-3, clause 5.1.8.9.1.1, REQ-SEC-OCLOUD-O2ims-1 to REQ-SEC-OCLOUD-O2ims-3, clause 5.1.8.9.1.2, REQ-SEC-O-CLOUD-NotifAPI-1 to REQ-SEC-O-CLOUD-NotifAPI-2, clause 5.1.8.9.1.3, REQ-SEC-A1-1, REQ-SEC-A1-2, clause 5.2.1.1, REQ-SEC-E2-1, clause 5.2.4.1, REQ-SEC-R1-1, REQ-SEC-R1-2, clause 5.2.6.1, REQ-SEC-Y1-1 to REQ-SEC-Y1-3, clause 5.2.7.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Correctness of X.509 certificate

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_X509_CERT_KEY_USAGE_VERIFICATION

Purpose: The purpose of this test is to confirm that the certificate's key usage and extended key usage extensions are correctly defined.

Procedure and execution steps

Preconditions

- Prepare certificates with different key usage and extended key usage extensions.

Execution steps

1. **Verify a Certificate with Correct Usage Extensions:**
    - Set up a certificate with proper key usage and extended key usage extensions matching its intended purpose (e.g., server authentication).
    - Attempt to use the certificate for its designated purpose.
    - Verify that the certificate is accepted.
2. **Verify a Certificate with Incorrect or Missing Usage Extensions:**
    - Set up a certificate with incorrect or missing key usage or extended key usage extensions.
    - Attempt to use the certificate for its intended purpose.
    - Verify that the certificate is rejected.

Expected Results

Certificates with correct usage extensions are accepted, while those with incorrect or missing extensions are rejected.

Expected format of evidence

Log or report indicating successful validation and rejection for different key usage scenarios.

6.9.4 X.509 Certificate Chain Validation

Requirement Name: X.509 security

Requirement Reference: SEC-CTL-O1-1, clause 5.2.2.1.2, REQ-SEC-O2-1, clause 5.2.3.1, REQ-SEC-OCLOUD-O2dms-1 to REQ-SEC-OCLOUD-O2dms-3, clause 5.1.8.9.1.1, REQ-SEC-OCLOUD-O2ims-1 to REQ-SEC-

OCLOUD-O2ims-3, clause 5.1.8.9.1.2, REQ-SEC-O-CLOUD-NotifAPI-1 to REQ-SEC-O-CLOUD-NotifAPI-2, clause 5.1.8.9.1.3, REQ-SEC-A1-1, REQ-SEC-A1-2, clause 5.2.1.1, REQ-SEC-E2-1, clause 5.2.4.1, REQ-SEC-R1-1, REQ-SEC-R1-2, clause 5.2.6.1, REQ-SEC-Y1-1 to REQ-SEC-Y1-3, clause 5.2.7.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Correctness of X.509 certificate chain

Threat References: T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_X509_CERT_CHAIN_VALIDATION

Purpose: The purpose of this test is to validate the certificate chain's integrity and trustworthiness. This test is applicable to scenarios where certificates are part of a chain (e.g., intermediate and root certificates).

Procedure and execution steps

Preconditions

- Prepare a certificate chain with correct and incorrect configurations.

Execution steps

1. **Verify a Certificate Chain with Correct Order and Valid Signatures:**
    - Set up a valid certificate chain with proper order and valid signatures.
    - Attempt to use the certificate chain for its intended purpose.
    - Verify that the certificate chain is accepted.
2. **Verify a Certificate Chain with Incorrect Order or Invalid Signatures:**
    - Set up a certificate chain with incorrect order or invalid signatures.
    - Attempt to use the certificate chain for its intended purpose.
    - Verify that the certificate chain is rejected.

Expected Results

A valid certificate chain is accepted, while an invalid chain is rejected.

Expected format of evidence

Log or report indicating successful validation and rejection for different certificate chain scenarios.

6.10 eCPRI

6.10.1 Void

6.10.2 eCPRI Input Validation

Requirement Name: eCPRI security

Requirement Reference: REQ-SEC-TRAN-1, clause 5.3.4.1 in O-RAN Security Requirements and Controls Specifications [5].

Requirement Description: eCPRI protocol robustness – ability to handle unexpected inputs (not in-line with protocol specification) without functional compromise.

Threat References: T-FRHAUL-01, T-FRHAUL-02

DUT/s: O-RU, O-DU

Test Name: TC_eCPRI_INPUT_VALIDATION

Purpose: The purpose of this test is to ensure that the DUT properly validates and sanitizes input through the eCPRI interface to prevent common security vulnerabilities such as injection attacks.

Procedure and execution steps

Preconditions

- eCPRI interface is accessible.
- Input fields requiring validation are identified.

Execution steps

1. Positive Case:
    a) Send requests through the eCPRI interface with valid and expected input values.
    b) Verify that the DUT processes the requests successfully and provides the expected responses.
2. Negative Case:
    a) Generate requests through the eCPRI interface by systematically applying fuzzing techniques to introduce deliberately malicious input values containing potential security threats.
    b) Verify that the DUT detects and rejects the malicious input, responding with appropriate error messages or status codes.

Expected Results: The DUT validates and sanitizes input through the eCPRI interface to prevent security vulnerabilities related to improper input handling.

Expected format of evidence

- A log file documenting the requests sent to the DUT through the eCPRI interface, including valid and malicious inputs.
- Screenshots of the DUT responses through the eCPRI interface showing the handling of valid inputs and appropriate error messages for malicious inputs.

6.10.3 eCPRI Error and Timeout Handling

Requirement Name: eCPRI security

Requirement Reference: REQ-SEC-TRAN-1, clause 5.3.4.1 in O-RAN Security Requirements and Controls Specifications [5].

Requirement Description: eCPRI protocol robustness - ability to handle unexpected inputs (not in-line with protocol specification) without functional compromise.

Threat References: T-FRHAUL-01, T-FRHAUL-02

DUT/s: O-RU, O-DU

Test Name: TC_eCPRI_ERROR_TIMEOUT_HANDLING

Purpose: The purpose of this test is to ensure that the DUT securely handles errors on the eCPRI interface, including malformed packets, unexpected messages, and timeout scenarios, without disclosing sensitive information or compromising DUT stability.

Procedure and execution steps

Preconditions

- eCPRI interface is accessible.
- Various error scenarios (malformed, unexpected, or delayed eCPRI packets) are identified.

Execution steps

1. Attempt to force error conditions on the eCPRI interface:
    - Transmit eCPRI packets with anomalies such as invalid headers, incorrect protocol versions, unsupported message types, or corrupted payloads.
    - Introduce significant delays or slow down the network connection on the eCPRI interface.
2. Verify that the DUT detects and handles the errors appropriately, responding with informative error messages without revealing sensitive information.
3. Validate that the error messages provide helpful and actionable information for troubleshooting.
4. Restore normal connectivity.
5. Resend a normal request to the DUT through the eCPRI interface.
6. Verify that the DUT processes the request successfully and provides the expected response.

Expected Results: The DUT handles errors securely, providing meaningful error messages without disclosing sensitive information and recovering seamlessly when the connection is restored.

Expected format of evidence

- Screenshots of the error messages or status codes received from the DUT through the eCPRI interface in response to triggered errors.
- A log file documenting the requests and responses during error scenarios.

6.10.4 Void

6.10.5 eCPRI Logging and Auditing

Requirement Name: eCPRI security

Requirement Reference: REQ-SEC-SLM-APP-EVT-1, clause 5.3.8.8.11.4 in O-RAN Security Requirements and Controls Specifications [5].

Requirement Description:

Threat References: T-FRHAUL-01, T-FRHAUL-02

DUT/s: O-RU, O-DU

Test Name: TC_eCPRI_LOGGING_AUDITING

Purpose: The purpose of this test is to validate that the DUT logs relevant security events and activities on the eCPRI interface and supports auditing capabilities.

Procedure and execution steps

Preconditions

- eCPRI interface is accessible.
- Logging and auditing mechanisms are enabled and configured.

Execution steps

1. Perform various eCPRI security events. Key eCPRI security events include [26]:
    - Total count of received messages (valid and erroneous) for user and control planes
    - Messages received within the expected time window
    - Early and late arrivals of messages
    - Sequence ID errors for on-time messages
    - Messages dropped due to protocol violations or corruption
    - Timing and sequence errors in control plane messages
    - Total outbound messages for user and control planes
    - Messages discarded due to errors, resource constraints, or policy violations
    - Identified duplicated packets.
2. Verify that the DUT generates appropriate log entries for each security event, capturing relevant security-related information.
3. Access and review the generated logs to ensure they contain the necessary details for security auditing purposes.

Expected Results: The DUT generates accurate logs, recording security-related events and activities for auditing and forensic analysis.

Expected format of evidence

- The generated log files containing recorded security events and activities during the testing process.
- Screenshots of log entries highlighting relevant security events and timestamps.

6.10.6 Void

6.11 SCTP

6.11.1 Void

6.11.2 Void

6.11.3 Void

6.11.4 Void

6.11.5 SCTP DoS Prevention Rate Limiting

Requirement Name: SCTP security

Requirement Reference: REQ-SEC-TRAN-1, clause 5.3.4.1 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Ability to handle unexpected input

Threat References: T-E2-01, T-E2-02, T-E2-03

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU
Test Name: TC_SCTP_ DOS_PREVENTION_RATE_LIMITING

Purpose: The purpose of this test is to verify that the SCTP protocol effectively handles DoS attacks and prevents resource exhaustion.

Procedure and execution steps

Preconditions

- Enable DoS prevention mechanisms.
- The rate limiting parameters, such as the maximum number of connections or allowed data transfer rate, are properly defined.
- Use SCTP library.
  EXAMPLE: the sctplib library in the C programming language

Execution steps

1. Simulate a DoS attack by overwhelming the SCTP protocol with a large number of connection requests (send data at a rate that exceeds the defined rate limiting parameters).
    - EXAMPLE: Sample SCTP commands:
        - **for (int i = 0; i < num_connections; i++) {sctp_socket = sctp_socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP);**
          // Establish connections rapidly beyond system limits
          }
2. Monitor the SCTP protocol's response and behaviour during the excessive connection and data transfer attempts.

Expected Results

- The SCTP protocol detects the excessive usage and applies rate limiting measures to restrict or reject connections or data transfers that exceed the defined limits.
- The system handles the rate limiting effectively, ensuring that resources are not exhausted or overwhelmed.

Expected format of evidence

- Test logs showing successful handling of the DoS attack, such as connection limits or rejection messages.
- System performance metrics or logs indicating the proper handling of excessive connection requests.

6.11.6 SCTP Input Validation

Requirement Name: SCTP security

Requirement Reference: REQ-SEC-TRAN-1, clause 5.3.4.1 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Ability to handle unexpected input

Threat References: T-E2-01, T-E2-02, T-E2-03

DUT/s: Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_SCTP_INPUT_VALIDATION

Purpose: To verify that the SCTP protocol performs proper input validation to prevent security vulnerabilities such as buffer overflows or injection attacks.

Procedure and execution steps

Preconditions

- The SCTP protocol is configured with input validation enabled.
- Use SCTP library.
  EXAMPLE: the sctplib library in the C programming language

Execution steps

1. Attempt to establish a connection using the SCTP protocol and provide invalid or malicious input.
    - EXAMPLE: Sample SCTP command: **sctp_socket = sctp_socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP);**
2. Send data containing invalid or malicious content over the connection.
    - EXAMPLE: Sample SCTP command: **sctp_sendmsg(sctp_socket, malicious_data_buffer, data_length, NULL, 0, 0, 0, stream_id, 0, 0);**

Expected Results

- The SCTP protocol performs input validation and rejects or sanitizes the invalid or malicious input.
- The connection is not established, or the malicious data is handled safely.

Expected format of evidence

- Test logs showing the rejection or sanitization of invalid or malicious input.
- Output from the application indicating the successful validation and rejection of malicious data.

6.11.7 Void

6.11.8 Void

6.12 Transactional APIs

6.12.1 Transactional API Authentication

Requirement Name: RESTful API protection

Requirement Reference: 'REQ-SEC-O-CLOUD-NotifAPI-1, REQ-SEC-O-CLOUD-NotifAPI-2' clause 5.1.8.9.1.3 [5], 'REQ-SEC-API-1, REQ-SEC-API-2, REQ-SEC-API-3, REQ-SEC-API-4, REQ-SEC-API-5, REQ-SEC-API-6, REQ-SEC-API-8, REQ-SEC-API-9, REQ-SEC-API-10, REQ-SEC-API-13, REQ-SEC-API-15' clause 5.3.10.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: API robustness

Threat References: T-O-RAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_REST_API_AUTHENTICATION

Purpose: The purpose of this test is to verify the authentication mechanism of an O-RAN NF supporting RESTful API.

Procedure and execution steps

Preconditions

- An O-RAN NF supporting the RESTful API is provisioned and running.
- Access to the O-RAN NF management system or command-line interface.

Execution steps

1. Positive Case:

    a) Authenticate using valid credentials or API tokens:

EXAMPLE: **curl -X POST -H "Content-Type: application/json" -d '{"username":"<username>", "password":"<password>"}' http://<ORAN_IP>/auth**

    b) Capture the authentication token from the response.

    c) Execute an authenticated request against an O-RAN NF resource (e.g., get cell status).

    d) Verify that the request is successful and returns the expected response.

2. Negative Case:

Attempt to access the O-RAN RESTful API without providing valid authentication credentials:
curl http://<ORAN_IP>/cell-status

    a) Verify that the request fails and returns an unauthorized response.

Expected Results

1. Positive Case:
    - Authentication using valid credentials or API tokens is successful.
    - Authorized requests to O-RAN NF resources return the expected responses.
2. Negative Case:
    - Requests without valid authentication credentials are rejected with an unauthorized response.

Expected format of evidence

- Screenshots or logs showing the successful authentication and authorized requests.
- Screenshots or logs showing the failed authentication attempts.

6.12.2 Transactional API Authorization and Access Control

Requirement Name: RESTful API protection

Requirement Reference: ‘REQ-SEC-O-CLOUD-NotifAPI-1, REQ-SEC-O-CLOUD-NotifAPI-2’ clause 5.1.8.9.1.3 [5], ‘REQ-SEC-API-1, REQ-SEC-API-2, REQ-SEC-API-3, REQ-SEC-API-4, REQ-SEC-API-5, REQ-SEC-API-6, REQ-SEC-API-8, REQ-SEC-API-9, REQ-SEC-API-10, REQ-SEC-API-13, REQ-SEC-API-15’ clause 5.3.10.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: API robustness

Threat References: T-O-RAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_REST_AUTHORIZATION_ACCESS_CONTROL

Purpose: The purpose of this test is to ensure that the RESTful API enforces proper authorization and access control mechanisms.

Procedure and execution steps

Preconditions

- An O-RAN NF supporting the RESTful API is provisioned and running.
- Access to the O-RAN NF management system or command-line interface.
- User roles and permissions are defined and configured.

Execution steps

1. Positive Case:
    a) Authenticate using credentials associated with a user assigned to a role with necessary permissions:

EXAMPLE: **curl -X POST -H "Content-Type: application/json" -d '{"username":"<username>", "password":"<password>"}' http://<ORAN_IP>/auth**

    b) Capture the authentication token from the response.
    c) Execute a request that requires the permissions granted by the user's role (e.g., update configuration).
    d) Verify that the request is successful and returns the expected response.
2. Negative Case:
    a) Authenticate using credentials associated with a user not assigned to a role with necessary permissions:

EXAMPLE: **curl -X POST -H "Content-Type: application/json" -d '{"username":"<username>", "password":"<password>"}' http://<ORAN_IP>/auth**

    b) Capture the authentication token from the response.
    c) Execute a request that requires the permissions beyond the user's role (e.g., perform a restricted operation).
    d) Verify that the request fails and returns a forbidden response.

Expected Results

1. Positive Case:
    - Users with appropriate roles and permissions can perform authorized actions.
    - Requests requiring specific permissions return the expected responses.
2. Negative Case:
    - Users without necessary roles or permissions are restricted from performing unauthorized actions.
    - Requests requiring permissions beyond the user's role return a forbidden response.

Expected format of evidence

- Screenshots or logs showing the successful authorization and access control enforcement.
- Screenshots or logs showing the failed authorization attempts.

6.12.3 Transactional API Input Validation and Sanitization

Requirement Name: RESTful API protection

Requirement Reference: 'REQ-SEC-O-CLOUD-NotifAPI-1, REQ-SEC-O-CLOUD-NotifAPI-2' clause 5.1.8.9.1.3 [5], 'REQ-SEC-API-1, REQ-SEC-API-2, REQ-SEC-API-3, REQ-SEC-API-4, REQ-SEC-API-5, REQ-SEC-API-6, REQ-SEC-API-8, REQ-SEC-API-9, REQ-SEC-API-10, REQ-SEC-API-13, REQ-SEC-API-15' clause 5.3.10.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: API robustness

Threat References: T-O-RAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_REST_INPUT_VALIDATION_SANITIZATION

Purpose: The purpose of this test is to validate that the RESTful API properly validates and sanitizes input data to prevent common security vulnerabilities.

Procedure and execution steps

Preconditions

- An O-RAN NF supporting the RESTful API is provisioned and running.
- Access to the O-RAN NF management system or command-line interface.

Execution steps

1. Positive Case:
    a. Construct a valid request with appropriate input data:

    EXAMPLE: curl -X POST -H "Content-Type: application/json" -d '{"parameter1":"value1", "parameter2":"value2"}' http://<ORAN_IP>/api-endpoint

    b. Verify that the request is successful and returns the expected response.
2. Negative Case:
    a. Construct a request with invalid or malicious input data:

    EXAMPLE: curl -X POST -H "Content-Type: application/json" -d '{"parameter1":"<script>alert(1)</script>", "parameter2":"value2"}' http://<ORAN_IP>/api-endpoint

    b. Verify that the request fails and returns an error response or rejects the malicious input.

Expected Results

1. Positive Case:
    a. Requests with valid and appropriate input data are successfully processed.
    b. Responses from the O-RAN NF RESTful API are as expected.
2. Negative Case:
    a. Requests with invalid or malicious input data are rejected or handled properly to prevent security vulnerabilities.

Expected format of evidence

- Screenshots or logs showing the successful input validation and sanitization.
- Screenshots or logs showing failed input validation or sanitization attempts.

6.12.4 Transactional API Security Logging and Monitoring

Requirement Name: RESTful API protection

Requirement Reference: 'REQ-SEC-O-CLOUD-NotifAPI-1, REQ-SEC-O-CLOUD-NotifAPI-2' clause 5.1.8.9.1.3 [5], 'REQ-SEC-API-1, REQ-SEC-API-2, REQ-SEC-API-3, REQ-SEC-API-4, REQ-SEC-API-5, REQ-SEC-API-6, REQ-SEC-API-8, REQ-SEC-API-9, REQ-SEC-API-10, REQ-SEC-API-13, REQ-SEC-API-15' clause 5.3.10.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: API robustness

Threat References: T-O-RAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_REST_SECURITY_LOGGING_MONITORING

Purpose: The purpose of this test is to verify that the O-RAN NF logs and monitors API activities for security and compliance purposes.

Procedure and execution steps

Preconditions

- An O-RAN NF supporting the RESTful API is provisioned and running.
- Access to the O-RAN NF management system or command-line interface.

Execution steps

1. Positive Case:
    a) Enable API logging and monitoring for the O-RAN NF.
    b) Generate a series of API requests and actions.
    c) Review the logs or monitoring system for the recorded activities.
2. Negative Case:
    a) Attempt unauthorized API actions or exploit security vulnerabilities.
    b) Verify that the logs or monitoring system captures and raises alerts for these activities.

Expected Results

1. Positive Case:
    - API activities are logged and monitored by the O-RAN NF.
    - Logs or monitoring system records the expected API requests and actions.
2. Negative Case:
    - Unauthorized or malicious API actions trigger alerts in the logs or monitoring system.
    - Logs or monitoring system captures and records failed security attempts.

Expected format of evidence

Screenshots or logs from the O-RAN NF management system showing the successful or failed API logging and monitoring settings.

7 Common Network Security Tests for O-RAN architecture elements

7.1 Overview

This clause contains a set of security evaluations that are performed from outside and inside of the network function in a network capacity. It is used to measure the external exposure and risk(s) of the function in place and leverages common techniques used in cyber security to evaluate the risk(s) device under test faces or has.

The objects in scope of these network-based security tests are SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU and O-Cloud.

7.2 Network Protocol and Service Enumeration

7.2.1 Network Protocol and Service Enumeration

Requirement Name: Network protocol and service enumeration

Requirement Reference: REQ-SEC-NET-1, clause 5.3.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Protocols and services are documented by vendor. No undocumented protocols and services are offered.

Threat References: T-O-RAN-01, T-O-RAN-02

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test name: TC_Network_Procotol_And_Enumeration

Purpose: To verify that the list of active network protocols and services on DUT is in line with vendor-provided list of network protocols and services supported. Probing of network protocols and services on DUT provides the information whether the service is active or not. This test case probes all possible TCP and SCTP ports in range 0-65535 using port scanner for presence of the active services.

This test case probes all documented UDP ports from vendor-provided list using port scanner for presence of the active services. Optionally, additional UDP ports may be scanned as well.

NOTE 1: In practice, such probing is often referred to as network scanning or port scanning.

NOTE 2: In practice, services may also run on ports different from ports defined in [i.1] .

Procedure and execution steps

Preconditions

- Port scanner with capabilities as defined in clause 5.3 of present document.
- Network access to DUT
- Vendor-provided list of network protocols and services supported by DUT

Execution steps

1. List of open ports are determined as follows:
    - Port scanner scans all TCP ports in range 0-65535 on the IP interface of DUT. TCP SYN/ACK response by DUT are interpreted as open port.

- Port scanner scans all SCTP ports in range 0-65535 on the IP interface of DUT. SCTP INIT-ACK response by DUT are interpreted as open port.
- All UDP ports documented in vendor-provided list are interpreted as open ports. Other UDP ports may be considered as open for the purpose of service detection.

NOTE 3: Due to the nature of UDP protocol, there is no simple method of open port detection similar to TCP/SCTP methods based on analysis of response message type (TCP: SYN/ACK, SCTP: INIT-ACK). In case of UDP, open port detection inevitably relies on service detection which is discussed in step 2 of this test procedure. In practice, port scans of entire UDP port range 0-65535 are impractical and time consuming. Typically, service detection is performed only for subset of UDP ports. UDP port subset selection is arbitrary and not standardized. Service detection in this test procedure is required for UDP ports from vendor-provided list and is optional for other UDP ports.

2. For each open port from previous step, port scanner performs service detection by sending service probe(s) as follows:
   - If open port is listed in vendor-provided list, port scanner uses service probe from its built-in database that exactly matches service documented in vendor-provided list.
   - If open port is not listed in vendor-provided list, port scanner uses service probe from its built-in database that exactly matches service defined in [i.1] for the that open port. If such service is not defined in [i.1] , port scanner may report service as "unknown". Alternatively, port scanner may perform further service detection attempts based on other service probes from its built-in database.

NOTE 4: Service detection for open ports that are also listed in vendor-provided list requires only one probe. However, service information can be helpful in discussion with DUT vendor. This test procedure therefore accommodates optional service detection based on one probe or multiple probes.

3. Port scanner produces list of detected active network protocols, ports and services on DUT.

Expected results

Comparison between the vendor-provided list of all supported network protocols and services and the list or active network protocols and services found by port scanner is performed.

All services found by port scanner are documented in vendor-provided list. This test case ends with success if:

- both lists match exactly
- list of network protocols and services found by port scanner has fewer items than vendor-provided list; all items found by port scanner exactly match items from vendor-provided list.

If any service is found by port scanner and it is not documented in vendor-provided list, this test case shall fail. It means that vendor-provided list is incorrect and undocumented attack surface exists.

Expected format of evidence: Result of probing the DUT is a list of active network protocols and services. Each item contains network protocol (TCP, UDP, SCTP), port number (from range 0-65535) and service name. If service type cannot be determined during probing, service name is "unknown".

Service name is in line with Service Name and Transport Protocol Port Number Registry defined by IANA [i.1] . If service name is not defined in [i.1] , vendor provided service name is used.

7.3 Password-Based Authentication

7.3.1 Password guessing

Requirement Name: Password-Based Authentication

Requirement Reference: SEC-CTL-PASS-2, clause 5.5.7.2, REQ-SEC-PASS-1, clause 5.3.7.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Password guessing protection mechanism is present on the DUT

Threat References: T-O-RAN-02, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Password_Guessing

Purpose: To verify that DUT has protection mechanism(s) implemented to prevent password guessing attacks against services using password-based authentication. . Simulation of password guessing attacks against services on DUT provides the information whether any protection mechanism is present.

This test case is run against all services on DUT that use password-based authentication. Vendor-provided list of all supported network protocols and services are used as a source.

NOTE 1: Vendor-provided list of all supported network protocols and services may not include the specific information about presence of password-based authentication as it is including network protocol, port and service name. In practice, only subset of services from vendor-provided list will use password-based authentication.

Procedure and execution steps

Preconditions

- Valid username for each tested service. This test case does not mandate any specific list of passwords to be used for testing.
- Network access to DUT
- Physical access to DUT (applicable if the DUT is in physical form)
- Vendor-provided list of network protocols and services supported by DUT
- Number of authentication attempts configured in account lock-out policy

Execution steps

1. List of services using password-based authentication is determined by analyzing the vendor-provided list as well as by analyzing local services that are not remotely accessible.
2. For services identified in the previous step, presence of protection mechanism is tested as follows:
    - combination of valid username and invalid password (or various invalid passwords) are used for authentication repeatedly.
    - after certain number of authentication attempts, protection mechanism of DUT are detected.
    - minimum number of authentication attempts is the calculated as (number configured in the lock-out policy + 1) attempts .

Expected results

In context of each of the services using password-based authentication, protection mechanism(s) is present. Applicable to local services and to remotely accessible services.

NOTE 2: In practice, brute-forcing and dictionary attacks are the most common classes of password guessing attacks. Traditional approach to brute-forcing and dictionary attacks uses fixed username with various candidate passwords. Password spraying is another approach that can be combined with brute-forcing and dictionary attacks; fixed password is tested with various candidate usernames. Example of protection mechanism is enforcing delay before next authentication attempt(s) by the same client. This test case cannot list all possible techniques that protection mechanisms can use. However, following list provides overview of the most common approaches:

- Increase the delay after each unsuccessful authentication attempt.
- Implement challenge-response authentication (example of such measure: CAPTCHA)
- In order to prevent more attempts, impose temporary lock out on the client when threshold of consecutive failed authentication attempts is reached. During defined period of time all authentication attempts by locked-out client are rejected.

EXAMPLE: Lock-out policy is configured for 10 invalid authentication attempts. If DUT uses protection mechanism based on delaying authentication attempts, such delay is observed when DUT receives 11th consecutive invalid authentication attempt.

This test case fails if one or more services using password-based authentication have no protection mechanism present.

Expected format of evidence: Report file, log files and/or screenshots.

7.3.2 Unauthorized Password Reset

Requirement Name: Password-Based Authentication

Requirement Reference: REQ-SEC-PASS-1, clause 5.3.7.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Out-of-band password recovery mechanism absent or deactivated on DUT

Threat References: T-O-RAN-02, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Unauthorized_Password_Reset

Purpose: To verify that password reset mechanism DUT cannot be circumvented, disabled, or misused to gain access to DUT, its configuration, and data. Test covers services using password-based authentication and out-of-band mechanisms of password reset present in DUT in physical form.

Procedure and execution steps

Preconditions

- Network access to DUT
- Physical access to DUT (applicable if the DUT is in physical form)
- Vendor-provided list of network protocols and services supported by DUT

Execution steps

List of services using password-based authentication are determined by analyzing the vendor-provided list as well as by analyzing local services that are not remotely accessible.

1. For services identified in the previous step, presence of password reset is tested.
2. For DUT that has physical form, it verifies that use of hardware factory reset switch or switches results in factory reset. Using any out-of-band mechanism, it is not possible to reset password only. If password reset is required, factory reset of O-RAN component is performed. Factory reset wipes O-RAN component, its configuration and data.

Expected results

In context of each of the services using password-based authentication, no password change mechanism is present. Applicable to local services and to remotely accessible services.

This test case fails if one or more services using password-based authentication have password reset mechanism exposed.

This test case fails if DUT in physical form has hardware switch or switches that can be used to reset password without triggering factory reset of DUT.

Expected format of evidence: Report file, log files and/or screenshots.

7.3.3 Password Policy Enforcement

Requirement Name: Password-Based Authentication

Requirement Reference: REQ-SEC-PASS-1, clause 5.3.7.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Secure password policy is supported and enforced on the DUT

Threat References: T-O-RAN-02, T-O-RAN-03, T-O-RAN-05, T-O-RAN-06

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Password_Policy_Enforcement

Purpose: To verify that password policy applied for services using password-based authentication is effectively enforced by DUT.

Procedure and execution steps

Preconditions

- Set of valid username and valid password for each tested service
- Network access to DUT
- Physical access to DUT (applicable if the DUT is in physical form)
- Vendor-provided list of network protocols and services supported by DUT
- Password policy that is actually applied on the DUT

Execution steps

1. List of services using password-based authentication is determined by analyzing the vendor-provided list as well as by analyzing local services that are not remotely accessible.
2. For services identified in the previous step, effectiveness of password policy enforcement is verified as follows:
    - combination of valid username and valid password are used to authenticate
    - password change is performed using password that does not conform to applied password policy

EXAMPLE: DUT uses password policy to set rules for password length, type of characters used (allowed and disallowed characters), complexity (character groups), and denied passwords (deny-list of passwords that cannot be set). Candidate password that does not conform to rules are chosen for this test. As password policy may be complex set of rules, multiple candidate passwords are tested to fully cover possible password policy violations.

Expected results

In context of each of the services using password-based authentication, applied password policy are effectively enforced and non-compliant passwords are rejected by DUT during password change. Applicable to local services and to remotely accessible services.

Expected format of evidence: Report file, log files and/or screenshots.

7.4 Network Protocol Fuzzing

Fuzzing is an automated process of sending invalid or random inputs to a SUT to cause it to malfunction or crash.

Fuzzing is effective for finding vulnerabilities because while most modern programs have extensive input fields, the test coverage of these areas is relatively small. Even though this process can be a powerful capability to ensure robustness, it needs to be sufficiently defined and implemented throughout the system development lifecycle to be helpful and achieve the required results in a multi-vendor environment.

While traditional fuzzing techniques involve fuzzing piece(s) of software and generating inputs through command line or input files, fuzzing telecommunication network protocols tends to be different, requiring sending information via network ports. Furthermore, the complex nature of network protocols in the SUT resulting from how they are layered over each other adds to the challenges of fuzzing such SUTs.

The following are examples of the protocols that fuzzing will cover;

General Transport Protocols
SCTP
IP
TCP
UDP
SSH
HTTP
HTTP/2

and

O-RAN Specific Protocols
NETCONF
E1AP
E2AP
A1
CTI
eCPRI
PTP

It is anticipated that many O-RAN architecture elements utilize common software frameworks used for the lower-level general communication. In this case it should be evaluated if these General Transport Protocols are being tested in extensive Fuzzing tests in other activities and can therefore be considered to have lower risk profiles compared to the O-RAN Specific Protocols with less testing in the general industry.

Many of the O-RAN specific protocols are state- machine based protocols that can have multiple end points served at the same time, e.g. the protocol needs to be tested in scale to understand if possible memory leaks or other similar aspects is available that could lead to buffer overflows (opening up for possible code execution) or software crashes of the O-RAN specific software.

Fuzzing on the M-Plane protocol inside the Configuration of the O-RAN Fronthaul can be a possible significant area as this is combining multiple technologies from many domains into a single solution. In order for the Fuzzing to be time and resource efficient, it is important that this Fuzzing is protocol and state machine aware so that the Fuzzing can focus on the relevant aspects of the SUT representing the most significant risk exposure. Further effectiveness can be achieved if the Fuzzing capability is able to intelligently respond to the SUT behaviour. The Fuzzing tool should be able to both perform test with and without access to relevant credentials. Many possible vulnerabilities would be present on the inside of the authenticated session of the management protocols and would lead to escalation of privileges.

In order to identify the possible risk for memory leaks, Denial of Service (DoS) or other similar aspects a robust logging of the underlying platform (hardware and software), the virtualization or container platform and the O-RAN function, the logging needs to be detailed enough to evaluate the trends early but not intrusive to degrade the performance of the platform and lead to inaccurate results.

As general guidance, vendors and operators running fuzzing tests aim to document the list of all of the protocols of the SUTs reachable externally on an IP-based interface, together with indications of whether adequate available robustness and fuzz testing tools have been used against them. The tool's name, their unambiguous version (also for plug-ins if applicable), user settings, and the relevant output evidenced and should be documented. Additionally, any input causing unspecified, undocumented, or unexpected behaviour and a description of this behaviour should be highlighted in the testing documentation.

Since fuzzing test cases are not exhaustive and difficult to define and replicate, it's likely that test results even from testing the same set of protocols by different vendors may end up resulting in different outputs. So further effort and time needs to be invested in fuzzing activities until a satisfactory approach based on the vendor's or/and operators adopted risk-based model is satisfied.

7.5 Denial of Service/Message Flooding

7.5.1 Protocol, Application and Volumetric Based DDoS Attacks

Requirement Name: Robustness against Volumetric DDoS Attack

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: “An O-RAN component with external network interface shall be able to withstand network transport protocol based volumetric DDoS attack without system crash and returning to normal service level after the attack”

Threat References: T-O-RAN-04, T-O-RAN-09, T-SMO-03

DUT/s: Components implementing O-RAN interfaces defined in clause 5.1 of this document.

Test Name: TC_Robustness_DDoS

Purpose: To verify the DUT is able to recover from a DDoS attack.

Procedure and execution steps

Each component interface is tested to validate how handling of large amounts of requests is done, similar to what is seen from denial of service (DoS) or/and distributed denial of service (DDoS) attempts. DoS/DDoS scenario can be occurred as a result of malicious attack or because of network/operator error. DoS/DDoS attacks may come in these forms: Protocol layer attacks (e.g. SYN Floods, UDP Floods, TCP Floods), Volume based attacks (e.g. ICMP floods, Smurf DDoS) and Application layer attacks (e.g. GET/POST floods, low-and-slow attacks, attacks that target specific software – application with exposed network services or operating system network services).

Preconditions

This test is executed against running O-RAN component or O-RAN system as the DUT.

Test prerequisites:

- Network access to DUT
- Vendor-provided list of network protocols and services supported by DUT

Execution steps

1. In case the call flow needs authentication:
    1.1. Set up a call flow that will send repeated requests after the authentication at an increasing rate over time. Mark the failure point of receiving rejection or response messages.
    1.2. Stop the attack.
    1.3. Set up a call flow that will send repeated requests before the authentication at an increasing rate over time. Mark the failure point of receiving acceptance or response messages.
    1.4. Stop the attack.
2. In case the call flow does not need authentication:
    2.1. Set up a call flow that will send repeated requests at an increasing rate over time. Mark the failure point of receiving response messages.
    2.2. Stop the attack.

Expected results

It is expected the component fails to serve requests after step 1.1, 1.3 and/or step 2.1.

After the attack/test stops, the DUT returns to a functional state, being able to respond to service requests again.

This test case fails if DUT does not return to a functional state after the test stops.

Expected format of evidence: Report file, log files and/or screenshots.

7.5.2 Void

7.5.3 Void

7.5.4 Void

7.5.5 Near-RT RIC A1 interface DoS/DDoS protection and recovery

Requirement Name: Near-RT RIC DoS/DDoS protection and recovery

Requirement Reference: REQ-SEC-NEAR-RT-6, REQ-SEC-NEAR-RT-7, clause 5.1.3, REQ-SEC-DOS-1, clause 5.3.5 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-04, T-O-RAN-09

DUT/s: Near-RT RIC

Test Name: TC_DoS_RECOV_NEAR_RT_RIC

Purpose: The purpose of this test is to evaluate the resilience of the Near-RT RIC against Denial-of-Service/Distributed-Denial-of-Service attacks on the A1 interface and the recovery process from those attacks.

Procedure and execution steps

Preconditions

- The Near-RT RIC is powered on and operational.
- DoS/DDoS protection mechanisms are implemented on the Near-RT RIC.
- The testing environment is isolated and does not impact production systems.
- Refer to TC_Robustness_DDoS.

Execution steps

1. In case the call flow needs authentication:
    1.1. Set up a call flow that will send repeated requests to the exposed A1 interface end point after the authentication at an increasing rate over time. Mark the failure point of receiving rejection or response messages.
    1.2. Stop the attack.
    1.3. Set up a call flow that will send repeated requests to the exposed A1 interface end point before the authentication at an increasing rate over time. Mark the failure point of receiving acceptance or response messages.

1.4. Stop the attack.

2. In case the call flow does not need authentication:

   2.1. Set up a call flow that will send repeated requests to the exposed A1 interface end point at an increasing rate over time. Mark the failure point of receiving response messages.

   2.2. Stop the attack.

Expected Results

- Near-RT RIC detects and demonstrates robustness against the DoS/DDoS attack on A1 interface, maintaining normal operations with acceptable performance and rejecting requests, regardless of whether they are malicious or not.
- Near-RT RIC successfully recovers from the DoS/DDoS attack on A1 interface and resumes normal operation within a reasonable recovery time.

Expected format of evidence:

- Observation logs during the DoS/DDoS attack, including any triggered countermeasures or rate limiting mechanisms, and validate that the Near-RT RIC effectively defends against the attack.
- Observation logs of the recovery process, including the time taken for the Near-RT RIC to regain stable operation, and validate that the recovery is timely and effective.

  NOTE: Recovery time specifies the maximum acceptable recovery time after the attack ceases (e.g., "Near-RT RIC recovers and returns to normal operation within 5 minutes after the attack stops").

7.6 Input validation and error handling

Input validation and error handling are pivotal security practices that guard against malformed or malicious data inputs, ensuring that systems behave predictably and securely. This clause elucidates a series of tests designed to validate the efficacy of the input validation and error handling mechanisms implemented in various O-RAN network functions (O-CU, O-DU, Near-RT RIC), safeguarding them from a myriad of potential vulnerabilities and ensuring robust, secure, and stable operations.

7.6.1 O-CU input validation and error handling

Requirement Name: Input validation and error handling on data provided through O1 and E2 interfaces.

Requirement Reference: REQ-SEC-OCU-1, clause 5.1.4 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-05

DUT/s: O-CU

Test Name: TC_INPUT_VALIDATION_ERR_HANDL_OCU

Purpose: The purpose of this test is to verify that the O-CU performs proper input validation on provided data via E2/O1 interfaces and rejects invalid or malicious inputs. It verifies that the O-CU correctly handles errors and responds appropriately.

Procedure and execution steps

Preconditions

- The O-CU is powered on and operational.

- Test environment is set up with E2 and O1 interfaces configured.
- Input validation mechanisms are implemented on O-CU.
- Error handling mechanisms (e.g., error codes, error messages) are implemented by O-CU.

Execution steps

1. Case of malformed input data
    a) The tester provides invalid or malformed input data to the O-CU via E2/O1 interfaces, violating the specified format or containing unexpected values.
    b) The tester captures and analyses the response from the E2/O1 interfaces.
    c) The tester verifies that the O-CU detects the invalid input and rejects it appropriately, returning an error message or taking necessary actions to mitigate the impact.

EXAMPLE: actions could be rejecting the message, sending an error indication, etc.

2. Case of malicious input data
    a) The tester provides malicious input data to the O-CU, aiming to exploit known vulnerabilities (e.g., CVE database, OWASP Top Ten, NIST National Vulnerability Database (NVD), vendor-specific vulnerability database) or perform unauthorized actions.
    b) The tester verifies that the O-CU identifies the malicious input and implements security measures to prevent exploitation, such as input sanitization, access controls, or anomaly detection.
3. Boundary case
    a) Provide input data at the boundaries of the allowed range or limits defined for specific inputs.
    b) Verify that the O-CU handles the boundary cases correctly, without encountering any unexpected behaviour or errors due to boundary conditions.

Expected Results

1. For case ‘malformed input data’, the O-CU properly validates incoming inputs form O1/E2 interfaces and rejects those with invalid or malformed data, returning an appropriate error response and preventing any potential security risks or system failures.
2. For case ‘malicious input data’, the O-CU detects and mitigates the malicious input, preventing any potential security breaches or unauthorized operations.
3. For case ‘boundary’, the O-CU properly handles the boundary cases, ensuring that inputs at the limits are processed accurately without causing any system instability or vulnerabilities.

Expected format of evidence:

1. Logs detailing the invalid or malformed input data provided to the O-CU via O1/E2 interfaces, alongside system logs capturing the O-CU's error messages or indications in response to the invalid input.
2. Logs documenting the malicious input data sent to the O-CU and the targeted vulnerabilities, complemented by system logs highlighting the O-CU 's detection and mitigation actions upon receiving the malicious input.
3. Logs of the boundary input data values provided to the O-CU, paired with system logs capturing the O-CU 's messages or behaviours in response to the boundary inputs.

7.6.2 O-DU input validation and error handling

Requirement Name: Input validation and error handling on data provided through O1/E2/FH interfaces.

Requirement Reference: REQ-SEC-ODU-1, clause 5.1.5 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-05

DUT/s: O-DU

Test Name: TC_INPUT_VALIDATION_ERR_HANDL_ODU

Purpose: The purpose of this test is to verify that the O-DU performs proper input validation on provided data via E2/O1/FH interfaces and rejects invalid or malicious inputs. It verifies that the O-DU correctly handles errors and responds appropriately.

Procedure and execution steps

Preconditions

- The O-DU is powered on and operational.
- Test environment is set up with E2/O1/FH interfaces configured.
- Input validation mechanisms are implemented on O-DU.
- Error handling mechanisms (e.g., error codes, error messages) are implemented by O-DU.

Execution steps

1. Case of malformed input data

    a) The tester provides invalid or malformed input data to the O-DU via E2/O1/FH interfaces, violating the specified format or containing unexpected values.

    b) The tester captures and analyses the response from the E2/O1/FH interfaces.

    c) The tester verifies that the O-DU detects the invalid input and rejects it appropriately, returning an error message or taking necessary actions to mitigate the impact.

EXAMPLE: actions could be rejecting the message, sending an error indication, etc.

2. Case of malicious input data

    a) The tester provides malicious input data to the O-DU, aiming to exploit known vulnerabilities (e.g., CVE database, OWASP Top Ten, NIST National Vulnerability Database (NVD), vendor-specific vulnerability database) or perform unauthorized actions.

    b) The tester verifies that the O-DU identifies the malicious input and implements security measures to prevent exploitation, such as input sanitization, access controls, or anomaly detection.

3. Boundary case

    a) Provide input data at the boundaries of the allowed range or limits defined for specific inputs.

    b) Verify that the O-DU handles the boundary cases correctly, without encountering any unexpected behaviour or errors due to boundary conditions.

Expected Results

1. For case ‘malformed input data’, the O-DU properly validates incoming inputs form O1/E2/FH interfaces and rejects those with invalid or malformed data, returning an appropriate error response and preventing any potential security risks or system failures.
2. For case ‘malicious input data’, the O-DU detects and mitigates the malicious input, preventing any potential security breaches or unauthorized operations.

3. For case ‘boundary’, the O-DU properly handles the boundary cases, ensuring that inputs at the limits are processed accurately without causing any system instability or vulnerabilities.

Expected format of evidence:

1. Logs detailing the invalid or malformed input data provided to the O-DU via E2/O1/FH interfaces, alongside system logs capturing the O-DU's error messages or indications in response to the invalid input.
2. Logs documenting the malicious input data sent to the O-DU and the targeted vulnerabilities, complemented by system logs highlighting the O-DU's detection and mitigation actions upon receiving the malicious input.
3. Logs of the boundary input data values provided to the O-DU, paired with system logs capturing the O-DU's messages or behaviours in response to the boundary inputs.

7.6.3 Near-RT RIC input validation and error handling

Requirement Name: Error handling by Near-RT RIC

Requirement Reference: REQ-SEC-NEAR-RT-6, REQ-SEC-NEAR-RT-7, clause 5.1.3 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: NEAR-RT RIC

Test Name: TC_INPUT_VALIDATION_ERR_HANDL_NEAR_RT_RIC

Purpose: The purpose of this test is to verify that the Near-RT RIC performs proper input validation on provided data via O1/E2/A1/Y1 interfaces and rejects invalid or malicious inputs. It verifies that the Near-RT RIC correctly handles errors and responds appropriately.

Procedure and execution steps

Preconditions

1. Near-RT RIC is powered and operational.
2. Test environment is set up with O1/E2/A1/Y1 interfaces configured.
3. Input validation mechanisms are implemented on Near-RT RIC.
4. Error handling mechanisms (e.g., error codes, error messages) are implemented by Near-RT RIC.

Execution steps

1. Case of malformed input data

    a) The tester provides invalid or malformed input data to the Near-RT RIC via O1/E2/A1/Y1interfaces, violating the specified format or containing unexpected values.

    b) The tester captures and analyses the response from the O1/E2/A1/Y1interfaces.

    c) The tester verifies that the Near-RT RIC detects the invalid input and rejects it appropriately, returning an error message or taking necessary actions to mitigate the impact.

EXAMPLE: actions could be rejecting the message, sending an error indication, etc.

2. Case of malicious input data

    a) The tester provides malicious input data to the Near-RT RIC, aiming to exploit known vulnerabilities (e.g., CVE database, OWASP Top Ten, NIST National Vulnerability Database (NVD), vendor-specific vulnerability database) or perform unauthorized actions.

b) The tester verifies that the Near-RT RIC identifies the malicious input and implements security measures to prevent exploitation, such as input sanitization, access controls, or anomaly detection.

3. Boundary case

a) Provide input data at the boundaries of the allowed range or limits defined for specific inputs.

b) Verify that the Near-RT RIC handles the boundary cases correctly, without encountering any unexpected behaviour or errors due to boundary conditions.

Expected Results

1. For case ‘malformed input data’, the Near-RT RIC properly validates incoming inputs form O1/E2/A1/Y1interfaces and rejects those with invalid or malformed data, returning an appropriate error response and preventing any potential security risks or system failures.
2. For case ‘malicious input data’, the Near-RT RIC detects and mitigates the malicious input, preventing any potential security breaches or unauthorized operations.
3. For case ‘boundary’, the Near-RT RIC properly handles the boundary cases, ensuring that inputs at the limits are processed accurately without causing any system instability or vulnerabilities.

Expected format of evidence:

1. Logs detailing the invalid or malformed input data provided to the Near-RT RIC via O1/E2/A1/Y1 interfaces, alongside system logs capturing the Near-RT RIC's error messages or indications in response to the invalid input.
2. Logs documenting the malicious input data sent to the Near-RT RIC and the targeted vulnerabilities, complemented by system logs highlighting the Near-RT RIC 's detection and mitigation actions upon receiving the malicious input.
3. Logs of the boundary input data values provided to the Near-RT RIC, paired with system logs capturing the Near-RT RIC 's messages or behaviours in response to the boundary inputs.

7.6.4 Near-RT RIC input validation and error handling of data received from xApp

Requirement Name: Error handling by Near-RT RIC

Requirement Reference: SEC-CTL-NEAR-RT-18, clause 5.1.3.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: The Near-RT RIC shall verify data received through E2 related APIs initiated by xApp as follows:

The data values are valid.

The Near-RT RIC shall log security event(s) if any of the verification steps fail.

Threat References: T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: NEAR-RT RIC

Test Name: TC_E2API_INPUT_VALIDATION_ERR_HANDL_NEAR_RT_RIC_XAPP

Purpose: The purpose of this test is to verify that the Near-RT RIC performs proper input validation on received data from xApp through E2 related APIs and rejects malformed inputs. It verifies that the Near-RT RIC correctly handles errors and responds appropriately.

Procedure and execution steps

Preconditions:

- Near-RT RIC is deployed.
- A testing software that supports communication with Near-RT RIC over E2 related APIs [27].
- Error handling mechanisms are implemented by Near-RT RIC.
  EXAMPLE: E2 Error or Failure message with Cause as defined in clause 9.2.1 of [28].

Execution Steps:

1. Malformed input data
   a. Provide malformed input data to the Near-RT RIC via E2 related APIs, violating the specified format.

   EXAMPLE: Input data containing unknown IEs, incorrect format and mandatory IEs not present.

   b. Capture and analyze the response from Near-RT RIC via E2 related APIs.
   c. Verify that the Near-RT RIC detects the malformed input and rejects it appropriately, returning an error message or taking necessary actions to mitigate the impact.

   EXAMPLE: Actions could be rejecting the message, sending an error rejection or failure indication, etc.

2. Valid and Invalid boundary cases
   a. Provide invalid (wrong value) data outside of the boundaries or limits of the acceptable values or ranges defined for specific parameters in clause 5.2.2 & 5.2.3, O-RAN E2 Interface Test Specification [29] for E2 related APIs.
   b. Provide valid data at the boundaries or inside the allowed ranges defined for specific parameters in clause 5.2.2 & 5.2.3, O-RAN E2 Interface Test Specification [29] for E2 related APIs.
   c. Verify that the Near-RT RIC handles these boundary cases correctly, without encountering any unexpected behaviour or errors due to boundary conditions.

Expected Results:

1. Malformed input data
   a. The Near-RT RIC properly validates incoming inputs via E2 related APIs from xApp and rejects those with malformed data, returning an appropriate error response.
2. Valid and Invalid boundary cases
   a. The Near-RT RIC properly handles the valid and invalid boundary cases.

Expected format of evidence:

1. Malformed input data
   a. Logs detailing the malformed input data provided to the Near-RT RIC via E2 related APIs, alongside system logs capturing the Near-RT RIC's error messages.
2. Valid and Invalid boundary cases
   a. Logs of the input data values provided to the Near-RT RIC, paired with system logs capturing the Near-RT RIC 's messages or behaviours in response to the boundary inputs.

7.7 Secure configuration enforcement

Requirement Name: O-CU configuration security enforcement

Requirement Reference: SEC-CTL-OCU-1, clause 5.1.4.2, SEC-CTL-ODU-1, clause 5.1.5.2, ,REQ-SEC-ORU-1, clause 5.1.6.1 in O-RAN Security Requirements and Controls Specifications [5], 3GPP TS 33.501 clause 5.3.4 [25]

Requirement Description: The DUT ensures that settings and software configurations are protected from unauthorized modifications.

Threat Reference: T-O-RAN-02

DUT/s: O-CU, O-DU, O-RU

Test Name: TC_CONF_ENFORCEMENT

Purpose: Ensure the DUT prevents unauthorized settings and software configurations changes, as required by 3GPP TS 33.501 clause 5.3.4 [25].

Procedure and execution steps

Preconditions:

1. The DUT is powered on and operational.
2. Settings and software configurations are defined and applied on the DUT.
3. Successful access to the DUT is established.

Execution steps:

1. Attempt to modify the DUT settings and software configurations with proper authorization.
2. Verify that the DUT detects and accepts any authorized modification attempts.
3. Attempt to modify the DUT settings and software configurations without proper authorization.
4. Verify that the DUT detects and rejects any unauthorized modification attempts.

Expected results:

- DUT accepts authorized modifications.
- DUT detects and rejects any unauthorized modifications.

Expected format of evidence:

- Logs indicating the detection and acceptance of authorized modifications.
- Logs indicating the detection and rejection of unauthorized modifications.

EXAMPLE: examples of settings and software configurations include:

Security configuration: e.g., protocols and keys used for authentication, authorization, and secure communication.

Software management: e.g., current software version

Interface settings

7.8 Logging and monitoring

The tests outlined here aim to scrutinize the logging and monitoring capabilities of various O-RAN components, ensuring they are up to the mark and can effectively detect, log, and alert any anomalies.

7.8.1 O-CU logging and monitoring

Requirement Name: O-CU logging and monitoring

Requirement Reference: REQ-SEC-OCU-1, clause 5.1.4 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-07

DUT/s: O-CU

Test Name: TC_LOG_OCU

Purpose: The purpose of this test is to verify that the O-CU correctly logs and monitors security-related events effectively.

Procedure and execution steps

Preconditions

- The O-CU is powered on and operational.
- Logging and monitoring configurations are properly set up on the O-CU.

Execution steps

1. Logging

    a) The tester triggers an error or failure condition in the O-CU, such as connection attempts with invalid credentials, unauthorized access and a dropped connection.

    b) The tester verifies that the O-CU logs the error by capturing the relevant log entry.

2. Monitoring

    a) The tester monitors the key performance indicators (KPIs) of the O-CU, such as throughput, latency, or signal quality.

    b) The tester verifies that the monitoring system accurately collects and displays the KPI values in real-time.

    c) The tester introduces a simulated degradation or overload scenario on the O-CU, such as increasing network traffic or reducing available resources.

    d) The tester monitors the O-CU performance under the simulated scenario.

    e) The tester verifies that the monitoring system detects and raises alerts for the degraded performance or overload condition.

Expected Results

1. O-CU logs and generates alerts for security-related events, providing necessary information and timestamps for incident investigation and analysis.

2. The monitoring system provides accurate and real-time KPI values for the O-CU. The monitoring system detects and raises appropriate alerts for the degraded performance or overload condition.

Expected format of evidence:

1. Capture and analyse the logged error in the O-CU logs or logging system and document the presence of the log entry.

2. Document the monitored KPI values and the raised alerts, validate them against the expected values, and ensure they are triggered accurately in the monitoring system.

7.8.2 O-DU logging and monitoring

Requirement Name: O-DU logging and monitoring

Requirement Reference: REQ-SEC-ODU-1, clause 5.1.5 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-07

DUT/s: O-DU

Test Name: TC_LOG_ODU

Purpose: The purpose of this test is to ensure that the O-DU correctly logs and monitors security-related events effectively.

Procedure and execution steps

Preconditions

- The O-DU is powered on and operational.
- Logging and monitoring configurations are properly set up on the O-DU.

Execution steps

1. Logging
    a) The tester triggers an error or failure condition in the O-DU, such as connection attempts with invalid credentials, unauthorized access and a dropped connection.
    b) The tester verifies that the O-DU logs the error by capturing the relevant log entry.
2. Monitoring
    a) The tester monitors the key performance indicators (KPIs) of the O-DU, such as throughput, latency, or signal quality.
    b) The tester verifies that the monitoring system accurately collects and displays the KPI values in real-time.
    c) The tester introduces a simulated degradation or overload scenario on the O-DU, such as increasing network traffic or reducing available resources.
    d) Th tester monitors the O-DU performance under the simulated scenario.
    e) The tester verifies that the monitoring system detects and raises alerts for the degraded performance or overload condition.

Expected Results

1. O-DU logs and generates alerts for security-related events, providing necessary information and timestamps for incident investigation and analysis.
2. The monitoring system provides accurate and real-time KPI values for the O-DU. The monitoring system detects and raises appropriate alerts for the degraded performance or overload condition.

Expected format of evidence:

1. Capture and analyse the logged error in the O-DU logs or logging system and document the presence of the log entry.

2. Document the monitored KPI values and the raised alerts, validate them against the expected values, and ensure they are triggered accurately in the monitoring system.

7.8.3 O-RU logging and monitoring

Requirement Name: O-RU logging and monitoring

Requirement Reference: REQ-SEC-ORU-1, REQ-SEC-ORU-2, clause 5.1.6 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-07

DUT/s: O-RU

Test Name: TC_LOG_ORU

Purpose: The purpose of this test is to ensure that the O-RU correctly logs and monitors security-related events effectively.

Procedure and execution steps

Preconditions

- The O-RU is powered on and operational.
- Logging and monitoring configurations are properly set up on the O-RU.

Execution steps

1. Logging
    a) The tester triggers an error or failure condition in the O-RU, such as connection attempts with invalid credentials, unauthorized access and a dropped connection.
    b) The tester verifies that the O-RU logs the error by capturing the relevant log entry.
2. Monitoring
    a) The tester monitors the key performance indicators (KPIs) of the O-RU, such as throughput, latency, or signal quality.
    b) The tester verifies that the monitoring system accurately collects and displays the KPI values in real-time.
    c) The tester introduces a simulated degradation or overload scenario on the O-RU, such as increasing network traffic or reducing available resources.
    d) Th tester monitors the O-RU performance under the simulated scenario.
    e) The tester verifies that the monitoring system detects and raises alerts for the degraded performance or overload condition.

Expected Results

1. O-RU logs and generates alerts for security-related events, providing necessary information and timestamps for incident investigation and analysis.
2. The monitoring system provides accurate and real-time KPI values for the O-RU. The monitoring system detects and raises appropriate alerts for the degraded performance or overload condition.

Expected format of evidence:

1. Capture and analyse the logged error in the O-RU logs or logging system and document the presence of the log entry.
2. Document the monitored KPI values and the raised alerts, validate them against the expected values, and ensure they are triggered accurately in the monitoring system.

7.8.4 Near-RT RIC logging and monitoring

Requirement Name: Near-RT RIC logging and monitoring

Requirement Reference: REQ-SEC-NEAR-RT-4, clause 5.1.3 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-04

DUT/s: NEAR-RT RIC

Test Name: TC_LOG_NEAR_RT_RIC

Purpose: The purpose of this test is to ensure that the Near-RT RIC correctly logs and monitors security-related events effectively.

Procedure and execution steps

Preconditions

1. The Near-RT RIC is powered on and operational.
2. Logging and monitoring configurations are properly set up on the Near-RT RIC.

Execution steps

1. Logging
    a) The tester triggers an error or failure condition in the Near-RT RIC, such as connection attempts with invalid credentials, unauthorized access, or a dropped connection.
    b) The tester verifies that the Near-RT RIC logs the error by capturing the relevant log entry.
2. Monitoring
    a) The tester monitors the key performance indicators (KPIs) of the Near-RT RIC, such as throughput, latency, or signal quality.
    b) The tester verifies that the monitoring system accurately collects and displays the KPI values in real-time.
    c) The tester introduces a simulated degradation or overload scenario on the Near-RT RIC, such as increasing network traffic or reducing available resources.
    d) Th tester monitors the Near-RT RIC performance under the simulated scenario.
    e) The tester verifies that the monitoring system detects and raises alerts for the degraded performance or overload condition.

Expected Results

1. Near-RT RIC logs and generates alerts for security-related events, providing necessary information and timestamps for incident investigation and analysis.

2. The monitoring system provides accurate and real-time KPI values for the Near-RT RIC. The monitoring system detects and raises appropriate alerts for degraded performance or overload conditions.

Expected format of evidence:

1. Capture and analyse the logged error in the Near-RT RIC logs or logging system and document the presence of the log entry.
2. Document the monitored KPI values and the raised alerts, validate them against the expected values, and ensure they are triggered accurately in the monitoring system.

8 System security evaluation for O-RAN architecture elements

8.1 Overview

This clause contains security evaluations to be performed at the system level of an O-RAN architecture element, covering vulnerability scanning, data and information protection and system logging.

The objects in scope of these system security evaluation are SMO, Non-RT RIC and rApps, Near-RT RIC and rApps, O-CU-CP, O-CU-UP, O-DU, O-RU and O-Cloud.

8.2 System Vulnerability Scanning

8.2.1 System Vulnerability Scanning

Requirement Name: Robustness of OS and Applications

Requirement Reference: REQ-SEC-SYS-1, clause 5.3.6, REQ-SEC-ALM-PKG-1, clause 5.3.2.1.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Operating System (OS) and applications vulnerability scan of DUT

Threat References: T-O-RAN-01

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Vulnerability_Scanning

Purpose: To verify the DUT does not contain known vulnerabilities in the OS and applications. Perform vulnerability scanning to ensure that there are no known vulnerabilities on DUT, both in the Operating System(OS) and the applications installed, that can be detected by means of automatic testing tools via the IP enabled network interfaces, or to identify the known vulnerabilities on DUT and have a clear mitigation plan for the ones of high severity.

Known vulnerabilities are considered those which are publicly disclosed, found by users or reported by security researchers. Those vulnerabilities are widely detected by commercial, or open-source tools designed for this purpose.

Procedure and execution steps

Preconditions

DUT is the O-RAN architecture element with IP enabled network interfaces.

Execution steps

1. Run the vulnerability scanning tool and check the potential known vulnerabilities existing on OS and applications levels.
2. Evaluate the severity level of the existing vulnerabilities.

Expected results

The DUT is free from known vulnerabilities or there are security controls in place to mitigate the exploits associated with the vulnerabilities of high severity.

Expected format of evidence: Report files, log files and/or screenshots.

8.3 Data and Information Protection

Void

8.4 System logging

8.4.1 Introduction

This clause contains test cases related to security log management.

8.4.2 Security log format and related log fields

Requirement Name: Security logs check for date, time and location field IP address.

Requirement Reference: SEC-CTL-SLM-FLD-1, SEC-CTL-SLM-FLD-2, clause 5.3.8.8, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Support for security logs containing date, time and location field IP address.

Threat References: T-O-RAN-07

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Logs_Datetime_Fields_Validation

Purpose: To verify the log fields of security log data from DUT as per clause 5.3.8.8 of Security Requirements and Controls specifications [5] . The security log has the recommended date and time in ISO 8601 format and mandatorily log the location field IP address (IP address of the host from which security events are generated).

Procedure and execution steps

Preconditions

DUT is any O-RAN architecture element that creates/generates security event logs which acts as server. DUT also offers one or more services through which it can be accessed.

Client is the test system equipped to communicate securely with O-RAN architecture element and able to perform security related operations on DUT.

Execution steps

Table 8-1 Test and measurement equipment list

Scenario ID	Configuration
1	Login to the DUT via test system with authorized credentials.
2	Execute valid operations on the DUT which triggers/generates the security logs.

Expected results

Table 8-2 Test and measurement equipment list

Scenario ID	Expected result	Reason
1	Connection established.	Authentication successful.
2	All the security logs generated by the DUT have: 1] Date and time format as per ISO 8601 format as recommended by clause 5.3.8.8 of [5] 2] Location field IP address (IP address of the DUT) as mandated by clause 5.3.8.8 of [5]	Security log generation is successful.

Expected format of evidence: Log files

8.4.3 Authenticated Time Stamping

Requirement Name: Authenticated Time-Stamping

Requirement Reference: Clause 5.3.8.9.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Optional support NTPv4

Threat References: T-O-RAN-07

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Logs_Authenticated_Time_Stamping

Purpose: To verify that the DUT fulfills the optional requirement of supporting Network Time Protocol (NTP) version 4 as specified by RFC5905 for authenticated time stamping in the client role only.

Procedure and execution steps

Preconditions

1. The DUT is powered on and operational.
2. The NTP server specified for testing is reachable and configured to support authenticated time stamping.

Execution steps

Verify NTP Client Version:

- Access the DUT configuration settings related to NTP.
- Confirm that the DUT specifies NTP version 4 as the selected protocol.

Authentication Setup

- Configure the DUT to use the necessary authentication methods and -credentials (AES-CMAC/RFC4493, certificates for Autokey/RFC5906) required by RFC5905 for authenticated time stamping.
- Provide valid authentication credentials (certificates) for NTP communication.

Time Synchronization

- Initiate an NTP time synchronization process from the DUT to the specified NTP server.
- Monitor the communication between the DUT and the NTP server to ensure that the NTP packets are properly constructed with the required authentication parameters.
- Verify that the DUT successfully receives the authenticated time stamps from the NTP server.

Time Accuracy Check

- After synchronization, record the DUT internal clock time.
- Obtain the time from the NTP server's authenticated time stamp.
- Calculate the time difference between the DUT internal clock time and the received authenticated time stamp.
- Ensure that the time difference is within an acceptable tolerance, considering network latency and authentication processing.

Expected results

The DUT fulfills the requirement of supporting Network Time Protocol (NTP) version 4 for authenticated time stamping, as specified by RFC5905. The NTP communication successfully employs the configured authentication methods, and the time synchronization process ensures accurate timekeeping within the specified tolerance. An accuracy below 1 second should be measured to pass.

Expected format of evidence: Log files, traffic captures and/or screenshots.

8.4.4 Network Security and System Security Events

Requirement Name: Network Security Events to be Logged and System Security Events to be Logged.

Requirement Reference: REQ-SEC-SLM-NET-EVT-1, REQ-SEC-SLM-GEN-EVT-1, REQ-SEC-SLM-GEN-EVT-2, REQ-SEC-SLM-GEN-EVT-3, REQ-SEC-SLM-HYP-EVT-1, REQ-SEC-SLM-HYP-EVT-2, REQ-SEC-SLM-HYP-EVT-3, REQ-SEC-SLM-CON-EVT-1, REQ-SEC-SLM-CON-EVT-2, REQ-SEC-SLM-CON-EVT-3.

Requirement Description: Logging of network and system security events in O-Cloud

Threat References: T-O-RAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-09, T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06, T-IMG-01, T-IMG-02, T-ADMIN-02.

DUT/s: O-Cloud

Test Name: TC_Logs_Network_System_Security_Events

Purpose: The purpose of the test is to verify the logging of security events from O-Cloud as per the Security Requirements and Controls Specifications [5].

Procedure and execution steps

Preconditions

A tester has access to testing equipment that can connect to the O-Cloud with administrative privileges to the operating system, hypervisor, and container engine.

Execution steps

1. Login to the DUT via testing equipment with administrative credentials.
2. Execute the following operations on the DUT.
    2.1. Create a new network configuration.
    2.2. Modify an existing network configuration.
    2.3. Disable a port.
    2.4. Enable a port.
    2.5. Generate packets that exceed configured firewall limits.
    2.6. Generate at least one network connection.
    2.7. Reboot a virtual machine and then reboot the host operating system.
    2.8. Shutdown a virtual machine then shutdown the host operating system.
    2.9. Create a scheduled job within the host operating systems, hypervisor, and container engine.
    2.10. Make a configuration change to the host operating system and hypervisor.
    2.11. Attach and detach a virtual disk to a virtual machine.
    2.12. Create a virtual machine.
    2.13. Start a virtual machine.
    2.14. Stop a virtual machine.
    2.15. Delete a virtual machine.
    2.16. Add an image to the container repository.
    2.17. Modify an image to the container repository.
    2.18. Remove an image to the container repository.
    2.19. Create a container.
    2.20. Start a container.
    2.21. Stop a container.
    2.22. Restart a container.
    2.23. Delete a container.
    2.24. Create a container volume.
    2.25. Mount a container volume.
    2.26. Delete a container volume.

Expected results

All the security logs produced by O-Cloud contain log messages that describe the actions taken in the execution steps.

- For execution step 2.1 the log message indicates the creation of a new network configuration.
- For execution step 2.2 the log message indicates the modification of an existing network configuration.

- For execution step 2.3 the log message indicates the disabling of a port.
- For execution step 2.4 the log message indicates the enabling a port.
- For execution step 2.5 the log message indicates that packets have exceeded configured firewall limits.
- For execution step 2.6 the log message indicates a network connection has been attempted along with details about that network connection including source and destination IP addresses.
- For execution step 2.7 the log message indicates that a virtual machine was rebooted, and a subsequent log message indicates that a host operating system has been rebooted.
- For execution step 2.8 the log message indicates that a virtual machine has been shut down and a subsequent log message indicates that the host operating system has been shut down.
- For execution step 2.9 the log message indicates that a scheduled job was created within the host operating system, a subsequent log message indicates that a scheduled job was created in the hypervisor, and a subsequent log message indicates that a scheduled job was created in the container engine.
- For execution step 2.10 the log message indicates that a configuration change was made to the host operating system and a subsequent log message indicates that a configuration change was made to the hypervisor.
- For execution step 2.11 the log message indicates that a virtual disk was attached to a virtual machine, and a subsequent log message indicates that a virtual disk was detached from a virtual machine.
- For execution step 2.12 the log message indicates that a virtual machine was created.
- For execution step 2.13 the log message indicates that a virtual machine was started.
- For execution step 2.14 the log message indicates that a virtual machine was stopped.
- For execution step 2.15 the log message indicates that a virtual machine was deleted.
- For execution step 2.16 the log message indicates that an image was added to the container repository.
- For execution step 2.17 the log message indicates that an image was modified in the container repository.
- For execution steps 2.18 the log message indicates that an image was removed from the container repository.
- For execution step 2.19 the log message indicates a container was created.
- For execution step 2.20 the log message indicates that a container was started.
- For execution step 2.21 the log message indicates that a container was stopped.
- For execution step 2.22 the log message indicated that a container was restarted.
- For execution step 2.23 the log message indicates that a container was deleted.
- For execution step 2.24 the log message indicates that a container volume was created.
- For execution step 2.25 the log message indicates that a container volume was mounted.
- For execution step 2.26 the log message indicates that a container volume was deleted.

Expected format of evidence: Generated Log Files from DUT/s.

8.4.5 Application Security Events

Requirement Name: Application Security Events to be Logged.

Requirement Reference: REQ-SEC-SLM-APP-EVT-1, REQ-SEC-SLM-APP-EVT-2.

Requirement Description: Support for the logging of security events in network functions

Threat References: T-OPENSRC-01, T-xAPP-01, T-xAPP-02, T-xAPP-03, T-xAPP-04, T-rAPP-01, T-rAPP-02, T-rAPP-03, T-rAPP-04, T-rAPP-05, T-rAPP-06, T-rAPP-07, T-PNF-01.

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Logs_Application_Security_Events

Purpose: The purpose of the test is to verify the logging of security event data from O-RAN Network Functions as per the Security Requirements and Controls Specifications [5].

Procedure and execution steps

Preconditions

A tester has access to testing equipment that can connect to any O-RAN network function.

Execution steps

NOTE 1: Execution steps not applicable to the DUT may be skipped.

1. Login to the DUT via test equipment with authorized credentials.
2. Conduct an operation on the DUT that is known to generate an error.
3. Conduct an operation on the DUT that is known to load a dynamic library.

Expected results

All the security logs produced by O-RAN Network Functions contain log messages that pertain to the actions taken in the execution steps.

- For execution step 2 the log message contains an error message.
- For execution step 3 the log message contains a message indicating that a dynamic library loaded and details about that library.

Expected format of evidence: Generated Log Files from DUT/s.

8.4.6 Data Access Security Events

Requirement Name: Data Access Security Events to be Logged.

Requirement Reference: REQ-SEC-SLM-DAT-EVT-1, REQ-SEC-SLM-DAT-EVT-2, REQ-SEC-SLM-DAT-EVT-3, REQ-SEC-SLM-DAT-EVT-4, REQ-SEC-SLM-DAT-EVT-5, REQ-SEC-SLM-DAT-EVT-6, REQ-SEC-SLM-DAT-EVT-7, REQ-SEC-SLM-DAT-EVT-8.

Requirement Description: Logging of data access security events.

Threat References: T-VM-C-01, T-NEAR-RT-03, T-O-RAN-07, T-O-RAN-08, T-GEN-05

DUT/s: SMO, Non-RT RIC, Near-RT RIC, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Logs_Data_Access_Security_Events

Purpose: The purpose of the test is to verify the logging of data access security events as per the O-RAN Security Requirements and Controls Specifications [5].

Procedure and execution steps

Preconditions

A tester has access to testing equipment that can communicate securely with the DUT and is able to perform security and administrative related operations.

Execution steps

NOTE 1: Execution steps not applicable to the DUT may be skipped.

1. Login to the DUT via testing equipment with authorized credentials.
2. Execute the following operations on the DUT.
    2.1. Add a new file.
    2.2. Delete an existing file.
    2.3. Attempt to add a file where you are not authorized to do so.

2.4. Attempt to delete a file where you are not authorized to do so.

2.5. Read an existing file.

2.6. Write to an existing file.

2.7. Attempt to read to a file where you are not authorized to do so.

2.8. Attempt to write to a file where you are not authorized to do so.

2.9. Create a new directory.

2.10. Delete an existing directory.

2.11. Attempt to create a directory where you are not authorized to do so.

2.12. Attempt to delete a directory where you are not authorized to do so.

2.13. Add data to a datastore or database.

2.14. Delete data from a datastore or database.

2.15. Attempt to add data to a datastore or database where you are not authorized to do so.

2.16. Attempt to delete data from a datastore or database where you are not authorized to do so.

2.17. Read data from a datastore or database.

2.18. Write data from a datastore or database.

2.19. Attempt to read data from a datastore or database where you are not authorized to do so.

2.20. Attempt to write data to a datastore or database where you are not authorized to do so.

2.21. Make a permissions change to a file.

2.22. Make a permissions change to a directory.

2.23. Make a permissions change to a datastore or database.

Expected results

All the security logs produced by O-RAN architecture elements contain log messages that document appropriately the actions taken in the execution steps.

- For execution step 2.1 the log message indicates that a new file was added.
- For execution step 2.2 the log message indicates an existing file was deleted.
- For execution step 2.3 the log message indicates an unauthorized attempt to add a file.
- For execution step 2.4 the log message indicates an unauthorized attempt to delete a file.
- For execution step 2.5 the log message indicates an existing file was read.
- For execution step 2.6 the log message indicates an existing file was written.
- For execution step 2.7 the log message indicates an unauthorized attempt to read to a file.
- For execution step 2.8 the log message indicates an unauthorized attempt to write to a file.
- For execution step 2.9 the log message indicates a new directory was created.
- For execution step 2.10 the log message indicates an existing directory was deleted.
- For execution step 2.11 the log message indicates an unauthorized attempt to create a directory.
- For execution step 2.12 the log message indicates an unauthorized attempt to delete a directory.
- For execution step 2.13 the log message indicates data was added to a datastore or database.
- For execution step 2.14 the log message indicates data was deleted from a datastore or database.
- For execution step 2.15 the log message indicates an unauthorized attempt to add data to a datastore or database.

- For execution step 2.16 the log message indicates an unauthorized attempt to delete data from a datastore or database.
- For execution step 2.17 the log message indicates that data was read from a datastore or database.
- For execution step 2.18 the log message indicates that data was written to a datastore or database.
- For execution step 2.19 the log message indicates an unauthorized attempt to read data from a datastore or database.
- For execution step 2.20 the log message indicates an unauthorized attempt to write data to a datastore or database.
- For execution step 2.21 the log message indicates a permissions change to a file.
- For execution step 2.22 the log message indicates a permissions change to a directory.
- For execution step 2.23 the log message indicates a permissions change to a datastore or database.

Expected format of evidence: Generated Log Files from DUT.

8.4.7 Account and Identity Security Events

Requirement Name: Account and Identity Security Events to be Logged.

Requirement Reference: REQ-SEC-SLM-AAI-EVT-1, REQ-SEC-SLM-AAI-EVT-2, REQ-SEC-SLM-AAI-EVT-3, REQ-SEC-SLM-AAI-EVT-4, REQ-SEC-SLM-AAI-EVT-5, REQ-SEC-SLM-AAI-EVT-6, REQ-SEC-SLM-AAI-EVT-7, REQ-SEC-SLM-AAI-EVT-9, REQ-SEC-SLM-AAI-EVT-10.

Requirement Description: Logging of account and identity security events.

Threat References: T-GEN-02, T-O-RAN-02, T-O-RAN-06, T-O-RAN-07, T-ProtocolStack-02, T-SMO-02, T-SMO-05, T-SMO-08, T-SMO-25, T-SMO-30, T-NEAR-RT-03.

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_Logs_Account_and_Identity_Security_Events

Purpose: The purpose of the test is to verify the logging of account and identity access security events as per O-RAN Security Requirements and Controls Specifications [5]

Procedure and execution steps

Preconditions

A tester will have access to testing equipment that can communicate securely with the DUT and is able to perform security and administrative related operations.

Execution steps

NOTE 1: Execution steps not applicable to the DUT may be skipped.

1. Login to the DUT via testing equipment with authorized credentials.
2. Execute the following operations on the DUT.
    2.1. Create an account.
    2.2. Modify an existing account.
    2.3. Delete an existing account.
    2.4. Attempt to create an account where you are not authorized to do so.
    2.5. Change the privilege level of an existing account from a lower privilege to a higher privilege.
    2.6. Attempt to change the privilege level of an existing account where you are not authorized to do so.
    2.7. Change the group membership of an existing account.

2.8. Attempt to change the group membership of an existing account where you are not authorized to do so.

2.9. Use a function in the DUT that requires a specific authorization that you have been assigned.

2.10. Attempt to use a function in the DUT that requires a specific authorization that you have not been assigned.

2.11. Authenticate an account to the DUT that has been configured to access that DUT.

2.12. Attempt to authenticate an account to the DUT that has not been configured to access that DUT.

2.13. Change the privilege level of an existing account from a higher privilege to a lower privilege.

2.14. Access the DUT with an account the does not require authentication.

2.15. End a session with the DUT.

Expected results

All the security logs produced by O-RAN architecture elements contain log messages that document appropriately the actions taken in the execution steps.

- For execution step 2.1 the log message indicates that an account was created.
- For execution step 2.2 the log message indicates that an existing account was modified.
- For execution step 2.3 the log message indicates that an existing account was deleted.
- For execution step 2.4 the log message indicates an unauthorized attempt to create an account.
- For execution step 2.5 the log message indicates a privilege level change of an existing account from a lower privilege to a higher privilege.
- For execution step 2.6 the log message indicates an unauthorized attempt to change the privilege level of an existing account.
- For execution step 2.7 the log message indicates that the group membership had changed for an existing account.
- For execution step 2.8 the log message indicates an unauthorized attempt to change the group membership of an existing account.
- For execution step 2.9 the log message indicates the use of a restricted function.
- For execution step 2.10 the log message indicates an unauthorized attempt to use a restricted function.
- For execution step 2.11 the log message indicates the successful authentication of an account.
- For execution step 2.12 the log message indicates the unsuccessful attempt to authenticate an account.
- For execution step 2.13 the log message indicates a privilege level change of an existing account from a higher privilege to a lower privilege.
- For execution step 2.14 the log message indicates access with an account the does not require authentication.
- For execution step 2.15 the log message indicates the end of a session.

Expected format of evidence: Generated Log Files from DUT.

8.4.8 General Security Events

Requirement Name: General Security Events to be Logged.

Requirement Reference: REQ-SEC-SLM-GSE-1, REQ-SEC-SLM-GSE-2, REQ-SEC-SLM-GSE-3, REQ-SEC-SLM-GSE-4, REQ-SEC-SLM-GSE-5, REQ-SEC-SLM-GSE-6.

Requirement Description: Logging of general security events.

Threat References: T-ORAN-01, T-O-RAN-02, T-O-RAN-03, T-O-RAN-08, T-GEN-02, T-VM-C-01, T-VM-C-04, T-VM-C-06, T-IMG-01, T-IMG-04, T-VL-01, T-VL-02, T-xAPP-01, T-rAPP-03, T-HW-02.

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_General_Security_Events_Logged

Purpose: The purpose of the test is to verify the logging of general security events

Procedure and execution steps

Preconditions

A tester will have access to testing equipment that can communicate securely with the DUT and is able to perform security and administrative related operations.

Execution steps

NOTE 1: Execution steps not applicable to the DUT may be skipped.

1. Login to the DUT via testing equipment with authorized credentials.
2. Execute the following operations on the DUT.
    2.1. Enable security software such as firewalls, malware protection, data loss prevention or intrusion detection systems.
    2.2. Disable security software such as firewalls, malware protection, data loss prevention or intrusion detection systems.
    2.3. Log into DUT using an account with administrative privileges and perform a function that requires those privileges.
    2.4. Make a change to the security configuration of the DUT.
    2.5. View a certificate or key on the DUT.
    2.6. Export a certificate or key from the DUT.
    2.7. Renew a certificate or key on the DUT.
    2.8. Import a certificate or key from the DUT.
    2.9. Modify a certificate or key on the DUT.
    2.10. Delete a certificate or key from the DUT.
    2.11. Perform a cryptographic operation on the DUT that involves signatures, encryption, hashing, key generation or key destruction.
    2.12. Submit a security patch to the DUT but do not apply it.

Expected results

All the security logs produced by O-RAN architecture elements contain log messages that document appropriately the actions taken in the execution steps.

- For execution steps 2.1 and 2.2 the log message indicates that the security software has been enabled or disabled.
- For execution step 2.3 the log message indicates the use of administrative privileges.
- For execution step 2.4 the log message indicates that a change to the security configuration has occurred and the nature of the change.
- For executions 2.5 through 2.11 the log message is absent of any sensitive information related to the certificate or key.
- For execution 2.12 the log message indicates that a security patch was submitted but not applied.

Expected format of evidence: Generated Log Files from DUT.

8.4.9 Void

9 Software security evaluation for O-RAN architecture elements

9.1 Overview

This clause contains a set of software security evaluations of an O-RAN architecture element, covering Software Lifecycle Management.

9.2 Open-Source Software Component Analysis

Void

9.3 Binary Static Analysis

Void

9.4 Software Bill of Materials (SBOM)

9.4.1 SBOM Signature

Requirement Name: A digital signature is provided for the SBOM.

Requirement Reference: SEC-CTL-SBOM-001, clause 6.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: SBOM is authenticity, integrity protected and provided in a standard format.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test name: TC_SBOM_Signature

Purpose: To verify the SBOM is provided with a digital signature

Procedure and execution steps

Preconditions

SBOM is provided. Tools to verify the digital signature are available. Solution Provider key or certificate is provided.

Execution steps

Ensure the SBOM is provided with a digital signature in the format as described below. Verify SBOM digital signature is valid using the software provider’s public key or certificate. Depending on the format of the SBOM, there are various ways how to include and verify the digital signature of the SBOM. Below, the digital signature methods are detailed.

SPDX

YAML, RDF and tag data: The signature is in a separate file from the SPDX file. Digital signature format is CMS/PKCS#7/CAdES.

EXAMPLE: foo.spdx is accompanied by foo.spdx.sig containing its signature

XML: XML Signature 2.0

JSON: JSON Web Signature (JWS), and JSON Signature Format (JSF).

CycloneDX

XML: XML Signature 2.0

JSON: JSON Web Signature (JWS), and JSON Signature Format (JSF).

SWID

XML: XML Signature 2.0

Expected results

Digital signature of the SBOM is valid.

Expected format of evidence: Report file, screenshot, or log file from tool used for verification of digital signature.

9.4.2 SBOM Data Fields

Requirement Name: Data fields are according to NTIA guidance [13]

Requirement Reference: REQ-SBOM-002, REQ-SBOM-011, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: A minimum set of data fields are included in the SBOM and it is in an standard format.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SBOM_Data_Fields

Purpose: To verify the minimum set of data fields are included in the SBOM

Procedure and execution steps

Preconditions

SBOM file is provided. Tools to verify the data fields are available.

Execution steps

Run the SBOM check tool and verify that there is minimum set of data fields present in SBOM depending on the SBOM format used.

Table 9-1: Minimum set of data fields for SPDX [12]

NTIA field	NTIA description	SPDX 2.2.1 field
Supplier Name	The name of an entity that creates, defines, and identifies components	PackageSupplier
Component Name	Designation assigned to a unit of software defined by the original supplier	PackageName
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version	PackageVersion
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases	SPDXID (Package SPDX Identifier)
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y	Relationship: CONTAINS
Author of SBOM Data	The name of the entity that creates the SBOM data for this component	Creator
Timestamp	Record of the date and time of the SBOM data assembly	Created

Table 9-2: Minimum set of data fields for CycloneDX [13]

NTIA field	NTIA description	CycloneDX field
Supplier Name	The name of an entity that creates, defines, and identifies components	publisher
Component Name	Designation assigned to a unit of software defined by the original supplier	name
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version	version
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases	bom/serialNumber and component/bom-ref
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y	(Nested assembly/subassembly and/or dependency graphs)
Author of SBOM Data	The name of the entity that creates the SBOM data for this component	bom-descriptor:metadata/ manufacture/contact
Timestamp	Record of the date and time of the SBOM data assembly	timestamp

Table 9-3: Minimum set of data fields for SWID [13]

NTIA field	NTIA description	SWID tag
Supplier Name	The name of an entity that creates, defines, and identifies components	<Entity> @role (softwareCreator/publisher), @name
Component Name	Designation assigned to a unit of software defined by the original supplier	<softwareIdentity> @name
Version of the Component	Identifier used by the supplier to specify a change in software from a previously identified version	<softwareIdentity> @version
Other Unique Identifiers	Other identifiers that are used to identify a component, or serve as a look-up key for relevant databases	<softwareIdentity> @tagID
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y	<Link> @rel, @href
Author of SBOM Data	The name of the entity that creates the SBOM data for this component	<Entity> @role (tagCreator), @name
Timestamp	Record of the date and time of the SBOM data assembly	-

This test is part of the O-RAN software producer's Software Development Lifecycle (SDLC).

Expected results

Minimum set of data fields are present.

Expected format of evidence: Report file, screenshot, or log file from SBOM check tool.

9.4.3 SBOM Format

Requirement Name: SBOM is provided in one of the accepted formats: SPDX, CycloneDX, or SWID.

Requirement Reference: REQ-SBOM-011, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: SBOM is provided in a standard format.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SBOM_Format

Purpose: To verify that the SBOM is provided in one of standard formats.

Procedure and execution steps

Preconditions

SBOM is provided. Tools to verify the SBOM are available.

Execution steps

Run the SBOM check tool to verify the SBOM format.

Expected results

SBOM format is SPDX, CycloneDX, or SWID.

Expected format of evidence: Report file.

9.4.4 SBOM Depth

Requirement Name: SBOM Depth is in the required level.

Requirement Reference: REQ-SBOM-004, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: The SBOM Depth is the required for the different types of software.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SBOM_Depth

Purpose: To verify that the SBOM depth is provided to the level specified.

Procedure and execution steps

Preconditions

SBOM is provided. Tools to verify the SBOM are available.

Execution steps

Run the SBOM check tool to verify the SBOM depth provided.

At a minimum, all top-level dependencies are listed.

Expected results

SBOM depth is as specified in the requirements:
- top-level for every O-RAN software delivery

Expected format of evidence: Report file, log file from SBOM check tool.

9.4.5 Void

9.4.6 SBOM Version Verification

Requirement Name: The version in the SBOM is accurately and matches the actual O-RAN software package version.

Requirement Reference: REQ-SBOM-002, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Version of the software component included in the SBOM is matching the actual software component version.

Threat References: T-O-RAN-08, T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SBOM_VERSION_VERIFICATION

Purpose: The purpose of this test is to ensure the SBOM reflects the current version of software package.

Procedure and execution steps

Preconditions

SBOM is provided. Tools to verify the SBOM are available. Access to version information regarding O-RAN software package.

Execution steps:

1. For each software component in the package, compare the version listed in the SBOM with the actual software component version.
2. Ensure that the SBOM's version matches the component's version.
3. Document any discrepancies.

Expected Results:

The version specified in the SBOM aligns with the actual version of the software component.
Expected format of evidence:

A report detailing:

- name of the software package component(s), SBOM indicated version, actual version, notes on any discrepancies or issues found.

9.4.7 Void

9.4.8 SBOM Presence

Requirement Name: SBOM provided with all O-RAN Software.

Requirement Reference: REQ-SBOM-001, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: For every O-RAN software delivery, an SBOM is available.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SBOM_Presence

Purpose: The purpose of this test is to ensure that for every O-RAN software delivery an SBOM is available. This is applicable to all software components within the O-RAN system.

Procedure and execution steps

Preconditions

- SBOM is provided. Tools to verify the SBOM are available.

Execution steps

1. List O-RAN software delivery.
2. Look for associated files or documentation indicating the presence of an SBOM.
3. Validate the SBOM's content to ensure it's not just a placeholder.

4. Document any software delivery that lacks a genuine SBOM.

Expected Results

Every O-RAN software delivery has a genuine SBOM associated with it.
Expected Format of Evidence:

- A report detailing:
  - names of the software package components;
  - status of its SBOM (Present/Absent).
  - notes on any discrepancies or issues found.

9.4.9 SBOM Vulnerabilities Field

Requirement Name: Vulnerabilities field omission in SBOMs.

Requirement Reference: REQ-SBOM-003, clause 6.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Vulnerabilities are not included as an additional data field because it would represent a static view from a specific point in time, while vulnerabilities are constantly evolving. Therefore, the vulnerabilities field in SBOM for O-RAN software cannot be relied upon to determine the SBOM vulnerabilities by the Service provider. Service providers need to perform their own vulnerability assessment, at least at the moment of the SBOM release.

Threat References: T-O-RAN-09

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud **Test Name**: TC_SBOM_Vulnerabilities_Fields

Purpose: Verify that vulnerabilities fields is not included as an additional field to the SBOM.

Procedure and execution steps

Preconditions

SBOM is provided. Tools to verify the SBOM are available.

Execution steps

Verify that no vulnerabilities fields exist within the SBOM.

Expected Results

There are no vulnerabilities field(s) present in the SBOM.

Expected Format of Evidence: screenshot(s)

9.4.10 Void

9.5 Software Image Signing and Verification

9.5.1 Void

9.5.2 Software Signature Verification

Requirement Name: Any software image(s) of O-RAN architecture element(s) and/or app(s) is verified for its signature(s) by the Service provider for onboarding and/or deployment (instantiation process).

Requirement Reference: SEC-CTL-ALM-PKG-1, SEC-CTL-ALM-PKG-1B, SEC-CTL-ALM-PKG-3, clause 5.3.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Application package is signed and verified for integrity and authenticity protection.

Threat References: T-IMG-01, T-VM-C-02, T-Near-RT-01, T-Near-RT-02, T-xAPP-02

DUT/s: SMO, Non-RT RIC and rApps, Near-RT RIC and xApps, O-CU-CP, O-CU-UP, O-DU, O-RU, O-Cloud

Test Name: TC_SW_Img_Pkg_Verification

Purpose: Ensure signatures on O-RAN software image or application package and package artifacts are verified during onboarding and instantiation.

Procedure and execution steps

Preconditions

Digitally signed software image or Application package with shared necessary digital certificates or public key is validated.

EXAMPLE: Root CA certificate, any intermediate or RA certificates.

Execution steps

The signature of the software image or Application package is verified either manually or using a software signing service. The software used to verify the signature(s) could be provided by software producer or internally published by the Service Provider.

The Solution Provider signature of the ORAN software image(s) or application package is verified during onboarding.

For image or Application package artifact instantiation, Service Provider signature of the software image or Application package verification is executed first, if applicable. Subsequently, Solution Provider signature is verified during instantiation.

Expected results

For signature verification during onboarding:

The Solution Provider signature verification for software image or Application package is successful.

For signature verification during instantiation:

The Solution Provider signature verification for image or Application artifact during instantiation is successful. Additionally, if applicable, the Service Provider signature verification is also successful.

Expected Format of Evidence: screenshot(s)

10 ML security validation for O-RAN system

10.1 Overview

AI/ML technologies and models are adopted at the O-RAN system Non-RT RIC and Near-RT RIC to enable O-RAN use cases: traffic steering, massive MIMO optimization, radio resource allocation for UAV applications, position accuracy enhancement, beam management, and enhance CSI feedback. Other uses cases could be checked in document O-RAN Use Cases Detailed Specification [22].

10.2 ML Data Poisoning

Void

11 Security tests of O-RAN interfaces

11.1 FH

11.1.1 Overview

This clause contains security tests to validate the security protection mechanism of the O-RAN Open Fronthaul interface.

11.1.2 Open Fronthaul Point-to-Point LAN Segment

11.1.2.0 Introduction

IEEE 802.1X-2020 Port-based Network Access Control [11] provides the means to control network access in point-to-point LAN segments within the Open Fronthaul network. Port-based network access control in the O-RAN Alliance Open Fronthaul comprises supplicant, authenticator, and authentication of server entities described in IEEE 802.1X-2020 [11].

The security test cases in this clause cover the validation of the authenticator and supplicant functionalities of the 802.1X, affecting to all the elements acting as an O-RAN Open Fronthaul network elements, including but not limited to, O-DU, O-RU, switches, FHM, FHGW, TNE and PRTC-T/GM as defined in clause 5.2.5.5 of Security Requirements and Controls Specifications [5].

11.1.2.1 Authenticator Validation

Requirement Name: Authenticator function validation

Requirement Reference: SEC-CTL-OFHPLS-3, SEC-CTL-OFHPLS-4, SEC-CTL-OFHPLS-5, REQ-SEC-OFHPLS-1, clause 5.2.5.5, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Requirements of Authenticator in the Open Fronthaul network and its interface to an Authentication Server

Threat References: T-FRHAUL-02

DUT/s: O-DU

Test Name: TC_Authenticator_Validation

Purpose: To verify and validate the authenticator requirements of the network component to serve the request from supplicant(s) using EAP TLS authentication per 802.1X-2020 [11].

Procedure and execution steps

Preconditions

DUT supports authenticator role of the 802.1X for port-based network access control.

IP enabled network interface of DUT reachable to the authentication server and 802.1X enabled for its Open Fronthaul interface.

Set up an authentication server with root, server and client certificates, and the start the authentication server.

EXAMPLE: RADIUS[1] server (e.g. free radius on Linux) can be a possible authentication server.

Execution steps

Run the 802.1X test tool emulating the request(s) from the supplicant(s) towards the DUT, which is the authenticator and ensure the 802.1X authentication process runs to completion.

The following test scenarios are executed:

Table 11-1: Scenarios to be executed

Scenario ID	Configuration
1	Test tool (as supplicant) setting for 802.1X with EAPoL, correct Identity (Certificate DN) and Client Certificate (provisioned on the Radius server)
2	Test tool (as supplicant) setting for 802.1X with EAPoL, correct Identity (Certificate DN) and incorrect Client Certificate (not provisioned on the Radius server)
3	Test tool (as supplicant) setting for 802.1X with EAPoL and incorrect Identity (Certificate DN)
4	Test tool (as supplicant) setting for 802.1X with EAP non-TLS authentication

Expected results

DUT successfully completes the procedure for the 802.1X Authentication validation (being granted or denied), for each test scenario:

[1] Because Radius support is required over interface between an authenticator and authentication server in the security requirement specification, only Radius authentication server is called for in this security test environment setup. Diameter based authentication server could be used as an alternative.

Table 11-2: Expected results

Scenario ID	Expected result	Reason
1	Connection established	Authentication successfully
2	Connection not established	Fail Authentication because the certificate is wrong
3	Connection not established	Fail Authentication because the Identity is wrong
4	Connection not established	Fail Authentication because the authentication type is wrong

Expected format of evidence: log files and/or traffic captures.

11.1.2.2 Supplicant Validation

Requirement Name: Supplicant function validation

Requirement Reference: SEC-CTL-OFHPLS-2, SEC-CTL-OFHPLS-5, REQ-SEC-OFHPLS-1, clause 5.2.5.5, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Requirements of Supplicant in the Open Fronthaul network

Threat References: T-FRHAUL-02

DUT/s: O-RU, O-DU

Test Name: TC_Supplicant_Validation

Purpose: To verify the supplicant requirement of the network component for port connection request using EAP TLS authentication per 802.1X-2020 [11].

Procedure and execution steps

Preconditions

DUT supports supplicant role of the 802.1X for port-based network access control.

Set up an authentication server with root, server and client certificates, and start the authentication server.

EXAMPLE: RADIUS[2] server (e.g. free radius on Linux) can be a possible authentication server.

Set up the 802.1X test tool host/device as the authenticator with EAP TLS authentication for 802.1X protocol and configure the preset authentication server.

Execution steps

Start the test run as an emulated authenticator waiting for the supplicant request.

Configure and enable the DUT to start the port connection request as a supplicant towards the 802.1X test tool, which is the authenticator and verify the 802.1X authentication process runs to completion.

The following test scenarios are executed:

[2] Because Radius support is required over interface between an authenticator and authentication server in the security requirement specification, only Radius authentication server is called for in this security test environment setup. Diameter based authentication server could be used as an alternative.

Table 11-3: Scenarios to be executed

Scenario ID	Configuration
1	DUT (as supplicant) setting for 802.1X with EAPoL, correct Identity (Certificate DN) and Client Certificate (provisioned on the Radius server)
2	DUT (as supplicant) setting for 802.1X with EAPoL, correct Identity (Certificate DN) and incorrect Client Certificate (un-provisioned on the Radius server)
3	DUT (as supplicant) setting for 802.1X with EAPoL and incorrect Identity (Certificate DN)
4	DUT (as supplicant) setting for 802.1X with EAP non-TLS authentication (optional)

Expected results

DUT successfully completes the procedure for the supplicant validation (being granted or denied), for each test scenario:

Table 11-4: Expected results

Scenario ID	Expected result	Reason
1	Connection established	Authentication successfully
2	Connection not established	Fail Authentication because the certificate is wrong
3	Connection not established	Fail Authentication because the Identity is wrong
4	Connection not established	Fail Authentication because the authentication type is wrong

Expected format of evidence: log files and/or traffic captures.

11.1.3 M-Plane

11.1.3.1 SSH-based M-Plane authentication, authorization and access control protection

11.1.3.1.0 Introduction

The test cases outlined in this clause verify M-Plane authenticity, authorization, and access control protection over the FH interface using SSH.

11.1.3.1.1 Secure Password-Based Authentication and Authorization in FH_MPLANE Using SSH

Requirement Name: M-Plane authenticity protection over FH interface using SSH

Requirement Reference: clause 5.4 in O-RAN WG4 Management Plane Specification [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU

Test Name: TC_FH_MPLANE_SSH-PASSWORD-BASED_AUTHENTICATION_AUTHORIZATION

Purpose: The purpose of this test is to verify the SSH password-based authentication and authorization mechanisms on the front-haul (FH) of the O-DU by the O-RU.

Procedure and execution steps

Preconditions

1. The O-RU is properly configured and operational.
2. Test equipment (potentially an O-DU or a dedicated SSH client simulator) is configured to establish SSH connections to the O-RU
3. NACM with NETCONF is enabled and configured for authorization on the FH interface.
4. SSH is properly implemented and configured as defined in [2] clause 4.1.

Execution steps

1. Execute the test on the SSH protocol as defined in clause 6.2.
2. Positive Case: Successful SSH password-based authentication and authorization.
   - Test the successful SSH password-based authentication and authorization of the test equipment by the O-RU.
     a) Establish an SSH connection from the test equipment (acting as SSH client) to the O-RU (acting as SSH server) using the SSH password.

   EXAMPLE: "Command: ssh <username>@<O-RU_IP>"

     b) Verify that the O-RU successfully authenticates the test equipment using the SSH password.

   EXAMPLE: "Command: **show ssh sessions**"

     c) Validate that the test equipment is authorized to perform the requested operations on the FH interface after successful authentication. This operation should be within the scope of permitted actions for the authenticated entity.

   EXAMPLE of operations: "start up" installation, software management, configuration management, performance management, fault management and file management towards the O-RU

     - Monitor the responses from the O-RU to these operations.
     - Record whether each operation was successfully executed, partially executed, or rejected.
     - Verify the O-RU logs to confirm that the operations were authorized.
3. Negative Case: Failed SSH password-based authentication.
   - Test the handling of failed SSH password-based authentication attempts of the test equipment by the O-RU in different scenarios.
     a) Attempt with incorrect password
       - Attempt to establish an SSH connection from the test equipment to the O-RU using an incorrect password.

   EXAMPLE: Command: **ssh <valid_username>@<O-RU_IP>**", using an incorrect password

       - Verify that the O-RU rejects the SSH connection due to the authentication failure.

     b) Attempt with non-existent username

- Attempt to establish an SSH connection using a username that does not exist in the O-RU's user database.

EXAMPLE: Command: ssh <invalid_username>@<O-RU_IP>

- Verify that the O-RU rejects the SSH connection, confirming that authentication does not proceed with non-existent usernames.

Expected Results

For step 1): Expected results in clause 6.2

For step 2):

- The SSH connection is successfully established using the SSH password.
- The O-RU validates the test equipment's SSH password for authentication.
- The O-RU grants the necessary authorization for the requested operations.

For step 3):

- The SSH connection attempt fails due to the incorrect password.
- The O-RU identifies the authentication failure and denies access.
- The SSH connection attempt fails due to the invalid username.
- The O-RU identifies the authentication failure and prevents access.

Expected format of evidence

For step 1): Logs and screenshots showing adherence to SSH protocol specifications as defined in [2] clause 4.1.

For step 2): Logs showing successful SSH authentication and authorization events.

For step 3): Logs or error messages indicating failed SSH password-based authentication attempts for both incorrect password and invalid username scenarios.

11.1.3.1.2 FH M-Plane SSH-certificate-based authentication authorization

Requirement Name: M-Plane authenticity protection over FH interface using SSH

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specification [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_MPLANE_SSH-CERTIFICATE-BASED_AUTHENTICATION_AUTHORIZATION

Purpose: The purpose of this test is to verify the SSH-certificate-based authentication and authorization mechanisms on the front-haul (FH) interface between O-RU and O-DU, using test equipment as needed to simulate either party.

Procedure and execution steps

NOTE: Test equipment may simulate the role of O-DU or O-RU for the purpose of this test.

Preconditions

1. The O-RU and O-DU devices are properly configured and operational.
2. Test equipment capable of simulating SSH client/server functionality is prepared to represent either the O-DU or O-RU as required.

3. SSH keys and certificates are generated and installed on both the O-RU and O-DU devices.
4. NACM with NETCONF is enabled and configured for authorization on the FH interface.
5. SSH is properly implemented and configured as defined in [2] clause 4.1.

Execution steps

Execute the test on the SSH protocol as defined in clause 6.2.

Part A: Authentication and authorization of O-DU by O-RU (or test equipment simulating O-DU)

- Positive Case: Successful SSH-certificate-based authentication and authorization.
  - Establish an SSH connection from the O-RU to the O-DU using the SSH key and certificate.
  - Verify that the O-RU successfully authenticates the O-DU using the SSH certificate.

  EXAMPLE: "“Command: **show ssh sessions**"
  - Validate that the O-DU is authorized to perform the requested operations on the FH interface.
    - Perform an operation on the FH interface that requires authorization. This operation should be within the scope of permitted actions for the authenticated O-DU.

  EXAMPLE of operations: "start up" installation, software management, configuration management, performance management, fault management and file management towards the O-RU
    - Monitor the responses from the O-RU to these operations.
    - Record whether each operation was successfully executed, partially executed, or rejected.
    - Verify the O-RU logs to confirm that the operations were authorized.
- Negative Case: Failed SSH-certificate-based authentication.
  - Test the handling of failed SSH-certificate-based authentication attempts by the O-RU in different scenarios.
    - Attempt with invalid key or certificate
      - Attempt to establish an SSH connection using an incorrect or invalid SSH key or certificate.

EXAMPLE: "Command: **ssh -i <path_to_invalid_private_key> -o CertificateFile=<path_to_invalid_certificate> <valid_username>@<O-RU_IP>**"

      - Verify that the O-RU rejects the SSH connection due to the authentication failure.
    - Attempt with invalid username
      - Attempt to establish an SSH connection using a valid SSH key and certificate, but with a username that does not exist in the O-RU's system.

EXAMPLE: "Command: **ssh -i <path_to_valid_private_key> -o CertificateFile=<path_to_valid_certificate> <invalid_username>@<O-RU_IP>**"

      - Verify that the O-RU rejects the SSH connection, confirming that the system does not authenticate usernames that are not registered or recognized.

Part B: Authentication of O-RU by O-DU (or test equipment simulating O-RU)

- Positive Case: Successful SSH-certificate-based authentication:
  - Establish an SSH connection using the SSH key and certificate.

- Verify that the O-DU successfully authenticates the O-RU using the SSH certificate.

EXAMPLE: "Command: **show ssh sessions**"

- Negative Case: Failed SSH-certificate-based authentication.
  - Test the handling of failed SSH-certificate-based authentication attempts by the O-DU in different scenarios.
    - Attempt with invalid key or certificate
      - Attempt to establish an SSH connection using an incorrect or invalid SSH key or certificate.

EXAMPLE: "Command: **ssh -i <path_to_invalid_private_key> -o CertificateFile=<path_to_invalid_certificate> <valid_username>@<O-DU_IP>**"

      - Verify that the O-RU rejects the SSH connection due to the authentication failure.
    - Attempt with invalid username
      - Attempt to establish an SSH connection using a valid SSH key and certificate, but with a username that does not exist in the O-RU's system.

EXAMPLE: "Command: **ssh -i <path_to_valid_private_key> -o CertificateFile=<path_to_valid_certificate> <invalid_username>@<O-DU_IP>**"

      - Verify that the O-DU rejects the SSH connection, confirming that the system does not authenticate usernames that are not registered or recognized.

Expected Results

For step 1): Expected results in clause 6.2

For Parts A and B – Positive Case:

- The SSH connection is successfully established using the correct SSH key and certificate.
- The DUT (O-RU or O-DU) validates the test equipment's SSH certificate for authentication.
- The O-RU grants the necessary authorization to the O-DU for the requested operations.
- The SSH connection attempt fails due to the incorrect or invalid SSH key or certificate.
- The DUT identifies the authentication failure and denies access accordingly.

Expected format of evidence

For step 1): Logs and screenshots showing adherence to SSH protocol specifications as defined in [2] clause 4.1.

For Parts A and B – Positive Case: Logs showing successful SSH authentication and authorization events.

For Parts A and B – Negative Case: Logs or error messages indicating failed SSH-certificate-based authentication attempts for both invalid key/certificate and non-existent username scenarios.

11.1.3.1.3 FH M-plane SSH Certificate-Based NACM Access Control

Requirement Name: M-Plane access control protection over FH interface using SSH

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specification [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU

Test Name: TC_FH_MPLANE_SSH-CERTIFICATE-BASED_NACM_ACCESS_CONTROL

Purpose: The purpose of this test is to verify the SSH-certificate-based NACM access control on the FH interface between O-RU and O-DU.

Procedure and execution steps

Preconditions

1. NACM with NETCONF is enabled and configured for SSH-certificate-based authorization on the FH interface.
2. Access control rules and permissions are defined and configured on both the O-RU and O-DU.
3. SSH is properly implemented and configured as defined in [2] clause 4.1.

Execution steps

1. Execute the test on the SSH protocol as defined in clause 6.2.
2. Positive Case: Successful SSH-certificate-based NACM authorization and access control.
   - Test the successful enforcement of SSH-certificate-based NACM policies on the FH interface.
     a) Establish an SSH connection using the SSH key and certificate.
     b) Perform an operation on the FH interface with the O-RU using the SSH connection.
     c) Verify that the O-RU grants or denies access based on the SSH-certificate-based NACM rules and permissions.
3. Negative Case: Unauthorized access denial.
   - Test the denial of access to unauthorized operations on the FH interface, including attempts with invalid credentials and invalid usernames.
     a) Attempt with invalid key or certificate
        - Attempt to establish an SSH connection using an invalid key or certificate..
        - Confirm that the O-RU denies the SSH connection due to invalid credentials.
     b) Attempt with invalid username
        - Attempt to establish an SSH connection using a valid SSH key and certificate but with an invalid username.
        - Verify that the O-RU denies the SSH connection attempt due to the invalid username.

Expected Results

1. For step 1): Expected results in clause 6.2
2. For step 2):
   - The SSH connection is successfully established using the SSH key and certificate.
   - The O-RU evaluates the SSH certificate-based NACM rules and permissions.
   - The O-RU grants or denies access to the O-DU based on the SSH-certificate-based NACM configuration.
3. For step 3):
   - Denial of SSH connection due to invalid key or certificate.
   - Denial of SSH connection due to an invalid username.

Expected format of evidence

1. For step 1): Logs and screenshots showing adherence to SSH protocol specifications as defined in [2] clause 4.1.
2. For step 2), Logs or audit records indicating both successful access and access denial based on SSH-certificate-based NACM.
3. For step 3), Logs or error messages indicating access denial for unauthorized operations for both invalid credentials (key/certificate) and invalid usernames.

11.1.3.2 SSH-based M-Plane integrity, confidentiality and replay protection

The following test cases verify the M-Plane integrity, confidentiality, and replay protection over the FH interface using SSH.

Requirement Name: M-Plane confidentiality, integrity and replay protection over FH M-Plane interface using SSH

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specifications [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_MPLANE_SSH_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the FH M-Plane interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection using SSH.

Procedure and execution steps

Preconditions

1. The O-RU and O-DU devices are properly configured and operational.
2. SSH keys and certificates are generated and installed on both the O-RU and O-DU devices.
3. SSH configuration is enabled to enforce confidentiality, integrity and replay protection on the FH M-plane interface.
4. SSH is properly implemented and configured as defined in [2] clause 4.1.

Execution steps

1. Confidentiality verification:
   - Establish an SSH connection between the O-RU and O-DU using proper SSH keys and certificates.
   - Transmit data over this connection.
   - Capture and analyze the transmitted data to verify encryption, ensuring confidentiality.
2. Integrity protection verification:
   - During the same SSH session, modify the transmitted packets midway.
   - Attempt to deliver the modified packets to the DUT.
   - Verify that the DUT detects and discards these packets.
3. Replay protection verification:
   - Replay previously captured packets to the DUT within the same SSH session.

- Confirm that the DUT detects and discards replayed packets.

Expected Results

- Confidentiality: All sensitive data transmitted over the FH M-Plane interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring the data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected format of evidence

- Logs or screenshots showing SSH protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.1.
- Evidence of secure communication sessions established over the FH M-Plane interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.1.3.3 TLS-based M-Plane authentication, authorization and access control protection

11.1.3.3.0 Introduction

The test cases outlined in this clause verify M-Plane authenticity, authorization, and access control protection over the FH interface using TLS.

11.1.3.3.1 FH M-plane TLS Authentication

Requirement Name: M-Plane authenticity protection over FH interface using TLS

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specification [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_MPLANE_TLS_AUTHENTICATION

Purpose: The purpose of this test is to verify the authentication mechanism between the O-RU and O-DU components over the TLS-based NACM with NETCONF on the FH interface for M-Plane.

Procedure and execution steps

Preconditions

1. For positive case: The O-RU and O-DU components are configured with valid TLS certificates for mutual authentication.
2. For negative case: The O-RU and O-DU components have misconfigured or invalid TLS certificates.
3. The NETCONF server is configured to enforce client authentication as defined in [2] clause 4.3.
4. TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Execute the test on the TLS protocol as defined in clause 6.3.
2. Positive Case: Successful authentication.

- Test the successful authentication of the O-RU and O-DU components over the TLS-based NACM with NETCONF on the FH interface.

  a) Initiates a TLS handshake.

  - Use a command or tool that starts a TLS session

  b) Observe and validate that the O-RU checks the O-DU's certificate.

  - Check O-RU logs or use network monitoring tools to confirm certificate verification and presentation.

  c) Confirm that the O-RU successfully verifies the O-DU's certificate, completing the mutual authentication.

  - Review O-RU logs or use network monitoring tools to confirm the certificate verification.

3. Negative Case: Failed authentication.

- Test the failure of authentication over the TLS-based NACM with NETCONF on the FH interface due to invalid certificates.

  a) Initiate a TLS handshake from the O-DU with an invalid certificate.

  b) Check O-RU logs or network monitoring tools to observe the certificate verification attempt .

  c) Confirm that the TLS handshake fails and mutual authentication is not completed.

  - Look for error messages or handshake failure indicators in the network traffic or logs.

Expected Results

1. For step 1): Expected results in clause 6.3
2. For step 2), The O-RU and O-DU components successfully authenticate each other over the TLS-based NACM with NETCONF on the FH interface.
3. For step 3), The O-RU and O-DU components fail to authenticate each other over the TLS-based NACM with NETCONF on the FH interface.

Expected format of evidence

1. For step 1): Logs and screenshots showing adherence to TLS protocol specifications as defined in [2] clause 4.2.
2. For step 2), Logs or output indicating successful authentication.
3. For step 3), Logs or output indicating failed authentication.

11.1.3.3.2 FH M-plane TLS Authorization

Requirement Name: M-Plane authorization and access control protection over FH interface using TLS

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specification [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, T-FRHAUL-02, T-MPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_MPLANE_TLS_AUTHORIZATION

Purpose: The purpose of this test is to verify the authorization mechanism for the O-RU and O-DU components over the TLS-based NACM with NETCONF on the FH interface.

Procedure and execution steps

Preconditions

1. The O-RU and O-DU components are successfully authenticated and have established a secure connection with the NETCONF server.
2. For positive case: The NACM rules and policies are properly configured on the NETCONF server to enforce authorization.
3. For negative case: The NACM rules and policies are misconfigured, or the O-RU is attempting an unauthorized operation.

Execution steps

1. Positive Case: Successful authorization.
    - Test the successful authorization of the O-RU and O-DU components over the TLS-based NACM with NETCONF on the FH interface.
        a) The O-RU sends a NETCONF request to the O-DU component to perform an authorized operation.
        b) The NETCONF server evaluates the NACM rules and policies to determine if the O-RU is authorized to perform the requested operation.
        c) The O-DU component executes the authorized operation and sends a response to the O-RU.
2. Negative Case: Failed authorization.
    - Test the failure of authorization for the O-RU and O-DU components over the TLS-based NACM with NETCONF on the FH interface.
        a) The O-RU sends a NETCONF request to the O-DU component to perform an unauthorized operation.
        b) The NETCONF server evaluates the NACM rules and policies and denies the unauthorized operation.
        c) The O-DU component rejects the unauthorized operation and sends an error response to the O-RU.

Expected Results

1. For step 1) Positive case: successful authorization:
    a) The O-RU's NETCONF request for an authorized operation is successfully received by the O-DU.
    b) The NETCONF server, after evaluating the NACM rules and policies, grants permission for the authorized operation.
    c) The O-DU successfully executes the authorized operation and sends a confirmation response to the O-RU.
2. For step 2) Negative case: failed authorization:
    a) The O-RU's NETCONF request for an unauthorized operation is received by the O-DU.
    b) The NETCONF server, upon evaluating the NACM rules and policies, denies the unauthorized operation.
    c) The O-DU does not execute the unauthorized operation and sends an error response to the O-RU, indicating the rejection.

Expected format of evidence

1. For step 1), Logs or output indicating successful authorization.
2. For step 2), Logs or output indicating failed authorization.

11.1.3.4 TLS-based M-Plane integrity, confidentiality and replay protection

Requirement Name: M-Plane confidentiality, integrity, and replay protection over FH M-plane interface using TLS

Requirement Reference: Clause 5.4 in O-RAN WG4 Management Plane Specifications [21]

Requirement Description:

Threat References: T-O-RAN-05, T-FRHAUL-01, 02, T-MPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_MPLANE_TLS_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the FH M-Plane interface, ensuring that M-Plane data remains protected through confidentiality, integrity, and replay protection using TLS.

Procedure and execution steps

Preconditions

1. O-RU, O-DU support TLS and be connected in simulated/real network environment.
2. The FH M-Plane interface is configured for testing.
3. TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
   - Establish a secure communication session over the FH M-Plane interface.
   - Capture the network traffic during the session.
   - Analyse the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
   - Capture protected packets after the TLS handshake.
   - Modify the captured packets.
   - Inject the modified packets to the DUT.
   - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
   - Capture protected packets after the TLS handshake.
   - Replay the captured packets to the DUT.
   - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the Y1 interface is encrypted, with no data exposed in clear text.

- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected format of evidence

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the Y1 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.1.4 U-Plane

11.1.4.1 U-Plane eCPRI Unexpected Input

11.1.4.1.0 Introduction

The test cases in this clause focus on the O-DU's capability to recognize, handle, and respond appropriately to such anomalies in user plane packets over the eCPRI. This includes scenarios where packets are malformed or when they present unexpected payload sizes.

11.1.4.1.1 FH U-Plane Malformed Packet

Requirement Name: Handling and rejection of malformed or invalid user plane packets

Requirement Reference: Clause 5.2.5.2.1 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-FRHAUL-01, T-FRHAUL-02, T-UPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_U-PLANE_MALFORMED_PACKET

Purpose: The purpose of this test is to verify the O-DU's ability to handle and reject malformed or invalid user plane packets.

Procedure and execution steps

Preconditions

- A valid eCPRI connection between the O-RU and O-DU.

Execution steps

1. Generate a user plane packet with invalid or malformed data, such as incorrect headers, corrupted payload, or unsupported formats.
2. Transmit the malformed packet over the eCPRI.
3. Monitor the O-DU's response and behaviour.
4. Verify that the O-DU identifies and rejects the malformed packet.
5. Observe the impact on the O-DU, such as error messages, logging, or abnormal behaviour.

Expected Results

- The O-DU detects and rejects malformed or invalid user plane packets.
- It handles the rejection gracefully without affecting normal operation.
- Appropriate error messages or log entries are generated.

Expected Format of Evidence:

- Steps performed with detailed execution logs.
- Screenshots or logs indicating the detection and rejection of the malformed packet.

11.1.4.1.2 FH U-Plane Unexpected Payload Size

Requirement Name: Handling and rejection of malformed or invalid user plane packets

Requirement Reference: Clause 5.2.5.2.1 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-FRHAUL-01, T-FRHAUL-02, T-UPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_FH_U-PLANE_UNEXPECTED_PAYLOAD_SIZE

Purpose: The purpose of this test is to verify the O-DU's ability to handle unexpected payload sizes in user plane packets.

Procedure and execution steps

Preconditions

- A valid eCPRI connection between the O-RU and O-DU.

Execution steps

1. Generate a user plane packet with an unexpected payload size, exceeding the normal or allowed range.
2. Transmit the packet with the unexpected payload size over the eCPRI.
3. Monitor the O-DU's response and behaviour.
4. Verify that the O-DU detects the unexpected payload size and takes appropriate action.
5. Observe the impact on the O-DU, such as error handling, packet drops, or performance degradation.

Expected Results

- The O-DU detects and handles unexpected payload sizes in user plane packets.
- It either rejects the packet or handles it with appropriate error handling mechanisms.
- The O-DU maintains acceptable performance levels despite the unexpected payload size.

Expected Format of Evidence:

- Steps performed with detailed execution logs.
- Screenshots or logs indicating the detection and handling of the unexpected payload size.

11.1.5 S-Plane

11.1.5.1 DoS Attack against a Master Clock

11.1.5.1.0 Introduction

The tests outlined in this clause evaluate the system's defense capabilities against DoS attacks targeting the master clock, especially in different LLS configurations.

11.1.5.1.1 DOS Master Clock LLS C1 C2 C3

Requirement Name: S-Plane DoS protection

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_DOS_MASTER_CLOCK_LLS_C1_C2_C3

Purpose: The purpose of this test is to verify the protection of the S-plane against a denial of service (DoS) attack targeting the master clock in LLS-C1, LLS-C2, LLS-C3 configurations.

Procedure and execution steps

Preconditions

1. O-DU and O-RU are properly configured and connected.
2. For LLS-C1: The master clock functionality is enabled on the O-DU. O-DU is acting as a master and directly synchronizes O-RU.
3. For LLS-C2: One or more Ethernet switches are allowed in the fronthaul network. O-DU acting as master to distribute network timing toward O-RU.
4. For LLS-C3: One or more PRTC/T-GM are implemented in the fronthaul network to distribute network timing toward O-DU and O-RU.
5. A network monitoring tool is set up to capture and analyse network traffic.

Execution steps

1. Start monitoring the network traffic between the O-DU and O-RU.
2. Simulate a DoS attack by sending an excessive number of time protocol packets to the master clock using a testing tool.
    - Simulate DoS attack for LLS-C1
        - Use a command-line tool like **ptp4l** or **pgrptp** with appropriate options to flood the Master clock's IP address or hostname with an excessive number of time protocol packets.
    - Simulate DoS attack for LLS-C2
        - Use a custom script or tool that supports PTP communication to generate and send a large volume of time protocol packets targeting the IP address or hostname of the Master clock.

- Simulate DoS attack for LLS-C3
  - Use a custom script or tool that supports PTP communication to generate and send a large volume of time protocol packets targeting the PRTC/T-GM in the LLS-C3 configuration.

3. Verify the functionality of the master clock and the synchronization status between the O-DU and O-RU during the attack.
4. Observe the impact on the accuracy and availability of the master clock.
5. Verify the functionality of the slave clocks at the O-RUs and their synchronization status with the master clock during the attack.
6. Evaluate the impact on O-RUs relying on accurate timing information:
   - Measure the timing accuracy at the O-RUs before initiating the DoS attack to establish a baseline.
   - During the DoS attack, continuously monitor the timing accuracy at the O-RUs at regular intervals (e.g., every 10 seconds).
   - Compare the timing accuracy measurements taken during the attack to the baseline measurements.
   - Identify any deviations or discrepancies in timing accuracy that exceed acceptable thresholds.
   - Document any observed impact on O-RU operations that rely on precise timing, such as frame alignment, data transmission synchronization, or other time-sensitive processes.
   - After the DoS attack has concluded, continue monitoring the O-RUs to determine how quickly they recover and return to their baseline timing accuracy.

Expected Results

1. The S-plane detects and mitigates the DoS attack against the master clock for each LLS configuration (C1, C2, C3).
2. The master clock continues to operate with minimal impact on accuracy and availability.
3. The synchronization status between the O-DU and O-RU remains stable.
4. The slave clocks maintain synchronization with their respective master clocks, although some minor degradation may be expected.
5. O-RUs relying on accurate timing information should continue to function, although some degradation may be observed during the attack.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided for each LLS configuration (C1, C2, C3):

1. Network traffic logs showing the excessive time protocol packets sent to the mater clock during the attack in LLS-C1, through Ethernet switches in LLS-C2 and targeting PRTC/T-GM in LLS-C3.
2. Monitoring reports indicating the behaviour of the master clock and synchronization status between the O-DU and O-RU during the attack.
3. Analysis of the impact on the accuracy and availability of the master clock.
4. Evaluation of the synchronization status of the slave clocks during the attack.

11.1.5.1.2 DOS Master Clock LLS C4

Requirement Name: S-Plane DoS protection

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-01

DUT/s: O-RU, O-DU

Test Name: TC_DOS_MASTER_CLOCK_LLS_C4

Purpose: The purpose of this test is to verify the protection of the S-plane against a denial of service (DoS) attack targeting the local PRTC timing in an LLS-C4 configuration.

Procedure and execution steps

Preconditions

1. The O-RU is configured with a local PRTC timing that provides time synchronization.
2. Local PRTC timing is enabled that provides time synchronization to the O-RU (it could be embedded in the O-RU).
3. A network monitoring tool is set up to capture and analyse network traffic.

Execution steps

1. Start monitoring the network traffic between the O-RU and the fronthaul network.
2. Simulate a DoS attack by sending an excessive number of time protocol packets to the O-RU's local PRTC timing using a testing tool.
3. Monitor the behaviour of the local PRTC timing and the synchronization status between the O-RU and the fronthaul network during the attack.
4. Observe the impact on the accuracy and availability of the local PRTC timing.
5. Verify the functionality of the O-RU during the attack to ensure that it can still operate normally despite the DoS attack on the local PRTC timing.

Expected Results

1. The S-plane detects and mitigates the DoS attack against the local PRTC timing in the O-RU.
2. The local PRTC timing continues to operate with minimal impact on accuracy and availability.
3. The synchronization status between the O-RU and the fronthaul network should remain stable.
4. The O-RU should continue to function normally, even with the DoS attack targeting the local PRTC timing.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

1. Network traffic logs showing the excessive time protocol packets sent to the local PRTC timing during the attack.
2. Monitoring reports indicating the behaviour of the local PRTC timing and synchronization status between the O-RU and the fronthaul network during the attack.
3. Analysis of the impact on the accuracy and availability of the local PRTC timing.

11.1.5.2 Spoofing of Master Clocks in the S-Plane

The tests presented in this clause focus on assessing the system's defenses against potential spoofing attacks on master clocks. Specifically, these tests examine scenarios where attackers may try to impersonate or manipulate the master clock's communications to disrupt accurate time synchronization.

11.1.5.2.1 Impersonation Master Clock

Requirement Name: Spoofing Prevention for Master Clocks in the S-Plane

Requirement Reference: REQ-SEC-OFSP-2, clause 5.2.5.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-02, T-SPLANE-03

DUT/s: O-RU, O-DU

Test Name: TC_IMPERSONATION_MASTER_CLOCK

Purpose: The purpose of this test is to verify the protection of the S-plane against an impersonation attack where an attacker sends fake ANNOUNCE messages to declare itself as the best clock (Grand Master).

Procedure and execution steps

Preconditions

1. For LLS-C1: The master clock functionality is enabled on the O-DU. O-DU is acting as a master and directly synchronizes O-RU.
2. For LLS-C2: One or more Ethernet switches are allowed in the fronthaul network. O-DU acting as master to distribute network timing toward O-RU.
3. For LLS-C3: One or more PRTC/T-GM are implemented in the fronthaul network to distribute network timing toward O-DU and O-RU.
4. For LLS-C4: Local PRTC timing is enabled that provides time synchronization to the O-RU (it could be embedded in the O-RU).
5. The master clock functionality of the O-DU is enabled and functioning correctly (not applicable in the LLS-C4 configuration).
6. A network monitoring tool is set up to capture and analyse network traffic.

Execution steps

1. Start monitoring the network traffic between the O-DU and O-RU.
2. Simulate an impersonation attack by sending a fake ANNOUNCE message declaring a different clock as the best clock in the network to the O-DU using a testing tool.
    - For LLS-C1, use a command-line tool like **ptp4l** or **pgrptp** with appropriate options to send fake ANNOUNCE messages to the IP address or hostname of the O-DU acting as the legitimate Master clock.
    - For LLS-C2, use a PTP simulation tool like **pysimulatedptp** or **ptpd** to generate fake ANNOUNCE messages with the attacker's clock information, targeting the IP address or hostname of the O-DU acting as the legitimate Master clock.

- For LLS-C3, use a custom script or tool that supports PTP communication to craft and send fake ANNOUNCE messages to the IP addresses or hostnames of the PRTC/T-GM devices within the fronthaul network.
- For LLS-C4, use a PTP simulation tool or a custom script that can craft and send PTP ANNOUNCE messages that impersonates a legitimate PRTC or Grand Master clock, declaring a different clock (controlled by the attacker) as the best clock, and target this message to the O-RU's IP address or hostname.

3. Verify the functionality of the O-DU and O-RU upon receiving the fake ANNOUNCE message.
4. Observe the synchronization status between the O-DU and O-RU.
5. Verify that the O-DU and O-RU reject the impersonated clock and maintain the synchronization based on the legitimate master clock.

Expected Results

1. The S-plane detects and mitigates the impersonation attack by recognizing the fake ANNOUNCE message.
2. The O-DU and O-RU reject the impersonated clock and maintain synchronization with the legitimate master clock.
3. The synchronization status between the O-DU and O-RU remains stable and accurate.
4. The O-RU continues to receive accurate timing information from the legitimate master clock.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided for each configuration (LLS-C1, LLS-C2, LLS-C3, LLS-C4):

1. Network traffic logs showing the transmission of the fake ANNOUNCE message to the O-DU. This includes logs for direct transmission to the O-DU (LLS-C1), through Ethernet switches (LLS-C2), to PRTC/T-GM devices (LLS-C3), and to O-RU with local PRTC (LLS-C4).
2. Monitoring reports indicating the behaviour of the O-DU and O-RU upon receiving the fake ANNOUNCE message.
3. Analysis of the synchronization status between the O-DU and O-RU.
4. Verification that the O-DU and O-RU reject the impersonated clock and maintain synchronization with the legitimate master clock.

11.1.5.2.2 Rogue PTP Instance

Requirement Name: Spoofing Prevention for Master Clocks in the S-Plane

Requirement Reference: REQ-SEC-OFSP-2, clause 5.2.5.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-02, T-SPLANE-03

DUT/s: O-RU, O-DU

Test Name: TC_ROGUE_PTP_INSTANCE

Purpose: The purpose of this test is to verify the protection of the S-plane against an attacker sending manipulated or malicious ANNOUNCE messages to declare itself as the best clock (Grand Master).

Procedure and execution steps

Preconditions

1. For LLS-C1: The master clock functionality is enabled on the O-DU. O-DU is acting as a master and directly synchronizes O-RU.
2. For LLS-C2: One or more Ethernet switches are allowed in the fronthaul network. O-DU acting as master to distribute network timing toward O-RU.
3. For LLS-C3: One or more PRTC/T-GM are implemented in the fronthaul network to distribute network timing toward O-DU and O-RU.
4. For LLS-C4: Local PRTC timing is enabled that provides time synchronization to the O-RU (it could be embedded in the O-RU).
5. The O-DU and O-RU are synchronized and functioning correctly.
6. A network monitoring tool is set up to capture and analyse network traffic.

Execution steps

1. Start monitoring the network traffic between the O-DU and O-RU.
2. Simulate an attack by injecting manipulated or malicious ANNOUNCE messages declaring the attacker as the best clock in the network by sending manipulated or malicious ANNOUNCE messages impersonating a Grand Master clock.
    - For LLS-C1, use a command-line tool like **ptp4l** or **pgrptp** with appropriate options to send manipulated ANNOUNCE messages to the IP address or hostname of the O-DU acting as the legitimate Master clock.
    - For LLS-C2, use a PTP simulation tool like **pysimulatedptp** or **ptpd** to generate manipulated ANNOUNCE messages with the attacker's clock information, targeting the IP address or hostname of the O-DU acting as the legitimate Master clock.
    - For LLS-C3, use a custom script or tool that supports PTP communication to craft and send manipulated ANNOUNCE messages to the IP addresses or hostnames of the PRTC/T-GM devices within the fronthaul network.
    - For LLS-C4, if the Master clock is embedded in the O-RU, simulate the attack by sending manipulated ANNOUNCE messages directly to the O-RU.
3. Verify the functionality of the O-DU and O-RU upon receiving the manipulated or malicious ANNOUNCE messages.
4. Observe the synchronization status between the O-DU and O-RU.
5. Verify that the O-DU and O-RU detect and reject the attacker's proposed grandmaster candidate.

Expected Results

1. The S-plane detects and mitigates the attack by recognizing the manipulated or malicious ANNOUNCE messages.
2. The O-DU and O-RU reject the attacker's proposed grandmaster candidate and maintain synchronization based on the legitimate master clock.
3. The synchronization status between the O-DU and O-RU remains stable and accurate.
4. The O-RU continues to receive accurate timing information from the legitimate master clock.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided for each configuration (LLS-C1, LLS-C2, LLS-C3, LLS-C4):

1. Network traffic logs showing the transmission of the manipulated or malicious ANNOUNCE messages. These logs should demonstrate the attack simulation for LLS-C1 (O-DU as master), LLS-C2 (with Ethernet switches), LLS-C3 (with PRTC/T-GM), and LLS-C4 (local PRTC timing in O-RU).
2. Monitoring reports indicating the behaviour of the O-DU and O-RU upon receiving the manipulated or malicious ANNOUNCE messages.
3. Analysis of the synchronization status between the O-DU and O-RU.
4. Verification that the O-DU and O-RU reject the attacker's proposed grandmaster candidate and maintain synchronization with the legitimate master clock.

11.1.5.3 Clock Accuracy Protection Against MITM Attacks

This clause delves into tests specifically designed to gauge the system's robustness when facing MITM attacks targeting clock synchronization. Such MITM attacks could manifest as the selective interception and removal of crucial PTP timing packets or the deliberate introduction of delays to these packets.

11.1.5.3.1 Selective Interception and Removal of PTP Timing Packets

Requirement Name: Clock Accuracy Protection Against MITM Attacks

Requirement Reference: REQ-SEC-OFSP-3, clause 5.2.5.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-04, T-SPLANE-05

DUT/s: O-RU, O-DU

Test Name: TC_SELECTIVE_INTERCEPTION_REMOVAL_PTP_TIMING_PACKETS

Purpose: The purpose of this test is to verify the resilience of the S-plane against an attack where PTP timing packets are selectively intercepted and removed.

Procedure and execution steps

Preconditions

1. For LLS-C1: The master clock functionality is enabled on the O-DU. O-DU is acting as a master and directly synchronizes O-RU.
2. For LLS-C2: One or more Ethernet switches are allowed in the fronthaul network. O-DU acting as master to distribute network timing toward O-RU.
3. For LLS-C3: One or more PRTC/T-GM are implemented in the fronthaul network to distribute network timing toward O-DU and O-RU.
4. For LLS-C4: Local PRTC timing is enabled that provides time synchronization to the O-RU (it could be embedded in the O-RU).
5. The network monitoring tool is set up to capture and analyse network traffic between the O-RU and O-DU.

Execution steps

1. Set up the test environment with the O-RAN O-RU, O-DU, and other relevant network components.

2. Configure the network monitoring tool to capture PTP timing packets between the O-RU and O-DU.
3. Start the network monitoring tool to capture the initial state of PTP timing packets.
4. Simulate an attack by using a packet manipulation tool to selectively intercept and remove specific PTP timing packets.
    - For LLS-C1, use a packet capture tool like Wireshark or tcpdump to capture PTP network traffic on the interface connected to the O-RU or O-DU. Modify the captured packets to selectively remove PTP timing packets using a packet editing tool like Scapy or custom scripts.
    - For LLS-C2, use a network device or software with packet interception capabilities to intercept PTP timing packets between the O-RU and O-DU. Modify the intercepted packets to selectively remove PTP timing packets.
    - For LLS-C3, use a network device or software capable of deep packet inspection (DPI) to intercept and analyse PTP timing packets. Modify the intercepted packets to selectively remove PTP timing packets.
    - For LLS-C4, if the O-RU embeds the local PRTC timing, use a network device or software to intercept PTP timing packets between the O-RU and O-DU. Modify the intercepted packets to selectively remove PTP timing packets.
5. Verify the functionality of the O-RU and O-DU during the attack simulation.
6. Observe the synchronization status and the impact on timing accuracy between the O-RU and O-DU.
7. Capture and analyse the network traffic using the network monitoring tool during the attack simulation.

    NOTE: The network monitoring tool can be Wireshark or tcpdump, configured to capture packets on the interfaces between the O-RU, O-DU and to identify the intercepted and removed PTP timing packets.

8. Stop the network monitoring tool to finalize the captured traffic.

Expected Results

1. Detection of missing PTP timing packets: The S-plane is able to detect the absence of specific PTP timing packets that were selectively intercepted and removed.
2. Synchronization maintenance: Despite the missing PTP timing packets, the O-RU and O-DU still maintain synchronization. Any deviations from expected synchronization are minimal and within acceptable thresholds.
3. Corrective actions: Upon detecting the missing PTP timing packets, the O-RU and O-DU initiate predefined corrective actions to restore synchronization and mitigate the effects of the missing packets.
4. Network traffic analysis: The captured network traffic clearly shows the instances where specific PTP timing packets were intercepted and removed.
5. No system failures: The system (O-RU and O-DU) doesn’t experience any catastrophic failures or shutdowns due to the missing PTP timing packets.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided for each configuration (LLS-C1, LLS-C2, LLS-C3, LLS-C4):

1. Recorded network traffic captured by the monitoring tool during the attack simulation showing selective interception and removal of PTP timing packets in LLS-C1 (O-DU as master), LLS-C2 (with Ethernet switches), LLS-C3 (with PRTC/T-GM), and LLS-C4 (local PRTC timing in O-RU).
2. Observations and analysis of the impact on synchronization and timing accuracy.

3. Any issues or anomalies encountered during the attack simulation.

11.1.5.3.2 Delay Attack on PTP Timing Packets

Requirement Name: Clock Accuracy Protection Against MITM Attacks

Requirement Reference: REQ-SEC-OFSP-3, clause 5.2.5.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-SPLANE-04, T-SPLANE-05

DUT/s: O-RU, O-DU

Test Name: TC_DELAY_ATTACK_PTP_TIMING_PACKETS

Purpose: The purpose of this test is to verify the S-plane's resilience against a delay attack on PTP timing packets.

Procedure and execution steps

Preconditions

1. For LLS-C1: The master clock functionality is enabled on the O-DU. O-DU is acting as a master and directly synchronizes O-RU.
2. For LLS-C2: One or more Ethernet switches are allowed in the fronthaul network. O-DU acting as master to distribute network timing toward O-RU.
3. For LLS-C3: One or more PRTC/T-GM are implemented in the fronthaul network to distribute network timing toward O-DU and O-RU.
4. For LLS-C4: Local PRTC timing is enabled that provides time synchronization to the O-RU (it could be embedded in the O-RU).
5. Time synchronization is established and operational within the network.

Execution steps

1. Start the network monitoring tool to capture the initial state of PTP timing packets.
2. Simulate an attack by introducing delays in PTP timing packets using a network emulation tool.
    - For LLS-C1, use a network emulator tool like WANem or NIST Net to introduce artificial delays in PTP timing packets between the O-RU and O-DU.
    - For LLS-C2 and LLS-C3, use a custom script or tool that supports packet manipulation and delay to introduce artificial delays in PTP timing packets between the O-RU and O-DU or between PRTC/T-GM devices.
    - For LLS-C4, if the O-RU embeds the local PRTC timing, use a network emulator tool or custom script to introduce delays in PTP timing packets between the O-RU and O-DU.
3. Verify the functionality of the O-RU and O-DU during the delay attack on PTP timing packets.
4. Observe the synchronization status and timing accuracy within the LLS configuration.

Expected Results

1. The S-plane detects the delay attack on PTP timing packets and applies appropriate measures to mitigate the impact within all LLS configurations.

2. The O-RU and O-DU detects the delayed PTP timing packets, compensate for the introduced delays, and maintain synchronization.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided for each configuration (LLS-C1, LLS-C2, LLS-C3, LLS-C4):

1. Recorded network traffic captured by the monitoring tool during the attack. This includes logs showing the introduction of delays in PTP timing packets for LLS-C1 (O-DU as master), LLS-C2 (with Ethernet switches), LLS-C3 (with PRTC/T-GM), and LLS-C4 (local PRTC timing in O-RU).
2. Observations and analysis of the impact on synchronization and timing accuracy within each LLS configuration.
3. Any issues or anomalies encountered during the attack simulation.

11.2 Y1

11.2.1 Y1 Authenticity

Requirement Name: Y1 protection in terms of authenticity

Requirement Reference: SEC-CTL-NEAR-RT-9, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-Y1-01, T-Y1-02, T-Y1-03

DUT/s: Near-RT RIC, Y1 consumers

Test Name: TC_Y1_AUTHENTICATION

Purpose: The purpose of this test is to verify the authenticity of the Y1 interface, ensuring that only legitimate and mutually authenticated Near-RT RIC, Y1 consumers can participate in the communication over the Y1 interface.

Procedure and execution steps

Preconditions

1. Near-RT RIC & Y1 Consumers support mTLS and be connected in a simulated/real network environment.
2. The test environment is set up with the Y1 interface configured.
3. The tester has access to the original data transported over the Y1 interface.
4. mTLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Execute the test on the mTLS protocol as defined in clause 6.3.
2. Valid Authentication Certificates (positive case):
    a. The tester sends a request to establish a connection with the Y1 interface using valid authentication certificates.
    b. The tester verifies the mutual certificate verification between Near-RT RIC and Y1 consumers.
    c. The tester captures and analyses the response received from the Y1 interface.

3. Invalid Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection with the Y1 interface with invalid certificates.
    b. The tester captures and analyses the response received from the Y1 interface.
4. No Authentication Certificates (Negative Case):
    a. The tester sends a request to establish a connection without any certificates.
    b. The tester captures and analyses the response from the Y1 interface.

Expected results

1. For 1. Expected results in clause 6.3
2. For 2. 'Valid Authentication Certificates': The Y1 interface accepts the valid certificates and responds with a successful authentication message. The mutual certificate verification process is successful.
3. For 3. 'Invalid Authentication Certificates': The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the use of invalid certificates.
4. For 4. 'No Authentication Certificates': The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the absence of certificates.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:
- Logs and screenshots showing adherence to mTLS protocol specifications as defined in [2] clause 4.2.
- Logs of authentication requests and responses on the Y1 interface
- Logs of the mutual certificate verification process.
- Screenshots or logs of error messages or unusual behaviours for both invalid and no certificate scenarios.

11.2.2 Y1 confidentiality, integrity, and replay protection

Requirement Name: Y1 protection in terms of confidentiality, integrity and replay

Requirement Reference: SEC-CTL-NEAR-RT-11, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-Y1-01, T-Y1-02, T-Y1-03

DUT/s: Near-RT RIC, Y1 consumers

Test Name: TC_Y1_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the Y1 interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection.

Procedure and execution steps

Preconditions

1. Near-RT RIC and Y1 consumers support TLS and connected within simulated or real network environments.
2. The Y1 interface is configured for testing.
3. TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
    - Establish a secure communication session over the Y1 interface.
    - Capture the network traffic during the session.
    - Analyse the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
    - Capture protected packets after the TLS handshake.
    - Modify the captured packets.
    - Inject the modified packets to the DUT.
    - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
    - Capture protected packets after the TLS handshake.
    - Replay the captured packets to the DUT.
    - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the Y1 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the Y1 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.2.3 Y1 Authorization

Requirement Name: Y1 protection in terms of mutual Authorization

Requirement Reference: SEC-CTL-NEAR-RT-10, clause 5.1.2.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-Y1-01, T-Y1-02, T-Y1-03

DUT/s: Near-RT RIC and Y1 consumers

Test Name: TC_Y1_AUTHORIZATION

Purpose: The purpose of this test is to validate that the Y1 interface enforces an authorization mechanism to prevent unauthorized access.

Procedure and execution steps

Preconditions

1. Near-RT RIC and Y1 consumers support OAuth 2.0 and are connected in simulated/real network environment.
2. The test environment is set up with Y1 interface configured.
3. The tester has access to the original data transported over the Y1 interface.
4. OAuth 2.0 is properly implemented and configured.

Execution steps

1. Execute the test on the OAuth 2.0 protocol as defined in clause 6.6.
2. Valid access tokens (positive case):
    a. The tester sends a request to access protected resources using a valid access token.
    b. The tester captures and analyses the response from the Y1 interface.
3. Invalid access tokens (negative case):
    a. The tester sends a request to access protected resources using an invalid or incorrect access token.
    b. The tester captures and analyses the response from the Y1 interface.
4. No access tokens (negative case):
    a. The tester sends a request to access protected resources without providing any access token.
    b. The tester captures and analyses the response from the Y1 interface.

Expected Results

- For 1. Expected results in clause 6.6
- For 2. ‘Valid access tokens’: The Y1 interface accepts the valid access tokens and responds with a successful authorization message.
- For 3. ‘Invalid access tokens’: The access is rejected, and an access failure message is received.
- For 4. ‘No access tokens’: The access is rejected due to the absence of tokens, and an appropriate error or unauthorized access message is received.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs of the request sent to access protected resources using valid access tokens.
- Screenshots or logs highlighting the successful authorization message.
- Logs of the request sent to access protected resources using invalid or incorrect access tokens.
- Screenshots or logs showing the rejection of the access and the access failure message.

11.3 O1

11.3.1 O1 Authenticity

Requirement Name: O1 protection in terms of authenticity

Requirement Reference: SEC-CTL-O1-2, SEC-CTL-O1-5, clause 5.2.2.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-MPLANE-01, T-O-RAN-05

DUT/s: SMO, O-CU, O-DU, Near-RT RIC

Test Name: TC_O1_AUTHENTICATION

Purpose: The purpose of this test is to verify the authenticity of the O1 interface, ensuring that only legitimate and authenticated O-RAN NFs can participate in the communication over the O1 interface.

Procedure and execution steps

Preconditions

- SMO, O-CU, O-DU, Near-RT RIC support mTLS and be connected in simulated/real network environment.
- The test environment is set up with O1 interface configured.
- The tester has access to the original data transported over the O1 interface.
- mTLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Execute the test on the mTLS protocol as defined in clause 6.3.
2. Valid Authentication Certificates (positive case):
    a. The tester sends a request to establish a connection with the O1 interface using valid authentication certificates.
    b. The tester verifies the mutual certificate verification between the ORAN NFs
    c. The tester captures and analyses the response received from the O1 interface.
3. Invalid Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection with the O1 interface with invalid certificates.
    b. The tester captures and analyses the response received from the O1 interface.
4. No Authentication Certificates (Negative Case):
    a. The tester sends a request to establish a connection without any certificates.
    b. The tester captures and analyses the response from the O1 interface.

Expected results

1. For 1. Expected results in clause 6.3
2. For 2. ‘Valid Authentication Certificates’: The O1 interface accepts the valid certificates and responds with a successful authentication message. The mutual certificate verification process is successful.
3. For 3. ‘Invalid Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the use of invalid certificates.
4. For 4. ‘No Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the absence of certificates.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs and screenshots showing adherence to mTLS protocol specifications as defined in [2] clause 4.2.

- Logs of authentication requests sent to the O1 interface.
- Logs of the mutual certificate verification process.
- Screenshots or logs of error messages or unusual behaviours for both invalid and no certificate scenarios.

11.3.2 O1 confidentiality, integrity and replay protection

Requirement Name: O1 protection in terms of confidentiality, integrity and replay

Requirement Reference:SEC-CTL-O1-1, SEC-CTL-O1-4, clause 5.2.2.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-MPLANE-01, T-O-RAN-05

DUT/s: SMO, O-CU, O-DU, Near-RT RIC

Test Name: TC_O1_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the O1 interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection.

Procedure and execution steps

Preconditions

- SMO, O-CU, O-DU, Near-RT RIC support TLS and be connected in simulated/real network environment.
- The O1 interface is configured for testing.
- TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
    - Establish a secure communication session over the O1 interface.
    - Capture the network traffic during the session.
    - Analyse the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
    - Capture protected packets after the TLS handshake.
    - Modify the captured packets.
    - Inject the modified packets to the DUT.
    - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
    - Capture protected packets after the TLS handshake.
    - Replay the captured packets to the DUT.
    - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the O1 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the O1 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.3.3 O1 Interface Network Configuration Access Control Model (NACM) Validation

11.3.3.0 Introduction

Following zero trust principles, O-RAN O1 interface shall enforce confidentiality, integrity and authenticity through an encrypted transport, and shall support least privilege access control using the network configuration access control model. The network configuration access control model (NACM) [14] provides the means to restrict access for users to a preconfigured subset of all available NETCONF protocol operations and content.

The security test case in this clause validates the NACM enforcement on the O-RAN component O1 interface for the role-based access control.

11.3.3.1 O1 Interface NACM Validation

Requirement Name: O1 Interface security requirements

Requirement Reference: REQ-NAC-FUN-1 to REQ-NAC-FUN-10, clause 5.2.2.1, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O-RAN-02, T-O-RAN-06

Requirement Description: Requirements of O1 Interface Confidentiality, Integrity & Authenticity protection and Least Privilege Access Control

DUT/s: Non-RT RIC, Near-RT RIC, O-CU-CP, O-CU-UP, O-DU

Test Name: TC_O1_NACM_VALIDATION

Purpose: O-RAN component(s) managed by SMO through O1 interface shall support secured NETCONF sessions over TLS and role-based least privilege access control enforced by NACM [14]. This test validates the O1 interface security requirements of the O-RAN component(s) with the focus on role-based NACM rule(s) set enforcement.

Procedure and execution steps

Preconditions

DUT shall be the O-RAN component with:

- IP enabled O1 interface, reachable from the authentication server;
- Valid certificate loaded for the server and necessary certificate authorities (CAs)

- Client's root CA required to validate NETCONF client certificate
- Valid TLS Client-to-NETCONF username mapping
- Configure the O-RAN element with the SMO details (SMO network address and port)

Execution steps

First set up a host/device with TLS client software installed, valid client certificates, keys, root CA certificate for the server (O-RAN component), and all intermediate CA certificates required to validate the client certificate.

The following test steps shall be validated:

1. Initiate NETCONF call home procedure from the O-RAN element towards SMO over O1 interface.

NOTE: The O-RAN element may initiate the NETCONF call home procedure as part of its initialization automatically.

2. SMO connects with O-RAN element over O1 interface using TLSv1.2 or TLSv1.3 - if available with a user account from the O1_nacm_management group
3. Verify the session is established and mapped to the correct NETCONF user
4. Verify the global NACM enforcement control setting of
    a. enable-nacm = true
    b. read-default = permit
    c. write-default = deny
    d. exec-default = deny
    e. enable-external-groups = true
5. Verify the NACM rule sets for the following pre-defined groups
    a. O1_nacm_management
    b. O1_user_management
    c. O1_network_management
    d. O1_network_monitoring
    e. O1_software_management for only PNFs
6. Close the NETCONF session and TLS connection

Upon availability of the NETCONF operations set(s) definition per NACM group, the NACM rule set(s) enforcement by the DUT shall be validated for each of those pre-defined groups listed above.

Expected results

The O-RAN component supports the NETCONF over TLS session over its O1 interface and NACM enforcement control settings.

Expected format of evidence:

Logs or screenshots showing:

- O1 interface setup.
- Valid server certificate and CA details.
- Client's root CA and intermediate CA certificates.
- TLS Client-to-NETCONF username mapping.

- O-RAN element configured with SMO details.
- Initiation of NETCONF call home procedure.
- TLSv1.2 or TLSv1.3 connection establishment.
- Correct NETCONF user session mapping.

11.4 O2

11.4.1 O2 Authenticity

Requirement Name: O2 protection in terms of authenticity

Requirement Reference: SEC-CTL-O-CLOUD-INTERFACE-3, clause 5.1.8.9.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O2-01

DUT/s: SMO, O-Cloud

Test Name: TC_O2_AUTHENTICATION

Purpose: The purpose of this test is to verify the authenticity of the O2 interface, ensuring that only legitimate and authenticated O-Cloud and SMO can participate in the communication over the O2 interface.

Procedure and execution steps

Preconditions

1. O-Cloud and SMO support mTLS and be connected in simulated/real network environment.
2. The test environment is set up with O2 interface configured.
3. The tester has access to the original data transported over the O2 interface.
4. mTLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Executes the tests on the mTLS protocol as defined in clause 6.3
2. Valid Authentication Certificates (positive case):
    a. The tester sends a request to establish a connection with the O2 interface using valid authentication certificates.
    b. The tester verifies the mutual certificate verification between the ORAN NFs.
    c. The tester captures and analyses the response from the O2 interface.
3. Invalid Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection with the O2 interface with invalid certificates.
    b. The tester captures and analyses the response from the O2 interface.
4. No Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection without any certificates.
    b. The tester captures and analyses the response from the O2 interface.

Expected results

1. For 1. Expected results in clause 6.3
2. For 2. ‘Valid Authentication Certificates’: The O2 interface accepts the valid certificates and respond with a successful authentication message.
3. For 3. ‘Invalid Authentication Certificates’: The connection is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the use of invalid certificates.
4. For 4. ‘No Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the absence of certificates.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

1. Logs and screenshots showing adherence to mTLS protocol specifications as defined in [2] clause 4.2.
2. Logs of authentication requests and responses on the O2 interface.
3. Logs of the mutual certificate verification process.
4. Screenshots or logs of error messages or unusual behaviours for both invalid and no certificate scenarios.

11.4.2 O2 confidentiality, integrity and replay protection

Requirement Name: O2 protection in terms of confidentiality, integrity and replay

Requirement Reference: SEC-CTL-O-CLOUD-INTERFACE-1, clause 5.1.8.9.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O2-01

DUT/s: SMO, O-Cloud

Test Name: TC_O2_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the O2 interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection.

Procedure and execution steps

Preconditions

1. O-Cloud and SMO support TLS and be connected in simulated/real network environment.
2. The O2 interface is configured for testing.
3. TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
    - Establish a secure communication session over the O2 interface.
    - Capture the network traffic during the session.
    - Analyse the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
    - Capture protected packets after the TLS handshake.
    - Modify the captured packets.

- Inject the modified packets to the DUT.
- Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.

3. Replay protection verification:
   - Capture protected packets after the TLS handshake.
   - Replay the captured packets to the DUT.
   - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the O2 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the O2 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.4.3 O2 Authorization

Requirement Name: O2 protection in terms of authorization

Requirement Reference: SEC-CTL-O-CLOUD-INTERFACE-2, clause 5.1.8.9.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O2-01

DUT/s: SMO, O-Cloud

Test Name: TC_O2_AUTHORIZATION

Purpose: The purpose of this test is to validate that the O2 interface enforces an authorization mechanism to prevent unauthorized access.

Procedure and execution steps

Preconditions

- O-Cloud and SMO support OAuth 2.0 and are connected in simulated/real network environment.
- The test environment is set up with O2 interface configured.
- The tester has access to the original data transported over the O2 interface.
- OAuth 2.0 is properly implemented and configured.

Execution steps

1) Execute the tests on the OAuth 2.0 protocol as defined in clause 6.6

2) Valid access tokens (positive case):

   a) The tester sends a request to access protected resources using a valid access token.

b) The tester captures and analyses the response from the O2 interface.

3) Invalid access tokens (negative case):

a) The tester sends a request to access protected resources using an invalid or incorrect access token.

b) The tester captures and analyses the response from the O2 interface.

4) No access tokens (negative case):

a) The tester sends a request to access protected resources without providing any access token.

b) The tester captures and analyses the response from the O2 interface.

Expected results

- For 1. Expected results in clause 6.6
- For 2. 'Valid access tokens': The O2 interface accepts the valid access tokens and responds with a successful authorization message.
- For 3. 'Invalid access tokens': The access is rejected, and an access failure message is received.
- For 4. 'No access tokens': The access is rejected due to the absence of tokens, and an appropriate error or unauthorized access message is received.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

1. Logs of the request sent to access protected resources using valid access tokens.
2. Screenshots or logs highlighting the successful authorization message.
3. Logs of the request sent to access protected resources using invalid or incorrect access tokens.
4. Screenshots or logs showing the rejection of the access and the access failure message.

11.5 E2

11.5.1 E2 confidentiality, integrity and replay protection

Requirement Name: E2 protection in terms of confidentiality, integrity, and replay

Requirement Reference: SEC-CTL-E2, SEC-CTL-NEAR-RT-2, SEC-CTL-NEAR-RT-7, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-05, T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: O-CU, O-DU, Near-RT RIC

Test Name: TC_E2_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the E2 interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection

Procedure and execution steps

Preconditions

- Near-RT RIC and E2 nodes support IPsec and are connected in simulated/real network environment.

- The E2 interface is configured for testing
- The tunnel mode IPsec ESP and IKE certificate authentication is implemented.
- The tester shall base the test on the profile defined in [2] clause 4.5.

Execution steps

1. Confidentiality verification:
    - Establish a secure communication session using IPsec over the E2 interface.
    - Capture the network traffic during the session.
    - Analyse the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
    - Capture traffic during a secure session.
    - Modify the captured packets.
    - Inject the modified packets to the DUT.
    - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
    - Capture protected packets during a secure session and attempt to replay them.
    - Replay the captured packets to the DUT.
    - Confirm that the DUT discards the replayed packets.

Expected Results

- Confidentiality: All sensitive data transmitted over the E2 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected format of evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.5.
- Evidence of secure communication sessions established over the E2 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.5.2 E2 Authenticity

11.5.2.1 E2 Authenticity with certificate

Requirement Name: Data Authentication over E2 interface

Requirement Reference: SEC-CTL-NEAR-RT-2, clause 5.1.3.2, SEC-CTL-E2-1, clause 5.2.4.2 in O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-05, T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: O-CU, O-DU, Near-RT RIC

Test Name: TC_E2_AUTHENTICATION_CERT

Purpose: The purpose of this test is to verify the authenticity of the E2 interface with valid certificates, ensuring that only legitimate and authenticated Near-RT RIC and E2 nodes can participate in the communication over the E2 interface.

Procedure and execution steps

Preconditions

- Near-RT RIC and E2 nodes support IPsec and are connected in simulated/real network environment.
- Near-RT RIC and E2 nodes support IPsec and are configured to use certificate-based authentication.
- The test environment is set up with E2 interface configured. Communication sessions over the E2 interface are established.
- The vendor provides documentation describing how authenticity protection is achieved for the data transmission over the E2 interface.
- The tunnel mode IPsec ESP and IKE certificate authentication is implemented.
- Tester has knowledge of the security parameters of tunnel for decrypting the ESP packets.
- Tester has access to the original user data transported over the E2 interface.
- IPsec is properly implemented and configured. The tester bases the test on the profile defined in [2] clause 4.5.

Execution steps

1. Execute the tests on the IPsec protocol as defined in clause 6.5.
2. Valid Authentication Credentials:
    a. The tester sends a request to establish a connection with the E2 interface using valid certificates.
    b. The tester captures and analyses the response from the E2 interface.
3. Invalid Authentication Credentials:
    a. The tester sends a request to establish a connection with the E2 interface using invalid certificates.
    b. The tester captures and analyses the response from the E2 interface.
4. No Authentication Credentials:
    a. The tester sends a request to establish a connection with the E2 interface without providing any certificates.
    b. The tester captures and analyses the response from the E2 interface.

Expected Results

- For 1. Expected results in clause 6.5.4
- For 2. ‘Valid Authentication Credentials’: The E2 interface accepts the valid certificate and responds with a successful authentication message.
- For 3. ‘Invalid Authentication Credentials’: The connection is rejected due to the certificate verification failure, and an authentication failure message is received.
- For 4. ‘No Authentication Credentials’: The connection attempt fails due to the absence of certificates, and an authentication failure message is received.

Expected format of evidence:

- Logs and screenshots showing adherence to IPsec protocol specifications as defined in [2] clause 4.5.
- Screenshots or logs of request-response messages confirming authentication with valid credentials.
- Screenshots or logs capturing the rejection of requests with invalid credentials.
- Screenshots or logs documenting attempts to connect without credentials and their rejection.

11.5.2.2 E2 Authenticity with PSK

Requirement Name: Data Authentication over E2 interface

Requirement Reference: SEC-CTL-NEAR-RT-2, clause 5.1.3.2, SEC-CTL-E2-1, clause 5.2.4.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-05, T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: O-CU, O-DU, Near-RT RIC

Test Name: TC_E2_AUTHENTICATION_PSK

Purpose: The purpose of this test is to verify the authenticity of the E2 interface with valid PSK, ensuring that only legitimate and authenticated Near-RT RIC and E2 nodes can participate in the communication over the E2 interface.

Procedure and execution steps

Preconditions

- Near-RT RIC and E2 nodes support IPsec and are connected in simulated/real network environment.
- Near-RT RIC and E2 nodes support IPsec and are configured to use PSK-based authentication.
- The test environment is set up with E2 interface configured. Communication sessions over the E2 interface are established.
- The vendor provides documentation describing how authenticity protection is achieved for the data transmission over the E2 interface.
- The tunnel mode IPsec ESP and IKE certificate authentication is implemented.
- Tester has knowledge of the security parameters of tunnel for decrypting the ESP packets.
- Tester has access to the original user data transported over the E2 interface.
- IPsec is properly implemented and configured. The bases the test on the profile defined in [2] clause 4.5.

Execution steps

1. Execute the tests on the IPsec protocol as defined in clause 6.5.
2. Valid Authentication Credentials:
    a. The tester sends a request to establish a connection with the E2 interface using valid PSKs.
    b. The tester captures and analyses the response from the E2 interface.
3. Invalid Authentication Credentials (Incorrect PSKs):
    a. The tester sends a request to establish a connection with the E2 interface with incorrect PSKs.
    b. The tester captures and analyses the response from the E2 interface.
4. No Authentication Credentials (No PSKs):
    a. The tester sends a request to establish a connection with the E2 interface without providing any PSKs.
    b. The tester captures and analyses the response from the E2 interface.

Expected Results

- For 1. Expected results in clause 6.5.4
- For 2. ‘Valid Authentication Credentials’: The E2 interface accepts the valid PSK and responds with a successful authentication message.
- For 3. ‘Invalid Authentication Credentials (Incorrect PSKs)’: The connection is rejected due to PSK verification failure, and an authentication failure message is received.
- For 4. ‘No Authentication Credentials (No PSKs)’: The connection attempt fails due to the absence of PSKs, and an authentication failure message is received.

Expected format of evidence:

- Logs and screenshots showing adherence to IPsec protocol specifications as defined in [2] clause 4.5.
- Logs or screenshots documenting request and response messages for successful authentication using valid credentials.
- Logs or screenshots capturing the request and response messages when invalid credentials are rejected.
- Logs or screenshots documenting the request and response messages for rejections of connections without PSKs.

11.5.2.3 E2 Interface data validation by Near-RT RIC

Requirement Name: Validation of the data received via E2 interface by Near-RT RIC

Requirement Reference: SEC-CTL-NEAR-RT-17,clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: The Near-RT RIC shall verify data received through the E2 interface as follows:

The data values are valid.

The data is being received at or below a pre-defined rate.

The Near-RT RIC shall log security event(s) if any of the verification steps fail.

Threat References: T-NEAR-RT-01, T-xApp-01

DUT/s: Near-RT RIC

Test Name: TC_E2_Interface_data_validation_by_NearRTRIC

Purpose: To validate the E2 traffic that is received by Near-RT RIC via E2 interface. The Near-RT RIC uses E2 interface to collect near real-time information (EXAMPLE:- UE basis, Cell basis) and provide value added services. These real-time information needs to be validated when it gets received at Near-RT RIC and security events are to be logged if data validation fails. E2 interface connects the Near-Real-Time RIC with other E2 nodes like O-CU, O-DU, and O-eNB.

Procedure and execution steps

EXAMPLE: One of the incoming data values to the Near-RT RIC are the measurement reports (carried in E2 Indication messages) that include:

- Channel quality reports: Signal strength, signal-to-noise ratio (SNR), and modulation quality.
- Interference reports: Identifying sources of interference.
- Load reports: Current load on E2 nodes

Preconditions

Client is the test system which simulates E2 data traffic towards Near-RT RIC. This includes all the supported E2 support services (EXAMPLE:- E2 RESET procedure), Measurement reports and other supported services. Test system is also capable of simulating multiple E2 connections where E2 traffic can be pushed.

Precondition:

1. Near-RT RIC is fully operational and data value validation in Near-RT RIC is defined, and the pre-defined data threshold rate is set. By fully operational Near-RT RIC, this means the Near-RT RIC is enabled with necessary xApps and configurations at the platform level.
2. Client system is logged in and the initial control connections are up with Near-RT RIC. At this point the Near-RT RIC has subscribed with E2 Nodes in the Client system and is expecting data at a predefined rate.
3. Login to the DUT with authorized credentials and start data collection required for checking the data handling.

Execution steps

1. From the client system, Initiate the E2 traffic with valid data values towards Near-RT RIC over single E2 connection
2. From the client test system, Initiate the E2 traffic with valid data on multiple E2 connections simultaneously
3. From the client test system, initiate invalid E2 traffic data

   EXAMPLE: Invalid values (or) Invalid format in the measurement reports (or) Invalid E2 Node configuration information sent in E2 setup request (or) Invalid cause in the E2 reset request
4. From the client test system, initiate the E2 data which is equal to the Near-RT RIC predefined data rate via multiple E2 connections simultaneously
5. From the client test system, initiate sudden burst of E2 data which is more than the Near-RT RIC predefined data rate via multiple E2 connections simultaneously

Expected results

After step 1, the DUT processes the data traffic received over a single E2 connection.

After step 2, the DUT processes the data traffic received simultaneously over multiple E2 connections.

After step 3, the DUT discards the data and security event is logged with the logs fields as per clause 5.3.8.8 of [5].

After step 4, the DUT receives E2 traffic and handles it because E2 data is at, or below the pre-defined rate in DUT.

After step 5, the DUT discards the spilled over data and security events are logged with the log fields are as per clause 5.3.8.8 of [5] because the E2 data rate is higher than the pre-defined rate in DUT.

Expected format of evidence: Log files, traffic captures and/or report files.

11.6 A1

11.6.1 A1 Authenticity

Requirement Name: A1 protection in terms of authenticity

Requirement Reference: SEC-CTL-A1-2, clause 5.2.1.2,n O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-A1-01

DUT/s: Non-RT RIC, Near-RT RIC

Test Name: TC_A1_Authentication

Purpose: The purpose of this test is to verify the authenticity of the A1 interface, ensuring that only legitimate and authenticated Non-RT RIC, Near-RT RIC can participate in the communication over the A1 interface.

Procedure and execution steps

Preconditions

- Non-RT RIC & Near-RT RIC support mTLS and be connected in a simulated/real network environment.
- The test environment is set up with the A1 interface configured.
- The tester has access to the original data transported over the A1 interface.
- mTLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Execute the test on the mTLS protocol as defined in clause 6.3.
2. Valid Authentication Certificates (positive case):
    a. The tester sends a request to establish a connection with the A1 interface using valid authentication certificates.
    b. The tester verifies the mutual certificate verification between Non-RT RIC and Near-RT RIC.
    c. The tester captures and analyses the response received from the A1 interface.
3. Invalid Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection with the A1 interface with invalid certificates.
    b. The tester captures and analyses the response received from the A1 interface.
4. No Authentication Certificates (negative Case):
    a. The tester sends a request to establish a connection without any certificates.
    b. The tester captures and analyses the response from the A1 interface.

Expected results

- For 1. Expected results in clause 6.3
- For 2. ‘Valid Authentication Certificates’: The A1 interface accepts the valid certificates and responds with a successful authentication message. The mutual certificate verification process is successful.
- For 3. ‘Invalid Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the use of invalid certificates.
- For 4. ‘No Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the absence of certificates.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs and screenshots showing adherence to mTLS protocol specifications as defined in [2] clause 4.2.
- Logs of authentication requests and responses on the A1 interface.
- Logs of the mutual certificate verification process.
- Screenshots or logs of error messages or unusual behaviours for both invalid and no certificate scenarios.

11.6.2 A1 confidentiality, integrity and replay protection

Requirement Name: A1 protection in terms of confidentiality, integrity, and replay

Requirement Reference: SEC-CTL-A1, clause 5.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-A1-02, T-A1-03

DUT/s: Non-RT RIC, Near-RT RIC

Test Name: TC_A1_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the A1 interface, ensuring that sensitive data remains protected through confidentiality, integrity, and replay protection.

Procedure and execution steps

Preconditions

- Non-RT RIC & Near-RT RIC support TLS and connected within simulated or real network environments.
- The A1 interface is configured for testing.
- TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
   - Establish a secure communication session over the A1 interface.
   - Capture the network traffic during the session.
   - Analyze the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
   - Capture protected packets after the TLS handshake.
   - Modify the captured packets.
   - Inject the modified packets to the DUT.
   - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
   - Capture protected packets after the TLS handshake.
   - Replay the captured packets to the DUT.
   - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the A1 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the A1 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.6.3 A1 Authorization

Requirement Name: A1 protection in terms of authorization

Requirement Reference: SEC-CTL-A1-3, clause 5.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-A1-01

DUT/s: Non-RT RIC, Near-RT RIC

Test Name: TC_A1_Authorization

Purpose: The purpose of this test is to validate that the A1 interface enforces an authorization mechanism to prevent unauthorized access.

Procedure and execution steps

Preconditions

- Non-RT RIC and Near-RT RIC support OAuth 2.0 and are connected in simulated/real network environment.
- The test environment is set up with A1 interface configured.
- The tester has access to the original data transported over the A1 interface.
- OAuth 2.0 is properly implemented and configured.

Execution steps

1. Execute the test on the OAuth 2.0 protocol as defined in clause 6.6.
2. Valid access tokens (positive case):
    a. The tester sends a request to access protected resources using a valid access token.
    b. The tester captures and analyses the response from the A1 interface.
3. Invalid access tokens (negative case):
    a. The tester sends a request to access protected resources using an invalid or incorrect access token.
    b. The tester captures and analyses the response from the A1 interface.
4. No access tokens (negative case):
    a. The tester sends a request to access protected resources without providing any access token.
    b. The tester captures and analyses the response from the A1 interface.

Expected Results

- For 1. Expected results in clause 6.6
- For 2. ‘Valid access tokens’: The A1 interface accepts the valid access tokens and responds with a successful authorization message.
- For 3. ‘Invalid access tokens’: The access is rejected, and an access failure message is received.
- For 4. ‘No access tokens’: The access is rejected due to the absence of tokens, and an appropriate error or unauthorized access message is received.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs of the request sent to access protected resources using valid access tokens.
- Screenshots or logs highlighting the successful authorization message.
- Logs of the request sent to access protected resources using invalid or incorrect access tokens.
- Screenshots or logs showing the rejection of the access and the access failure message.

11.7 R1

11.7.1 R1 Authenticity

Requirement Name: R1 protection in terms of authenticity

Requirement Reference: SEC-CTL-R1-2, clause 5.2.6.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-R1-03

DUT/s: Non-RT RIC, rApps

Test Name: TC_R1_AUTHENTICATION

Purpose: The purpose of this test is to verify the authenticity of the R1 interface, ensuring that only legitimate and authenticated Non-RT RIC, rApps can participate in the communication over the R1 interface.

Procedure and execution steps

Preconditions

- Non-RT RIC & rApps support mTLS and be connected in a simulated/real network environment.
- The test environment is set up with the R1 interface configured.
- The tester has access to the original data transported over the R1 interface.
- mTLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Execute the test on the mTLS protocol as defined in clause 6.3.
2. Valid Authentication Certificates (positive case):
    a. The tester sends a request to establish a connection with the R1 interface using valid authentication certificates.
    b. The tester verifies the mutual certificate verification between Non-RT RIC and rApps.
    c. The tester captures and analyses the response received from the R1 interface.
3. Invalid Authentication Certificates (negative case):
    a. The tester sends a request to establish a connection with the R1 interface with invalid certificates.
    b. The tester captures and analyses the response received from the R1 interface.
4. No Authentication Certificates (negative Case):
    a. The tester sends a request to establish a connection without any certificates.
    b. The tester captures and analyses the response from the R1 interface.

Expected results

For 1. Expected results in clause 6.3

For 2. ‘Valid Authentication Certificates’: The R1 interface accepts the valid certificates and responds with a successful authentication message. The mutual certificate verification process is successful.

For 3. ‘Invalid Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the use of invalid certificates.

For 4. ‘No Authentication Certificates’: The connection attempt is rejected, and an authentication failure message is received. The mutual certificate verification process fails due to the absence of certificates.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs and screenshots showing adherence to mTLS protocol specifications as defined in [2] clause 4.2.
- Logs of authentication requests and responses on the R1 interface.
- Logs of the mutual certificate verification process.
- Screenshots or logs of error messages or unusual behaviours for both invalid and no certificate scenarios.

11.7.2 R1 confidentiality, integrity and replay protection

Requirement Name: R1 protection in terms of confidentiality, integrity and replay

Requirement Reference: SEC-CTL-R1-1, clause 5.2.6.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-R1-06, T-R1-07

DUT/s: Non-RT RIC, rApps

Test Name: TC_R1_CONFIDENTIALITY_INTEGRITY_REPLAY

Purpose: To verify the enforcement of security policies over the R1 interface, ensuring that sensitive data remains protected through confidentiality, integrity and replay protection.

Procedure and execution steps

Preconditions

- Non-RT RIC & rApps supporting TLS, connected within simulated or real network environments.
- The R1 interface is configured for testing.
- TLS is properly implemented and configured as defined in [2] clause 4.2.

Execution steps

1. Confidentiality verification:
    - Establish a secure communication session over the R1 interface.
    - Capture the network traffic during the session.
    - Analyze the captured traffic to verify that all data is encrypted, ensuring confidentiality.
2. Integrity protection verification:
    - Capture protected packets after the TLS handshake.
    - Modify the captured packets.
    - Inject the modified packets to the DUT.
    - Confirm that the DUT discards the injected packets, e.g., does not deliver it to the higher layer.
3. Replay protection verification:
    - Capture protected packets after the TLS handshake.
    - Replay the captured packets to the DUT.
    - Confirm that the DUT discards the replayed packets.

Expected results

- Confidentiality: All sensitive data transmitted over the R1 interface is encrypted, with no data exposed in clear text.
- Integrity protection: The DUT detects and discards altered packets, ensuring data has not been tampered with.
- Replay protection: The DUT detects and discards replayed packets, preventing replay attacks.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs or screenshots showing TLS protocol adherence, as defined in the O-RAN Security Protocols Specifications [2] clause 4.2.
- Evidence of secure communication sessions established over the R1 interface, including details of encryption verification.
- Logs or screenshots showing the DUT's response to replayed and integrity-compromised packets, demonstrating the effectiveness of the security mechanisms in place.

11.7.3 R1 Authorization

Requirement Name: R1 protection in terms of authorization

Requirement Reference: SEC-CTL-R1-3, clause 5.2.6.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-R1-01, T-R1-04, T-R1-05

DUT/s: Non-RT RIC, rApps

Test Name: TC_R1_AUTHORIZATION

Purpose: The purpose of this test is to validate that the R1 interface enforces an authorization mechanism to prevent unauthorized access.

Procedure and execution steps

Preconditions

- Non-RT RIC and rApps support OAuth 2.0 and are connected in simulated/real network environment.
- The test environment is set up with R1 interface configured.
- The tester has access to the original data transported over the R1 interface.
- OAuth 2.0 is properly implemented and configured.

Execution steps

1. Execute the test on the OAuth 2.0 protocol as defined in clause 6.6.
2. Valid access tokens (positive case):
    a. The tester sends a request to access protected resources using a valid access token.
    b. The tester captures and analyses the response from the R1 interface.
3. Invalid access tokens (negative case):
    a. The tester sends a request to access protected resources using an invalid or incorrect access token.
    b. The tester captures and analyses the response from the R1 interface.
4. No access tokens (negative case):
    a. The tester sends a request to access protected resources without providing any access token.
    b. The tester captures and analyses the response from the R1 interface.

Expected Results

1. For 1. Expected results in clause 6.6

2. For 2. ‘Valid access tokens’: The R1 interface accepts the valid access tokens and responds with a successful authorization message.
3. For 3. ‘Invalid access tokens’: The access is rejected, and an access failure message is received.
4. For 4. ‘No access tokens’: The access is rejected due to the absence of tokens, and an appropriate error or unauthorized access message is received.

Expected Format of Evidence:

The following evidence, in one or more formats as applicable, should be provided:

- Logs of the request sent to access protected resources using valid access tokens.
- Screenshots or logs highlighting the successful authorization message.
- Logs of the request sent to access protected resources using invalid or incorrect access tokens.
- Screenshots or logs showing the rejection of the access and the access failure message.

12 Security test of O-RU

12.1 Overview

This clause contains security tests to validate the security protection mechanism specific to O-RU.

12.2 SSH on M-Plane interface

Requirement Name: Network Security Protocol - SSH

Requirement Reference: Clause 5.4, O-RAN WG4 Management Plane Specification [21]

Requirement Description: Robust protocol implementation with adequately strong cipher suites is being required for SSH

Threat References: T-O-RAN-05

DUT/s: O-RU

Test name: TC_SSH_MPlane

Purpose: To verify implementation of the SSH protocol in O-RU along with validation of supported SSH version and robustness of cryptographic algorithms used for host key, symmetric encryption, key exchange, and MACs as specified in [21]

Procedure and execution steps

Preconditions

DUT is the O-RU with SSH service enabled as server. Client is a test equipment with SSH audit tool which is used for server-side testing.

Execution steps

This test case follows the "server-side testing" procedure for SSH specified in TC_SSH_Server_and_Client_Protocol, clause 6.2 of the present document.

Expected results

O-RU as SSH server supports only SSHv2 version with no older version supported and algorithms (for host key, symmetric encryption, key exchange, and MACs) defined in clause 5.4 of [21].

Expected format of evidence: As defined in clause 6.2 of the present document.

12.3 TLS on M-Plane interface

Requirement Name: Network Security Protocol - TLS

Requirement Reference: Clause 5.4, O-RAN WG4 Management Plane Specification [21]

Requirement Description: Support TLS v1.2 and/or TLS v1.3 with protocol profiles

Threat References: T-O-RAN-05

DUT/s: O-RU

Test name: TC_TLS_MPlane

Purpose: To verify implementation of the TLS protocol in O-RU along with validation of mandated/optional TLS versions and cipher suites specified in clause 5.4 of [21]. Since NETCONF implementations support X.509v3 certificate-based authentication using TLS 1.2, mutual authentication shall also be tested using both valid and invalid client certificates.

Procedure and execution steps

Preconditions

DUT is the O-RU with TLS service enabled as server equipped with CA cert for signing client certificate(s). Client is a testing equipment with TLS scanning tool with client certificate(s).

Execution steps

This test case follows the execution steps for TLS specified in TC_TLS_Protocol, clause 6.3 of the present document.

Expected results

O-RU as TLS server shall support TLS starting from version 1.2 with no older version enabled along with protocol profiles/Cipher suites defined in clause 5.4 of [21].

Expected format of evidence: As defined in clause 6.3 of the present document.

12.4 Security functional requirements and test cases

The 802.1X Supplicant Validation test cases in clause 11.2.2 of the present document apply to O-RU.

13 Security test of Near-RT RIC

13.1 Overview

This clause contains security tests to validate the security protection mechanism specific to Near-RT RIC.

13.2 Void

13.3 Transactional APIs

13.3.1 Introduction

Transactional APIs in the Near-RT RIC are APIs that are based on HTTP/TLS, i.e. APIs based on REST or gRPC.

13.3.2 TLS for transactional APIs

Requirement Name: TLS for transactional APIs

Requirement Reference: SEC-CTL-NEAR-RT-6, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5].

Requirement Description: “Transactional APIs (REST and gRPC) shall support TLS to provide message confidentiality and integrity.”

Threat References: T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: xApp, Near-RT RIC

Test name: TC_TLS_APIs

Purpose: To verify the transactional APIs (REST and gRPC) support TLS to provide message confidentiality and integrity.

Procedure and execution steps

Preconditions

DUT is configured and with TLS support enabled.

The other end may be simulated or a testing equipment.

Execution steps

This test case follows the execution steps for TLS specified in TLS Execution steps, clause 6.3 of the present document.

Expected results

The transaction APIs provides confidentiality and integrity protection for data in transit.

Expected format of evidence: Tool reports, log files, traffic captures and/or screenshots.

13.3.3 mTLS for transactional APIs

Requirement Name: mTLS for transactional APIs

Requirement Reference: REQ-SEC-NEAR-RT-3, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5].

Requirement Description: “The communication between xApps and Near-RT RIC platform APIs shall be mutually authenticated.”

Threat References: T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: xApp, Near-RT RIC

Test Name: TC_mTLS_APIs

Purpose: To verify the transactional APIs (REST and gRPC) support mutual TLS (mTLS) authentication via X.509v3 certificates.

Procedure and execution steps

Applicability: DUTs that support mTLS as a mutual authentication mechanism.

Preconditions

DUT is configured and with mTLS support enabled.The other end may be simulated or a testing equipment.

Execution steps

This test case follows the execution steps for TLS specified in TLS Execution steps, clause 6.3 of the present document.

Expected results

The transaction APIs supports mutual TLS (mTLS) authentication.

Expected format of evidence: Tool reports, log files, traffic captures and/or screenshots.

13.3.4 OAuth 2.0 for transactional APIs

Requirement Name: OAuth 2.0 for transactional APIs

Requirement Reference: REQ-SEC-NEAR-RT-4, REQ-SEC-NEAR-RT-5, clause 5.1.3.1, O-RAN Security Requirements and Controls Specifications [5].

Requirement Description: Near-RT RIC architecture provides an authorization framework.

Threat References: T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: xApp, Near-RT RIC

Test Name: TC_OAuth2.0_API

Purpose: To verify the transactional APIs (REST and gRPC) in the DUT support the OAuth 2.0 authorization framework.

Procedure and execution steps

Preconditions

DUT is configured and with OAuth 2.0 support enabled.

The other end may be simulated or a testing equipment.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6 of the present document.

Expected results

The transaction APIs supports the use of OAuth 2.0.

Expected format of evidence: Tool reports, log files, traffic captures and/or screenshots.

13.4 Security test of Near-RT RIC OAuth 2.0 Resource Owner/Server

13.4.1 Overview

This clause contains security tests to verify OAuth2.0 implementation on Near-RT RIC as resource owner/server for A1-P.

13.4.2 Near-RT RIC OAuth 2.0 Resource Owner/Server

Requirement Name: Near-RT RIC support as OAuth2.0 resource owner/server

Requirement Reference: Clause 5.1.3.2, Security Controls, Near-RT RIC and xApps, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for Near-RT RIC authorization of service requests

Threat References: T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: Near-RT RIC

Test Name: TC_NearRTRIC_OAuth2.0_Server

Purpose: To validate the Near-RT RIC support as OAuth 2.0 resource owner/server for A1-P, as specified in clause 4.7, O-RAN Security Protocols Specifications [2] for service requests received from a Near-RT RIC.

Procedure and execution steps

Preconditions

DUT is acting as a resource owner/server with OAuth 2.0 support enabled. OAuth2.0 Client is the test system equipped to send the service requests over a secured TLS communication with mutual TLS authentication.

Execution steps

This test case shall follow execution steps for OAuth2.0 specified in clause 6.6 of the present document.

Expected results

The Near-RT RIC shall be able to authorize/deny access to resources using OAuth 2.0.

Expected format of evidence: Log files, traffic captures and/or report files.

13.5 Security test of Near-RT RIC OAuth 2.0 client

13.5.1 Overview

This clause contains security tests to verify the implementation on Near-RT RIC as OAuth2.0 client for A1-EI.

13.5.2 Near-RT RIC OAuth 2.0 client

Requirement Name: Near-RT RIC support as OAuth2.0 client for A1-EI

Requirement Reference: Clause 5.1.3.2, Security Controls, Near-RT RIC and xApps, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for Near-RT RIC authorization of service requests

Threat References: T-NEAR-RT-01, T-NEAR-RT-02, T-NEAR-RT-03, T-NEAR-RT-04

DUT/s: Near-RT RIC

Test Name: TC_NearRTRIC_OAuth2.0_Client

Purpose: To validate the Near-RT RIC support as OAuth 2.0 client for A1-EI, as specified in clause 4.7, O-RAN-Security Protocols Specifications [2]

Procedure and execution steps

Preconditions

DUT is acting as a resource client with OAuth 2.0 support enabled.

Execution steps

This test case shall follow execution steps for OAuth2.0 specified in clause 6.6 of the present document.

Expected results

The Near-RT RIC shall be able to request and be permitted access to resources using OAuth2.0

Expected format of evidence: Log files, traffic captures and/or report files.

14 Security test of xApps

14.1 Overview

This clause contains security tests to validate the security protection mechanism specific to xApps deployed on Near-RT RIC.

14.2 xApp Signing and Verification

Security test cases "TC_SW_Img_Pkg_Signing" and "TC_SW_Img_Pkg_Verification" apply to this clause.

14.3 xAppID

This clause contains security tests to validate the xApp ID which is a string that uniquely identifies the xApp instance. The format of this string is a Universally Unique Identifier (UUID) version 4 (as described in IETF RFC 9562 [30]).

14.3.1 xApp ID format check

Requirement Name: xApp ID uniqueness check for the xApp instance

Requirement Reference: SEC-CTL-NEAR-RT-13, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: To validate the format of xApp ID string and the uniqueness of the same which will be a Universally Unique Identifier (UUID) version 4 (as described in IETF RFC 9562 [30]).

Threat References: T-NEAR-RT-05

DUT/s: Near-RT RIC

Test Name: TC_xApp_ID_validation

Purpose: To validate the xApp ID format that uniquely identifies the xApp instance. In this test, we are initiating registration requests from 3 xApp instances and validating the response from Near-RT RIC. The xApp ID format is checked against Universally Unique Identifier (UUID) version 4 (as described in IETF RFC 9562 [30]).

Procedure and execution steps

Preconditions

xApp instances are pre-provisioned with initial registration credential (OAuth 2.0 token), and the xApp instance CSR message.

NOTE: xApp instances can be instantiated of the same or different xApp images

Execution steps

1. Initiate the first xApp instance registration procedure with DUT and get for the registration response.
2. Initiate the second xApp instance registration procedure with DUT and get for the registration response.
3. Initiate the third xApp instance registration procedure with DUT and get for the registration response.

Expected results

After each execution step, the Registration response from DUT includes the xApp certificate for the xApp instance. The field "“Subject Alternative Name"” in the xApp instance certificate contains URI for the xApp ID as an URN. This URI contains the xApp ID of the different xApp instances.

The xApp ID generated in SAN field of xApp instance certificate after each execution step is unique and different from the others.

The assigned xApp ID format is complaint with Universally Unique Identifier (UUID) version 4 (as described in IETF RFC 9562 [30]).

NOTE: the certificate details could be seen with the openssl tool

Expected format of evidence: Log files, traffic captures, report files, certificates and/or screenshots.

14.3.2 xApp ID in xApp instance Certificate

Requirement Name: xApp ID presence in "Subject Alternative Name" field of the xApp instance certificate.

Requirement Reference: SEC-CTL-NEAR-RT-14, clause 5.1.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: "Subject Alternative Name" in the xApp instance certificate contains URI for the xApp ID as an URN. This URI contains the xApp ID of the xApp instance using the UUID format as described in IETF RFC 9562 [30].

Threat References: T-xApp-02, T-NEAR-TR-05

DUT/s: Near-RT RIC

Test Name: TC_xApp_ID_check_in_xApp_instance_certificate

Purpose: To check the xApp ID embedded in subject Alternate Name field of xApp instance certificate.

Procedure and execution steps

Preconditions

xApp image is available for instantiation.

xApp Registration procedure is successfully done and xApp instance certificate has been assigned to xApp as part of the Registration response.

Execution steps

1. Establish a TLS session to the xApp instance with authorized credentials.

EXAMPLE: TLS session may be established using one of the services that xApp instance provides

2. Capture the xApp instance certificate (X.509v3) on the xApp instance and open the certificate to check the details.

EXAMPLE: Openssl x509 -in <xApp_certificate.pem> -text -noout

3. Check the "Subject Alternative Name" field in the certificate details.

Expected results

The "Subject Alternative Name" contained in the certificate of the xApp instance, contains URI for the xApp ID as an URN. xApp ID of the xApp instance is conformant to UUID format as described in IETF RFC 9562 [30].

Expected format of evidence: Log files, traffic captures, screenshots, certificates and/or report files.

15 Security test of Non-RT RIC

15.1 Overview

This clause contains security tests to validate the security protection mechanism specific to Non-RT RIC and the R1 and A1 interfaces. Security test cases for rApps are covered in a separate sub-clause.

15.2 Non-RT RIC

Following zero trust principles, O-RAN Non-RT RIC shall enforce authorization using OAuth 2.0

15.2.1 Non-RT RIC OAuth 2.0 Resource Owner/Server

Requirement Name: Server authorization support

Requirement Reference: REQ-SEC-NonRTRIC-1, clause 5.1.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Non-RT RIC supports OAuth 2.0 as a Server

Threat References: T-NONRTRIC-01, T-NONRTRIC-02, T-NONRTRIC-03

DUT/s: Non-RT RIC

Test Name: TC_NonRTRIC_OAuth2.0_Server

Purpose: To verify the Non-RT RIC supports OAuth 2.0 resource owner/server for A1-EI.

Procedure and execution steps

Preconditions

The DUT is acting as a Resource Owner/Server and has OAuth 2.0 support enabled.

The rest of the elements of the setup may be real or simulated.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The Non-RT RIC is able to authorize/deny access to resources using OAuth 2.0.

Expected format of evidence: Log files, traffic captures and/or report files.

15.2.2 Non-RT RIC OAuth 2.0 Client

Requirement Name: Client authorization support

Requirement Reference: REQ-SEC-NonRTRIC-1, clause 5.1.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Non-RT RIC supports OAuth 2.0 as a Client

Threat References: T-NONRTRIC-01, T-NONRTRIC-02, T-NONRTRIC-03

DUT/s: Non-RT RIC

Test Name: TC_NonRTRIC_OAuth2.0_Client

Purpose: To verify the Non-RT RIC supports OAuth 2.0 client for A1-P.

Procedure and execution steps

Preconditions

The DUT is acting as a Client and has OAuth 2.0 support enabled.

The rest of the elements of the setup may be real or simulated.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The Non-RT RIC is able to request and be permitted access to resources using OAuth 2.0.

Expected format of evidence: Log files, traffic captures and/or report files.

15.2.3 Non-RT RIC Framework OAuth 2.0

Requirement Name: Framework Server authorization support

Requirement Reference: REQ-SEC-NonRTRIC-2, clause 5.1.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Non-RT RIC Framework supports OAuth 2.0 as a Server

Threat References: T-NONRTRIC-01, T-NONRTRIC-02, T-NONRTRIC-03

DUT/s: Non-RT RIC

Test Name: TC_NonRTRIC_OAuth2.0_Framework_Server

Purpose: To verify the Non-RT RIC Framework supports OAuth 2.0 as a resource owner/server.

Procedure and execution steps

Preconditions

The DUT is acting as a Resource Owner and has OAuth 2.0 support enabled.

The rest of the elements of the setup may be real or simulated.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The Non-RT RIC Framework is able to authorize access to resources using OAuth 2.0.

Expected format of evidence: Log files, traffic captures and/or report files.

15.3 R1 interface

Void

15.4 A1 interface

Void

16 Security test of rApps

16.1 Overview

This clause contains security tests to validate the security protection mechanism specific to rApps deployed on Non-RT RIC.

16.2 rApp Signing and Verification

Security test cases "TC_SW_Img_Pkg_Signing" and "TC_SW_Img_Pkg_Verification" apply to this clause.

16.3 rApp Authorization

16.3.1 rApp OAuth 2.0 Client

Requirement Name: rApp OAuth2.0 Client support

Requirement Reference: REQ-SEC-NonRTRIC-3, clause 5.1.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: “rApps shall provide client authorization requests to the Non-RT RIC Framework.”

Threat References: T-rAPP-04

DUT/s: rApps

Test Name: TC_OAuth2.0_rApp

Purpose: To verify the rApp supports OAuth 2.0 client.

Procedure and execution steps

Preconditions

The DUT is acting as an OAuth2.0 Client with OAuth 2.0 support enabled.

The rest of the elements of the setup may be real or simulated.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The rApp is able to request access and be permitted access to resources using OAuth 2.0.

Expected format of evidence: Log files, traffic captures and/or report files.

17 Security test of SMO

17.1 Overview

This clause contains security tests to validate security protection mechanisms related to the SMO. The test cases validate the security of SMO termination of O1 interfaces, SMO, SMO Services (SMOS) Communications, SMO External Interfaces, and SMO Logging are secured to zero trust principles for confidentiality, integrity, authentication, and authorization. Definitions for the O-RAN terms SMO Service (SMOS), SMO Function (SMOF), SMO External Interfaces, and SMO External System are provided in [1].

The test cases apply to the normative security requirements specified in [5] based upon the following approved security architecture:

The SMO enforces confidentiality, integrity and authenticity through an encrypted transport for the O1 interface and supports least privilege access control using the network configuration access control model (NACM) for authorization.

The SMO supports mutual authentication and authorization of SMO Functions (SMOF) and External Interfaces.

SMO Internal Communications provide communication and services between the SMO, SMOFs, Non-RT RIC Functions, and rApps. SMO Internal Communications shall provide confidentiality and integrity protection of data in transit and shall support mutual authentication and authorization for access to services and resources.

SMO External Interfaces provide import of AI enrichment data from external data sources to the SMO. SMO External Interfaces shall provide confidentiality and integrity protection of data in transit and shall support mutual authentication and authorization for access to services and resources.

17.2 Void

17.3 SMO

17.3.1 SMO OAuth 2.0 Resource Owner/Server

Requirement Name: SEC-CTL-SMO-3

Requirement Reference: Clause 5.1.1.2.1, Security Controls, SMO, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for SMO to authorize service requests from SMO Functions.

Threat References: T-SMO-02, T-SMO-05

DUT/s: SMO, SMO Functions

Test Name: TC_SMO_OAuth2.0_Resource_Owner_Server

Purpose: To verify the SMO shall support OAuth 2.0 resource owner/server.

Procedure and execution steps

Preconditions

DUT shall be the SMO with OAuth 2.0 support enabled.

Execution steps

This test case shall follow the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The SMO shall be able to authorize/deny access requests received from SMO Functions using OAuth 2.0.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.3.2 SMO OAuth 2.0 Client

Requirement Name: SEC-CTL-SMO-4

Requirement Reference: Clause 5.1.1.2.1, Security Controls, SMO, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for SMO to support client functionality for service requests to other SMO Functions

Threat References: T-SMO-02, T-SMO-05

DUT/s: SMO

Test Name: TC_SMO_OAuth2.0_Client

Purpose: To verify the SMO supports OAuth 2.0 client.

Procedure and execution steps

Preconditions

DUT shall be the SMO with OAuth 2.0 support enabled.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The SMO shall be able to request and be permitted/denied access to resources using OAuth 2.0.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.3.3 SMO mTLS for mutual authentication

Requirement Name: SEC-CTL-SMO-5

Requirement Reference: Clause 5.1.1.2.1, Security Controls, SMO, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: SMO support mTLS for mutual authentication with SMO Functions.

Threat References: T-SMO-01, T-SMO-04

DUT/s: SMO

Test Name: TC_SMO_mTLS

Purpose: To verify the SMO supports mutual authentication with SMO Functions using mTLS, with PKI and X.509 certificates.

Procedure and execution steps

Preconditions

DUT shall be the SMO with mTLS support enabled. An external OAuth 2.0 Authorization Server is available and configured.

Execution steps

This test case follows the execution steps for mTLS specified in mTLS Execution steps, clause 6.3.

Expected results

The SMO shall support mutual authentication of SMO Functions using mTLS.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.4 SMO Internal Communications

17.4.1 TLS for SMO Internal Communications

Requirement Name: SEC-CTL-SMO-Internal-1

Requirement Reference: Clause 5.1.1.2.2, Security Controls, SMO Internal Communications, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Data in transit protection with TLS for SMO Internal Communications

Threat References: T-SMO-09

DUT/s: SMO, Non-RT RIC

Test Name: TC_SMO_TLS_Internal_Communications

Purpose: To verify the SMO supports TLS on SMO Internal Communications.

Procedure and execution steps

Preconditions

DUT shall be the SMO with TLS support enabled.

Execution steps

This test case shall follow the execution steps for TLS specified in TLS Execution steps, clause 6.3.

Expected results

SMO Internal Communications shall provide confidentiality and integrity protection using TLS for data in transit.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.4.2 mTLS for SMO Internal Communications – SMO Functions

Requirement Name: SEC-CTL-SMO-Internal-2

Requirement Reference: Clause 5.1.1.2.2, Security Controls, SMO Internal Communications, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Mutual authentication with mTLS for SMO Internal Communications

Threat References: T-SMO-01, T-SMO-04

DUT/s: SMO

Test Name: TC_SMO_mTLS_Internal_Communications

Purpose: To verify SMO Functions support mutual authentication using mTLS, with PKI and X.509 certificates, for SMO Internal Communications.

Procedure and execution steps

Preconditions

DUT shall be the SMO Function with mTLS support enabled.

Execution steps

This test case follows the execution steps for mTLS specified in mTLS Execution steps, clause 6.3.

Expected results

The SMO Function shall support mutual authentication using mTLS.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.5 SMO External Interfaces

17.5.1 TLS for SMO External Interfaces

Requirement Name: SEC-CTL-SMO-External-1

Requirement Reference: Clause 5.1.1.2.3, Security Controls, SMO External Interfaces, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Data in transit protection with TLS on SMO External Interfaces

Threat References: T-SMO-09

DUT/s: SMO

Test Name: TC_SMO_TLS_External_Interfaces

Purpose: To verify the SMO supports TLS on SMO External Interface.

Procedure and execution steps

Preconditions

DUT shall be the SMO with TLS support enabled.

Execution steps

This test case shall follow the execution steps for TLS specified in TLS Execution steps, clause 6.3.

Expected results

SMO External Interface shall provide confidentiality and integrity protection using TLS for data in transit.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.5.2 mTLS for SMO External Interfaces

Requirement Name: SEC-CTL-SMO-External-2

Requirement Reference: Clause 5.1.1.2.3, Security Controls, SMO External Interfaces, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Mutual authentication with mTLS on SMO External Interfaces

Threat References: T-SMO-01, T-SMO-04

DUT/s: SMO

Test Name: TC_SMO_mTLS_External_Interfaces

Purpose: To verify the SMO supports mutual authentication using mTLS, with PKI and X.509 certificates for SMO External Interfaces.

Procedure and execution steps

Preconditions

DUT shall be the SMO with mTLS support enabled.

Execution steps

This test case follows the execution steps for mTLS specified in mTLS Execution steps, clause 6.3.

Expected results

The SMO shall support mutual authentication of SMO Functions using mTLS for SMO External Interfaces.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.5.3 SMO Framework OAuth 2.0 Resource Owner/Server for External Interface

Requirement Name: SEC-CTL-SMO-External-3

Requirement Reference: Clause 5.1.1.2.3, Security Controls, SMO External Interfaces, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for SMO to authorize service requests from external systems

Threat References: T-SMO-02, T-SMO-05

DUT/s: SMO

Test Name: TC_SMO_OAuth2.0_Resource_Owner_Server_External_Interface

Purpose: To verify the SMO supports OAuth 2.0 resource owner/server for SMO External Interfaces.

Procedure and execution steps

Preconditions

DUT shall be the SMO with OAuth 2.0 support enabled. An external OAuth 2.0 Authorization Server is available and configured.

Execution steps

This test case shall follow the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The SMO shall be able to authorize/deny access requests received from an external system using OAuth 2.0.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.5.4 SMO Functions OAuth 2.0 Client

Requirement Name: SEC-CTL-SMO-External-4

Requirement Reference: Clause 5.1.1.2.3, Security Controls, SMO External Interfaces, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: OAuth 2.0 security controls for SMO support client functionality for service requests to external systems

Threat References: T-SMO-02, T-SMO-05

DUT/S: SMO

Test Name: TC_SMO_OAuth2.0_Client_External_Interface

Purpose: To verify the SMO supports OAuth 2.0 client for External Interfaces.

Procedure and execution steps

Preconditions

DUT shall be the SMO with OAuth 2.0 support enabled. An external OAuth 2.0 Authorization Server is available and configured.

Execution steps

This test case follows the execution steps for OAuth2.0 specified in OAuth Execution steps, clause 6.6.

Expected results

The SMO shall be able to request and be permitted/denied access to external resources using OAuth 2.0.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.6 SMO Logging

17.6.1 TLS for SMO Logging Export

Requirement Name: SEC-CTL-SMO-Log-1

Requirement Reference: Clause 5.1.1.2.4, Security Controls, SMO Logging, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: TLS for SMO Logging Export

Threat References: T-SMO-16

DUT/s: SMO

Test Name: TC_SMO_TLS_Logging_Export

Purpose: To verify the SMO supports TLS for SMO logging export.

Procedure and execution steps

Preconditions

DUT shall be the SMO with TLS support enabled.

Execution steps

This test case shall follow the execution steps for TLS specified in TLS Execution steps, clause 6.3.

Expected results

SMO shall provide confidentiality and integrity protection for logging export.

Expected format of evidence: Log entries, packet captures, and screenshots.

17.6.2 mTLS for SMO Logging Export

Requirement Name: SEC-CTL-SMO-Log-3

Requirement Reference: Clause 5.1.1.2.4, Security Controls, SMO Logging, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: mTLS on SMO Logging Export

Threat References: T-SMO-01

DUT/s: SMO

Test Name: TC_SMO_mTLS_Logging_Export

Purpose: To verify the SMO supports mutual authentication using mTLS, with PKI and X.509 certificates, for SMO logging export.

Procedure and execution steps

Preconditions

DUT shall be the SMO with mTLS support enabled.

Execution steps

This test case follows the execution steps for mTLS specified in mTLS Execution steps, clause 6.3.

Expected results

The SMO shall support mutual authentication using mTLS for SMO logging export.

Expected format of evidence: Log entries, packet captures, and screenshots.

18 Security test of O-Cloud

18.1 Overview

This clause contains security tests to validate the security protection mechanism specific to O-Cloud hosting the O-RAN components/system.

18.2 Void

18.3 O-Cloud virtualization layer

18.3.1 Secure authentication (positive case)

Requirement Name: Secure authentication to O-Cloud APIs

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Secure_Authentication_Positive

Purpose: The purpose of this test is to ensure secure authentication to O-Cloud APIs.

Procedure and execution steps

Preconditions

- O-Cloud authentication mechanism are enabled.
- Valid credentials are available for authentication.

Execution steps

1. Attempt to access O-Cloud APIs with valid authentication credentials:
    a. Send an API request with providing valid authentication credentials.

    EXAMPLE: Send an API request by executing a Kubernetes **curl** command or using a Kubernetes client using the valid API key or access token for authentication (e.g., valide kubeconfig file or service account token).

    b. Capture the response received, including the status code and response body.
    c. Verify that the API response returns a success status code.

Expected results

The API response returns a success status code.

18.3.2 Secure authentication (negative case)

Requirement Name: Secure authentication to O-Cloud APIs

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Secure_Authentication_Negative

Purpose: The purpose of this test is to intentionally validate the behaviour of the authentication mechanism when encountering invalid or unauthorized authentication credentials.

Procedure and execution steps

Preconditions

O-Cloud authentication mechanism is enabled.

Execution steps

1. Attempt to access O-Cloud APIs with invalid or expired authentication credentials:
    a. Send an API request with providing invalid or expired authentication credentials.

    EXAMPLE: Send an API request by executing a Kubernetes **curl** command or using a Kubernetes client using the invalid or expired API key or access token for authentication (e.g., invalid kubeconfig file, expired service account token).

    b. Capture the response received, including the status code and response body.
    c. Verify that the API response returns an authentication failure status code.

Expected results

The API response returns an authentication failure status code.

18.3.3 Secure authorization (positive case)

Requirement Name: Secure authorization for accessing O-Cloud APIs

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Secure_Authorization_Positive

Purpose: The purpose of this test is to verify that the authorization mechanism for accessing O-Cloud APIs is functioning correctly, ensuring that entities have appropriate permissions to perform specific actions on O-Cloud resources.

Procedure and execution steps

NOTE: Entities include Applications, SMO and O-Cloud software components.

Preconditions

- Valid authentication credentials.
- O-Cloud access control system is enabled containing different levels of permissions assigned to entities.

  EXAMPLE: Access control system such as Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC).

Execution steps

1. Authenticate with valid credentials:
    a. Use valid authentication credentials to establish a connection with the O-Cloud API.
2. Send an API request with authorized permissions:
    a. Construct a valid API request to perform a specific action,

   EXAMPLE: specific action includes creating a pod, updating a deployment, or deleting a service.

    b. Ensure that the requested action aligns with the entity's assigned permissions.
    c. Send the request to the O-Cloud API endpoint.
3. Validate the response:
    a. Verify that the API response returns a success status code indicating the action was successfully executed.

Expected results

The API response returns a success status code, confirming that the requested action was authorized and executed successfully.

18.3.4 Secure authorization (negative case)

Requirement Name: Secure authorization for accessing O-Cloud APIs

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Secure_Authorization_Negative

Purpose: The purpose of this test is to intentionally validate the behaviour of the authorization mechanism when encountering unauthorized or invalid access attempts.

Procedure and execution steps

Preconditions

- Valid authentication credentials.
- O-Cloud access control system is enabled containing different levels of permissions assigned to entities.

EXAMPLE: Access control system such as Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC).

Execution steps

1. Authenticate with valid credentials:
    a. Use valid authentication credentials to establish a connection with the O-Cloud API.
2. Send an API request with unauthorized permissions:
    a. Construct a valid API request to perform a specific action that exceeds the entity's assigned permissions,

   EXAMPLE: specific action includes creating a pod, updating a deployment, or deleting a service.

    b. Send the request to the O-Cloud API endpoint.
3. Validate the response:
    a. Verify that the API response returns a failure status code indicating the action was unauthorized.

Expected results

The API response returns a failure status code, indicating that the requested action was unauthorized.

18.3.5 Validate network connections allowed by network policies

Requirement Name: Isolation & secure communication between Applications

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Connection_Allowed_Policies

Purpose: The purpose of this test is to ensure that network connections between VMs/Containers allowed by network policies are successfully established.

Procedure and execution steps

Preconditions

O-Cloud with network policies is configured to allow specific VMs/Containers to VMs/Containers communication.

Execution steps

1. Deploy two VMs/Containers A and B, in different zones or with different environment.

   EXAMPLE: Zones such as namespaces in Kubernetes, environment such as labels in Kubernetes

2. Define network policies that explicitly allow communication between the two VMs/Containers.
3. Attempt to establish a network connection from VM/Container A to VM/Container B using tools.

   EXAMPLE: tools such as curl or ping in Kubernetes

4. Capture the response or output received.

Expected results

The network connection from VM/Container A to VM/Container B is successfully established, indicating that the network policies allow the communication between the VMs/Containers.

18.3.6 Validate network connections not allowed by network policies

Requirement Name: Isolation & secure communication between Applications

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Connection_Not_Allowed_Policies

Purpose: The purpose of this test is to ensure that network connections between VMs/Containers not allowed by network policies are blocked.

Procedure and execution steps

Preconditions

O-Cloud with network policies is configured to deny specific VM/Container to VM/Container communication.

Execution steps

1. Deploy two VMs/Containers A and B, in different zones or with different environment.

   EXAMPLE: Zones such as namespaces in Kubernetes, environment such as labels in Kubernetes

2. Define network policies that explicitly deny communication between the two VMs/Containers.
3. Attempt to establish a network connection from VM/Container A to VM/Container B using tools.

   EXAMPLE: tools such as curl or ping in Kubernetes

4. Capture the response or output received.

Expected results

The network connection from VM/Container A to VM/Container B is blocked, indicating that the network policies correctly deny the communication between the VMs/Containers.

18.3.7 Validate network connections from outside the allowed network ranges

Requirement Name: Isolation & secure communication between Applications

Requirement Reference: REQ-SEC-OCLOUD-ISO-1 to REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-VM-C-01, T-VM-C-02, T-VM-C-03, T-VM-C-04, T-VM-C-05, T-VM-C-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Connection_Allowed_Outside

Purpose: The purpose of this test is to ensure that network connections from IP addresses outside the allowed network ranges are denied.

Procedure and execution steps

Preconditions

O-Cloud with network policies is configured to restrict access based on IP ranges.

Execution steps

1. Define network policies that restrict access to certain IP ranges.
2. Attempt to access services or VMs/Containers from IP addresses outside the allowed ranges, either through direct IP access or using service names.
3. Capture the response or output received.

   EXAMPLE: In this test case, the service name refers to the Kubernetes service object's name. The service acts as a load balancer and provides a stable DNS name that can be used to access the pods associated with it. For example, suppose you have a service named my-service that is associated with a set of pods. In the test case, you would attempt to access my-service from IP addresses outside the allowed ranges. This can be done using tools like curl or by making HTTP requests to http://my-service.

Expected results

Access attempts from outside the allowed IP ranges is denied, and the response or output indicates a connection failure.

18.3.8 Exploitation of O-Cloud component vulnerabilities

Requirement Name: O-Cloud hardening and secure configuration.

Requirement Reference: REQ-SEC-OCLOUD-SU-1, clause 5.1.8.5.1, REQ-SEC-SYS-1, clause 5.3.6.1, REQ-SEC-O-CLOUD-ISO-7, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O-RAN-02, T-VM-C-01, T-VM-C-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Vulnerability_Scanning

Purpose: The purpose of this test is to identify and assess the presence of vulnerabilities in O-Cloud components and evaluate the effectiveness of their mitigation measures.

Procedure and execution steps

Preconditions

- O-Cloud with various O-Cloud components deployed.

  EXAMPLE: in the context of Kubernetes, components include etcd, kubelet

- O-Cloud with security best practices is implemented.

Execution steps

1. Identify known vulnerabilities specific to the versions of used O-Cloud components using vulnerability scanning tools.
2. If known vulnerabilities exist, follow publicly available exploit scenarios or utilize penetration testing tools to attempt exploitation.
3. Monitor the O-Cloud and capture any signs of successful exploitation or vulnerabilities being triggered.

Expected results

For step 1), no known vulnerabilities exist in the O-Cloud

For step 2), mitigation measures, such as applying security patches or configuration changes are implemented to address known vulnerabilities.

For step 3), Exploit attempts fails to compromise the O-Cloud.

18.3.9 Identification and remediation of insecure configuration settings

Requirement Name: O-Cloud hardening and secure configuration

Requirement Reference: REQ-SEC-OCLOUD-SU-1, clause 5.1.8.5.1, REQ-SEC-SYS-1, Clause 5.3.6.1 REQ-SEC-O-CLOUD-ISO-7, Clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Threat References: T-O-RAN-02, T-VM-C-01, T-VM-C-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Insecure_Configuration

Purpose: The purpose of this test is to identify insecure configuration settings in the O-Cloud and verify the effectiveness of remediation measures.

Procedure and execution steps

Preconditions

O-Cloud with a configuration review and hardening process in place.

Execution steps

1. Review the O-Cloud configuration for common security misconfigurations, such as weak authentication settings, insecure defaults, or unencrypted communication.
2. Identify and simulate scenarios where insecure configurations can be exploited.
3. Monitor the O-Cloud and capture any signs of insecure configurations being successfully exploited.

Expected results

- The O-Cloud configuration is hardened and securely configured to mitigate common security misconfigurations.
- Insecure scenarios are identified and remediated, ensuring a hardened O-Cloud. If insecure scenarios are rectified, testing has to be repeated.

18.3.10 Validation of logging and monitoring for security incidents

Requirement Name: logging and monitoring for security incidents

Requirement Reference: REQ-SEC-OCLOUD-O2dms-4, clause 5.1.8.9.1.1 REQ-SEC-OCLOUD-O2ims-4, clause 5.1.8.9.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-GEN-01, T-GEN-02, T-GEN-03, T-GEN-04, T-GEN-05, T-GEN-06

DUT/s: O-Cloud

Test Name: TC_OCloud_Security_Logs

Purpose: The purpose of this test is to validate logging and monitoring for security incidents.

Procedure and execution steps

Preconditions

O-Cloud with centralized logging and monitoring systems is implemented.

Execution steps

1. Simulate security incidents such as unauthorized access attempts or Application compromise:
    - Attempt to perform unauthorized API requests or access o-Cloud resources without appropriate permissions.
    - Mimic a compromised Application by running malicious code or attempting privilege escalation.
    - Monitor the O-Cloud and capture any signs of security incidents being logged or detected.
2. Monitor the O-Cloud for detection and alerting of security events:
    - Configure the logging and monitoring systems to capture relevant security events, such as failed authentication attempts, privilege escalation, or anomalous Application behaviour.
    - Monitor the O-Cloud in real-time or periodically to detect the simulated security incidents.
    - Verify that the monitoring system generates alerts or notifications for detected security events.

Expected results

- For the first step, unauthorized access attempts and Application compromise attempts are captured as security events in the logs.
- For the second step, the monitoring system detects and generates alerts for the simulated security incidents.

18.3.11 O-Cloud Privilege Escalation Prevention

Requirement Name: O-Cloud privilege escalation prevention

Requirement Reference: REQ-SEC-OCLOUD-ISO-1, REQ-SEC-OCLOUD-ISO-3, clause 5.1.8.4.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-01

DUT/s: O-Cloud

Test Name: TC_OCloud_Privilege_Escalation_Prevention

Purpose: To verify that privilege escalation is effectively prevented in O-Cloud by enforcing security policies (EXAMPLE: PodSecurity admission (PSA).

Procedure and execution steps

Preconditions:

- O-Cloud with security policies (EXAMPLE: Kubernetes cluster with PodSecurity admission (PSA)) configured and enforced.

Execution steps:

1. Attempt to create a VM or Container that attempts to escalate privileges

   EXAMPLE: in Kubernetes by specifying the **hostPID: true or hostNetwork: true** field in the pod's security context.

2. Monitor the API server response and logs

Expected results:

- For step 1: The VM or Container creation request is denied by the O-Cloud API server.
- For step 2: The O-Cloud API server logs should show a message indicating a violation of the security policies.

Expected format of evidence:

- Screenshot: Displaying the API server's response to the VM or Container creation attempt.
- Executed Commands: Details of the VM or Container creation parameters and security context used.
- API Server Logs: Messages indicating a violation of security policies.
- Conclusion Logs: Indicating whether the test passed or failed based on expected results.

18.3.12 O-Cloud mutual authentication

Requirement Name: O-Cloud mutual authentication between applications

Requirement Reference: REQ-SEC-OCLOUD-ISO-2, clause 5.1.8.4.2, SEC-CTL-O-CLOUD-ISO-2, clause 5.1.8.4.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-GEN-04

DUT/s: O-Cloud

Test Name: TC_OCLOUD_MUTUAL_AUTHENTICATION

Purpose: To verify that communication between different applications running on the O-Cloud is secured through mutual TLS (mTLS) authentication.

Procedure and execution steps

Preconditions:

- Environment: An O-Cloud is set up with two or more deployed applications

  EXAMPLE: A cluster with two applications, each running in separate pods.

- mTLS configuration: Deployed applications in the O-Cloud are configured with mTLS as defined in [2] clause 4.2.
- Tools setup: Network sniffers, packet capture and TLS inspection tools are deployed to monitor and verify TLS handshake process.

- Valid, expired, and revoked certificates are prepared for testing. Ensure that the infrastructure for checking revoked certificates (CRL/OCSP server) is operational and accessible to the applications.

Execution steps:

- Initiate mTLS-secured sessions between applications and capture the TLS handshake process.
- Validate the exchange and authentication of certificates using TLS inspection tools.

1. Attempt connections using valid certificates and record the outcomes.
2. Attempt connections using expired certificates and record the outcomes.
3. Attempt connections, confirm that applications recognize the certificates as revoked (evidenced by querying the CRL or OCSP server), and record outcomes.
4. Attempt to establish an unauthenticated session (no certificate presented) and record the outcome.

Expected results:

1. mTLS sessions are successfully established only with valid certificates.
2. mTLS session establishment with expired certificates fails.
3. mTLS session establishment with revoked certificates fails.
4. Any attempt to initiate an unauthenticated session (without presenting a certificate) is rejected.

Expected format of evidence:

Logs from network sniffers, packet captures and TLS inspection tools showing:

1. Successful mTLS handshakes with valid certificates.
2. Rejections due to expired certificates, ensuring the application appropriately identifies and handles certificates beyond their validity period.
3. Rejections due to revoked certificates, with specific emphasis on the application's process for recognizing revoked certificates through mechanisms such as CRL (Certificate Revocation List) and OCSP (Online Certificate Status Protocol) queries.
4. Rejection of unauthenticated sessions, demonstrating the system's enforcement of mTLS authentication by not allowing sessions without certificate authentication.

18.3.13 O-Cloud authorization

Requirement Name: O-Cloud authorization

Requirement Reference: REQ-SEC-OCLOUD-ISO-2, clause 5.1.8.4.2, SEC-CTL-O-CLOUD-ISO-3, clause 5.1.8.4.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-GEN-04

DUT/s: O-Cloud

Test Name: TC_OCLOUD_AUTHORIZATION

Purpose: To verify that authorization policies are correctly enforced according to the least privilege principle.

Procedure and execution steps

Preconditions:

- Environment: An O-Cloud is set up with two or more applications

  EXAMPLE: A cluster with two applications, each running in separate pods.

- Access control configuration: Access control policies are defined and applied to applications, ensuring permissions are scoped to the minimum necessary privileges.

- Tools setup: auditing tools are deployed to monitor and verify access control policies.

  EXAMPLE: Kubernetes audit logs for access control verification.

Execution steps:

- Map out the actions each application can perform on another according to the access control policies.
  - EXAMPLE: using ‘kubectl describe role’ and ‘kubectl describe rolebinding’ to detail the actions each application is permitted to perform on another under the access control policies.

1. Perform an action that is allowed by the access control policy and record the outcomes. Validate that permitted actions align with the mapped policies.
   - EXAMPLE: using ‘kubectl auth can-i’ to validate that the action is permitted.
2. Attempt actions that are not permitted by the access control policies and record the outcomes.
   - EXAMPLE: using ‘kubectl auth can-i’ to confirm that actions beyond the scope of granted permissions are denied.

Expected results:

1. All actions that are explicitly granted by the access control policies are successfully performed without errors. Audit logs reflect the correct enforcement of these policies.
2. Any attempts to perform actions outside the scope of granted permissions are denied, with audit logs accurately recording these access denials in accordance with the access control policies.

EXAMPLE: Monitor Kubernetes audit logs to capture policy decisions, noting both allowed and denied actions.

Expected format of evidence:

Detailed logs capturing:

1. Allowed actions, correlating with the defined access control policies.
2. Denied actions, specifically those attempted outside the granted permissions, highlighting the effective enforcement of access control policies.

18.4 Application deployment by O-Cloud

18.4.1 Verification of Application artifacts with valid signature by O-Cloud during deployment

Requirement Name: Verification of Application artifacts by O-Cloud during deployment

Requirement Reference: REQ-SEC-OCLOUD-PKG-2, clause 5.1.8.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-IMG-01, T-IMG-04, T-AppLCM-02

DUT/s: O-Cloud

Test Name: TC_OCloud_Valid_Signature_Verification_During_Deployment

Purpose: To verify the O-Cloud ability to check the cryptographic signature of an Application artifact during the process of deployment, ensuring the integrity of Applications deployed in the O-Cloud when the signature is valid.

Procedure and execution steps

Preconditions:

- O-Cloud configured to enforce artifact verification.
- Valid Application artifact with associated signatures.
- The Solution provider certificate (used to sign the Application artifact) is issued by a CA trusted by the Service provider, and the root certificate of the CA is pre-installed at O-Cloud.

Execution steps:

a. Try creating a VM or Container (EXAMPLE: Kubernetes Pod) specification using the valid Application artifact and deploy the VM or Container.

b. Monitor the O-Cloud logs (EXAMPLE: Kubelet logs) for any signature verification events related to the deployment.

c. Verify that the O-Cloud (EXAMPLE: Kubelet) successfully verifies the cryptographic signature of the Application artifact.

Expected results:

a. Logs show entries related to the signature verification, and the O-Cloud successfully verifies the cryptographic signature of the application artifact.

b. The VM or Container is successfully created and deployed without issues.

Expected format of evidence:

- Screenshot: Displaying the O-Cloud's response to the VM or Container deployment attempt, including the signature verification result.
- Executed Commands: Details of the VM or Container creation parameters, including the Application artifact and its cryptographic signature.
- O-Cloud Logs: Messages indicating signature verification events related to the deployment..

18.4.2 Verification of Application artifacts with incorrect signature by O-Cloud during deployment

Requirement Name: Verification of Application artifacts by O-Cloud during deployment

Requirement Reference: REQ-SEC-OCLOUD-PKG-2, clause 5.1.8.2.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-IMG-01, T-IMG-04, T-AppLCM-02

DUT/s: O-Cloud

Test Name: TC_OCloud_Incorrect_Signature_Verification_During_Deployment

Purpose: To verify the O-Cloud ability to detect and reject an incorrect cryptographic signature of an Application artifact during deployment, ensuring the integrity of Applications deployed in the O-Cloud when the signature is invalid.

Procedure and execution steps

Preconditions:

- O-Cloud configured to enforce artifact verification.
- Application artifact with an incorrect associated signature.
- The Solution provider certificate (used to sign the Application artifact) is issued by a CA trusted by the Service provider, and the root certificate of the CA is not pre-installed at O-Cloud.

Execution steps:

a. Try creating a VM or Container (EXAMPLE: Kubernetes Pod) specification using the artifact with an incorrect signature and deploy the VM or Container.

b. Monitor the O-Cloud logs (EXAMPLE: Kubelet logs) for any signature verification events related to the deployment.

c. Verify that the O-Cloud (EXAMPLE: Kubelet) detects the incorrect cryptographic signature and denies the deployment of the VM or Container.

Expected results:

Logs show entries related to the signature verification , and the O-Cloud detects the incorrect cryptographic signature, denying the VM or Container deployment.

Expected format of evidence:

- Screenshot: Displaying the O-Cloud's response to the VM or Container deployment attempt, including the denial due to the incorrect cryptographic signature.
- Executed Commands: Details of the VM or Container creation parameters, including the Application artifact and its cryptographic signature.
- O-Cloud Logs: Messages indicating failed signature verification events related to the deployment.

18.5 Resource Management and enforcement in O-Cloud

18.5.1 O-Cloud Resource Consumption Limit Enforcement

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Resource_Consumption_Limit_Enforcement

Purpose: To verify the DUT is able to ensure that resources (CPU, memory, etc.) consumed by VMs or Containers are within the defined limits, preventing any single application from monopolizing the system's resources.

Procedure and execution steps

Preconditions:

- O-Cloud environment with resource quotas and limits enforced.
- A configured SMO to set and enforce resource quotas and limits.

Execution steps:

1. **Set up resource quotas and limit ranges**:
   - Create a dedicated isolated environment for testing.

- Define a resource quota for the environment, specifying the maximum allowed CPU and memory.
- Define a limit range to set default request and limit values for resources.

2. **Attempt to deploy a VM or Container that requests resources beyond the defined limits**:
   - Create a VM or Container configuration that requests resources exceeding the set limits.
   - Try to deploy the VM or Container in the test environment.
3. **Monitor the deployment status and logs**:
   - Check the deployment status of the VM or Container.

Expected results:

- For step 1: Confirmation that a dedicated isolated environment for testing has been setup and both resource quota and limit range have been established.
- For step 2: The deployment request for the VM or Container is denied or remains in a "Pending" or equivalent state.
- For step 3: Logs or descriptions should show a message indicating a violation of the resource quotas or limits.

Expected format of evidence:

- Configuration Details: Information on the set resource quotas and limit ranges, including the maximum allowed CPU and memory.
- Executed Commands: Details of the VM or Container creation parameters, specifically the requested resources.
- O-Cloud Logs: Messages indicating any violations of the resource quotas or limits during the deployment attempt.
- Deployment Status: Logs or screenshots showing the status of the VM or Container deployment, especially if it's denied or remains in a "Pending" state due to resource constraints.

EXAMPLE using Kubernetes:

1. **Set up resource quotas and limit ranges in Kubernetes**:
   - Create a namespace: **kubectl create namespace test-limits**
   - Apply a ResourceQuota and LimitRange as previously detailed.
2. **Attempt to deploy a Pod in Kubernetes**:
   - Create a pod configuration (**resource-hog-pod.yaml**) that requests excessive resources.
   - Deploy using: **kubectl apply -f resource-hog-pod.yaml**
3. **Monitor the deployment status and logs in Kubernetes**:
   - Check pod status: **kubectl get pods -n test-limits**
   - Describe the pod for details: **kubectl describe pod resource-hog -n test-limits**

18.5.2 O-Cloud Storage Volume Limit Enforcement

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1 , O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Storage_Volume_Limit_Enforcement

Purpose: To verify the DUT is able to limit the storage volume allocations for applications predefined in a O-Cloud environment.

Procedure and execution steps:

Preconditions:

- O-Cloud environment with storage volume configurations.

- A configured SMO to set and enforce resource quotas for storage.

Execution steps:

1. **Set up Storage Volume Quotas**:
   - Create a dedicated isolated environment for testing.
   - Define a storage volume quota for the environment, specifying the maximum allowed storage volume size.
2. **Attempt to allocate a storage volume beyond the defined limits**:
   - Create a configuration that requests a storage volume size exceeding the set limits.
   - Deploy the configuration in the test environment.
3. **Monitor the storage allocation status and logs**:
   - Check the status of the storage allocation.

Expected results:

- For step 1: Confirmation that a dedicated isolated environment for testing has been setup and storage volume quota has been defined.
- For step 2: The storage volume allocation request is denied.
- For step 3: Logs or descriptions should show a message indicating a violation of the storage quotas.

Expected format of evidence:

- Configuration Details: Information on the set storage volume quotas, including the maximum allowed storage volume size.
- Executed Commands: Details of the storage volume allocation parameters, specifically the requested storage size.
- O-Cloud Logs: Messages indicating any violations of the storage volume quotas during the allocation attempt.
- Allocation Status: Logs or screenshots showing the status of the storage volume allocation, especially if it is denied due to exceeding the set limits.

EXAMPLE using Kubernetes:

- Create a namespace: kubectl create namespace test-storage
- Apply a ResourceQuota for storage:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-quota
  namespace: test-storage
spec:
  hard:
    requests.storage: 10Gi
```

- Apply the ResourceQuota: kubectl apply -f storage-quota.yaml
- Create and deploy a PersistentVolumeClaim (PVC) requesting 15Gi.
- Monitor the PVC status and logs.

18.5.3 O-Cloud CPU Overcommit Prevention

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_CPU_Overcommit_Prevention

Purpose: To verify that the O-Cloud does not overcommit CPU resources, leading to performance degradation or system instability.

Procedure and execution steps

Preconditions:

- O-Cloud with CPU allocation settings.
- A configured SMO to manage CPU overcommitment.

Execution steps:

1. **Set CPU Overcommit Ratios:**
    - Create a dedicated isolated environment for testing.
    - Define CPU overcommit ratios.
2. **Attempt to deploy multiple applications:**
    - Sequentially deploy applications until the CPU limits are reached based on the overcommit ratios.
    - Monitor the CPU utilization of each deployed application.
3. **Monitor the deployment status and CPU utilization metrics:**
    - Check the deployment status of the applications.
    - Monitor CPU utilization metrics.

Expected results:

- For step 1: Confirmation that a dedicated isolated environment for testing has been setup and CPU overcommit ratios has been defined.
- For step 2: Applications should not be deployed beyond the capacity determined by the CPU overcommit ratios.
- For step 3: CPU utilization metrics should remain stable and within acceptable thresholds.

Expected format of evidence:

- Configuration Details: Information on the set CPU overcommit ratios.
- Executed Commands: Details of the application deployments and their respective CPU utilization.
- O-Cloud Logs: Messages indicating any violations of the CPU overcommit ratios during application deployments.
- Deployment Status: Logs or screenshots showing the status of the application deployments, especially if any are denied due to reaching CPU limits.
- CPU Utilization Metrics: Graphs or logs showing the CPU utilization of each deployed application, ensuring they remain within acceptable thresholds.

EXAMPLE using Kubernetes:

1. **Set CPU Overcommit Ratios:**
    - Manage CPU overcommitment in Kubernetes by setting CPU requests and limits on Pods.
2. **Attempt to deploy multiple applications:**
    - Deploy a Pod with a CPU request of **‘500m’** (half a CPU core) and a limit of **‘1’** (one full CPU core).
3. **Monitor the deployment status and CPU utilization metrics:**
    - Use **‘kubectl describe node <NODE_NAME>’** to view CPU allocation and utilization.
    - Monitor CPU metrics using tools like Prometheus.

18.5.4 O-Cloud Memory Overcommit Prevention

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Memory_Overcommit_Prevention

Purpose: To verify that the O-Cloud does not overcommit memory resources.

Procedure and execution steps

Preconditions:

- O-Cloud with memory allocation settings.
- A configured SMO to manage memory overcommitment

Execution steps:

1. **Set Memory Overcommit Ratios:**
    - Create a dedicated isolated environment for testing.
    - Define memory overcommit ratios.
2. **Attempt to deploy applications:**
    - Sequentially deploy applications until memory limits are reached based on the overcommit ratios.
    - Monitor memory utilization of each deployed application.
3. **Monitor deployment status and memory utilization metrics:**
    - Check deployment status of the applications.
    - Monitor memory utilization metrics.

Expected results:

- For step 1: Confirmation that a dedicated isolated environment for testing has been setup and memory overcommit ratios has been defined.
- For step 2: Applications should not be deployed beyond the capacity determined by the memory overcommit ratios.
- For step 3: Memory utilization should remain stable and within acceptable thresholds.

Expected format of evidence:

- Configuration Details: Information on the set memory overcommit ratios.
- Executed Commands: Details of the application deployments and their respective memory use.
- O-Cloud Logs: Messages indicating any violations of the memory overcommit ratios during application deployments.
- Deployment Status: Logs or screenshots showing the status of the application deployments, especially if any are denied due to reaching memory limits.
- Memory Use Metrics: Graphs or logs showing the memory utilization of each deployed application, ensuring they remain within acceptable thresholds.

EXAMPLE using Kubernetes:

1. **Set Memory Overcommit Ratios:**
    - Manage memory overcommitment in Kubernetes by setting memory requests and limits on Pods.
2. **Attempt to deploy applications:**
    - Deploy a Pod with a memory request of **'256Mi'** and a limit of **'512Mi'**.
3. **Monitor deployment status and memory utilization metrics:**

- Use **kubectl describe node <NODE_NAME>** to view memory allocation and utilization.
- Monitor memory metrics using tools like Prometheus.

18.5.5 O-Cloud Network Overcommit Prevention

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-OCLOUD-ISO-6, clause 5.1.8.4.2 REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Network_Overcommit_Prevention

Purpose: To verify that the O-Cloud environment does not overcommit network bandwidth.

Procedure and execution steps

Preconditions:

- O-Cloud environment with network configurations.
- Tools to manage network overcommitment.

Execution steps:

1. **Set Network Overcommit Ratios:**
   - Define network bandwidth overcommit ratios.
2. **Attempt to utilize network bandwidth:**
   - Deploy applications designed to generate high network traffic.
   - Monitor network traffic.
3. **Monitor network traffic metrics:**
   - Check network traffic metrics for the applications.

Expected results:

- For step 1: Confirmation that network bandwidth overcommit ratios has been defined.
- For steps 2 and 3: Applications' network traffic should be throttled or limited once the bandwidth determined by the overcommit ratios is reached.

Expected format of evidence:

- Configuration Details: Information on the set network bandwidth overcommit ratios.
- Executed Commands: Details of the application deployments and their respective network traffic generation.
- O-Cloud Logs: Messages indicating any violations of the network overcommit ratios during high network traffic.
- Deployment Status: Logs or screenshots showing the status of the application deployments, especially if network traffic is throttled or limited.
- Network Traffic Metrics: Graphs or logs showing the network traffic of each deployed application, ensuring they remain within the set bandwidth limits.

EXAMPLE using Kubernetes:

1. **Set Network Overcommit Ratios:**
   - Native Kubernetes doesn't offer direct network bandwidth controls. However, third-party plugins like '**Calico**' or '**Cilium**' can be used to set network policies that limit bandwidth.
2. **Attempt to utilize network bandwidth:**

- Deploy a Pod and apply a network policy that limits its bandwidth.

3. **Monitor network traffic metrics:**
    - Use monitoring tools integrated with the network plugin (e.g., **'calicoctl'** for Calico) to observe the network traffic metrics.

18.5.6 O-Cloud Storage Overcommit Prevention

Requirement Name: Resource Management and enforcement in O-Cloud

Requirement Reference: REQ-SEC-LCM-SD-1 to REQ-SEC-LCM-SD-4, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VM-C-05, T-AppLCM-04, T-AppLCM-05

DUT/s: O-Cloud

Test Name: TC_OCloud_Storage_Overcommit_Prevention

Purpose: To verify that the O-Cloud does not overcommit storage resources.

Procedure and execution steps

Preconditions:

- O-Cloud environment with storage configurations.
- A configured SMO to manage Storage overcommitment.

Execution steps:

1. Set Storage Overcommit Ratios:
    - Define storage overcommit ratios.
2. Attempt to allocate storage beyond defined limits:
    - Deploy applications that request storage space.
    - Monitor storage allocation and utilization.
3. Monitor storage allocation and utilization metrics:
    - Check storage metrics for the applications.

Expected results:

- For step 1: Confirmation that storage overcommit ratios has been defined.
- For step 2: Storage allocations should not exceed the capacity determined by the storage overcommit ratios.
- For step 3: Storage usage should remain stable and within acceptable thresholds.

Expected format of evidence:

- Configuration Details: Information on the set storage overcommit ratios.
- Executed Commands: Details of the application deployments and their respective storage requests.
- O-Cloud Logs: Messages indicating any violations of the storage overcommit ratios during storage allocation.

- Deployment Status: Logs or screenshots showing the status of the application deployments, especially if storage allocations are denied or limited.
- Storage usage Metrics: Graphs or logs showing the storage utilization of each deployed application, ensuring they remain within the set storage limits.

EXAMPLE using Kubernetes:

1. **Set Storage Overcommit Ratios:**
   - Use PersistentVolumeClaims (PVCs) with specific storage requests. Overcommitment can occur if the total storage requested by PVCs exceeds the actual available storage.
2. **Attempt to allocate storage beyond defined limits:**
   - Deploy a Pod that uses a PVC requesting more storage than available on the PersistentVolume (PV).
3. **Monitor storage allocation and utilization metrics:**
   - Use '**kubectl get pvc**' and '**kubectl get pv**' to monitor storage allocations.
   - Monitor storage metrics using tools like Prometheus.

NOTE: Below are the general guidelines to ensure effective monitoring across all test cases:

1. Immediate Feedback Expectation: Upon executing any test case, especially those involving security or resource constraints, immediate feedback is typically anticipated. This feedback can be in the form of API responses, system alerts, or log entries.
2. Monitoring Duration: While immediate feedback is expected, it's recommended to monitor for an additional 1-3 minutes post-execution to capture any delayed logs, alerts, or system responses. This ensures that asynchronous events or alerts are not missed.
3. Tools and Logs: Use appropriate monitoring tools, logging systems, or commands (e.g., In Kubernetes like kubectl describe or kubectl logs) to gain insights into the test execution. Ensure that these tools are set up in advance and are accessible to the testing team.

18.6 Secure Update

18.6.1 O-Cloud Infrastructure Software Package Integrity - Positive

Requirement Name: O-Cloud software images authenticity and Integrity

Requirement Reference: SEC-CTL-ALM-PKG-1B, clause 5.3.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Ensure Authenticity and Integrity of O-Cloud Software Images

Threat References: T-GEN-01

DUT/s: O-Cloud

Test Name: TC_OCloud_Software_Package_Integrity

Purpose: To verify the O-Cloud software image authenticity and integrity.

Procedure and execution steps

Preconditions

- Signed O-Cloud software package as per Clause 5 of O-RAN Security Protocols Specifications [2]

- All necessary artifacts of the O-Cloud software image (public key, digitally signed certificates, signature (Signed hash) encryption key if any for security-sensitive artifacts) are provided.

  EXAMPLE: O-Cloud software includes AAL drivers, IMS, DMS, Host OS, Hypervisor, Container Engine.

Execution steps

1. The Tester is properly authenticated and have the required access privileges to perform the test activity.
2. The tester shall verify the authenticity and integrity of the list of images. The O-Cloud software package shall be verified with the provided X.509 certificate and signature provided by the O-Cloud Software Provider. The cryptographic hash of the software image is calculated and verified against the hash in the signature by the Software Provider.
3. On successful validation of O-Cloud software images in Step 2, the Service Provider shall sign the verified O-cloud software image with its private key and onboard it to the SMO.
4. The newly signed O-Cloud images shall be onboarded to the O-cloud Image Repository.
5. The tester shall verify the digital signature of the O-Cloud software image bundle provided by the Software and Service Provider before deployment.
6. Monitor the SMO logs for signature verification events related to the upgrade.
7. Monitor the O-Cloud logs for any signature verification events related to the upgrade.

Expected results

Logs show that the software package integrity check has been executed for the O-Cloud software at each stage

The signature validation for the O-Cloud software image during onboarding are checked and is successful.

Expected format of evidence:

Snapshots captured in SMO logs regarding the Signature verification success.

Logs from SMO and O-Cloud (O2ims logs) to indicate the successful signature verification from the Software Provider.

18.6.2 O-Cloud Infrastructure Software Package Integrity Failure – Negative

Requirement Name: O-Cloud software images authenticity and Integrity

Requirement Reference: SEC-CTL-ALM-PKG-1B, clause 5.3.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Ensure Authenticity and Integrity of O-Cloud Software Images

Threat References: T-GEN-01

DUT/s: O-Cloud

Test Name: TC_OCloud_Software_Package_Integrity_Failure

Purpose: To verify the O-Cloud software image authenticity and integrity validation failure for invalid O-cloud software image.

Procedure and execution steps

Preconditions

- O-Cloud software package obtained from the Software Provider.

- All necessary artifacts of the O-Cloud software image (public key, digitally signed certificates, Signature (signed hash) encryption key if any for security-sensitive artifacts) are provided.

  EXAMPLE: O-Cloud software includes AAL drivers, IMS, DMS, Host OS, Hypervisor, Container Engine.

Execution steps

1. The Tester is properly authenticated and has the required access privileges to perform the upgrade activity.
2. Attempt to validate the O-Cloud Software with the wrong public key.
3. Verify that the SMO detects the incorrect cryptographic signature and does not allow onboarding of the software package.
4. Monitor the SMO logs for any signature verification events related to the software integrity check.

Expected results

Logs show that the software package integrity check has failed.

The O-cloud software image shall not be onboarded due to the software integrity failure.

Expected format of evidence:

Snapshots captured in SMO regarding the signature verification failure.

SMO Logs: Onboarding failure logs to indicate that integrity failure for the O-Cloud software Package.

18.6.3 Secure Update procedure for O-Cloud Platform – Positive

Requirement Name: Secure update of O-Cloud software at the infrastructure level layer.

Requirement Reference: SEC-CTL-ALM-PKG-1B, clause 5.3.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Ensure secure update of O-Cloud Software Images at the Infrastructure level.

Threat References: T-GEN-01

DUT/s: O-Cloud

Test Name: TC_SECURE_UPDATE_OF_O-CLOUD_PLATFORM

Purpose: To verify the secure update procedure for the O-Cloud Infrastructure using verified O-cloud software image.

Procedure and execution steps

Preconditions

- Verified O-Cloud software package obtained from Service Provider.
- All necessary artifacts of the O-Cloud software image (public key, digitally signed certificates, encryption key if any for security-sensitive artifacts) shall be provided.
- All necessary documents related to the Upgrade procedure of the O-Cloud components shall be available.
- All necessary dependencies for O-Cloud software packages are considered prior to update.
- All documents related to backward compatibility are made available by the O-Cloud Software provider.

  EXAMPLE: O-Cloud software includes AAL drivers, IMS, DMS, Host OS, Hypervisor, Container Engine.

Execution steps

1. The Tester is properly authenticated and has the required access privileges to perform the upgrade activity.
2. The O-Cloud Platform to ensure image verification.
3. The tester performs all the necessary pre-upgrade steps on the O-Cloud Platform to ensure successful update.

EXAMPLE:

   a. Back up any important components, such as app-level state stored in a database, or state of critical nodes. EXAMPLE: Snapshots, Clones

4. As per the Upgrade documentation, the tester shall perform the upgrade of the O-Cloud Platform components.

EXAMPLE:

   a. Phased upgrades for service availability.
   b. Stage the Upgrade procedure: upgrade control plane nodes and upgrade the worker nodes.

5. Monitor the O-Cloud logs (EXAMPLE: O2ims logs) for the update steps performed on the platform.
6. Perform a Post-Update Audit to verify the status of the O-Cloud Platform.
7. Verify in the SMO that the software version for the O-Cloud platform components is updated to the required version.

Expected results

The version of the O-Cloud software components is updated to the required version.

EXAMPLE: AAL driver version, IMS version, DMS version, Host OS version, Hypervisor, Container Engine.

Expected format of evidence:

O-Cloud logs: Log captures indicating the Steps performed during the Update.

Snapshot: Executed command on CLI, GUI, API server

SMO Log: Notification on the successful upgrade of the O-Cloud components.

18.6.4 Secure Update failure for O-Cloud Platform – Negative

Requirement Name: Rollback to the previous version on the unsuccessful update of the O-Cloud Platform.

Requirement Reference: SEC-CTL-ALM-PKG-1B, clause 5.3.2.1.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: The O-Cloud platform maintains its initial state if updates fail or incidents occur during update.

Threat References: T-GEN-01

DUT/s: O-Cloud

Test Name: TC_SECURE_UPDATE_FAILURE_OF_O-CLOUD_PLATFORM

Purpose: To verify on failure of secure Update procedure for the O-Cloud platform, it shall remain in its initial working state.

Procedure and execution steps

Preconditions

- Verified O-Cloud platform software package obtained from Service Provider.
- All necessary artifacts of the O-Cloud platform software image (public key, digitally signed certificates, encryption key if any for security-sensitive artifacts) shall be provided.
- All necessary documents related to the Upgrade procedure of the O-Cloud components are made available.

EXAMPLE: O-Cloud software includes AAL drivers, IMS, DMS, Host OS, Hypervisor, Container Engine.

Execution steps

1. The Tester is properly authenticated and have the required access privileges to perform the upgrade activity.
2. The O-Cloud Platform to ensure image verification.
3. The tester performs all the pre-upgrade steps on the O-Cloud Platform.

   EXAMPLE:

   a. Back up any important components, such as app-level state stored in a database, state of critical nodes.
4. As per the Upgrade documentation, the tester shall stage and perform the upgrade of the O-Cloud Platform

   EXAMPLE:

   a. Phased upgrades for service availability.

   b. Stage the Upgrade procedure: Upgrade control nodes and upgrade the worker nodes.
5. Attempt to simulate an upgrade failure scenario. EXAMPLE: Unexpected upgrade termination, Abrupt Power failure, Network disruption.
6. Monitor the O-Cloud logs (EXAMPLE: O2ims logs) for the upgrade steps performed on the platform.
7. Perform a Post-Update Audit to verify the status of the O-Cloud Platform.
8. The O-Cloud Platform shall automatically roll-back to its previous version.
9. Verify in the SMO that the software version for the O-Cloud platform components remains the same as the previous version.

Expected results

The O-Cloud Platform shall automatically roll-back to its previous version and initial working state.

SMO logs to indicate the notification of the Update failure and the version of the O-Cloud components maintains its initial state.

Expected format of evidence:

O-Cloud and SMO logs: Log captures indicating the Steps performed during the Upgrade and the version of the O-Cloud components

Snapshot: Executed command on CLI, GUI, API server

18.7 Secure Storage

18.7.1 Sensitive data protection in O-Cloud

Requirement Name: Sensitive data protection in O-Cloud

Requirement Reference: REQ-SEC-ALM-PKG-13, clause 5.3.2.1.1, REQ-SEC-OCLOUD-SS-1, clause 5.1.8.6.1, SEC-CTL-OCLOUD-SS-1, clause 5.1.8.6.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-GEN-05

DUT/s: O-Cloud

Test Name: TC_DATA_PROTECTION_OCLOUD

Purpose: To validate that the O-Cloud ensures the integrity and confidentiality of sensitive data at rest, in use, and in transit, using state-of-the-art encryption and security practices.

Procedure and execution steps:

Preconditions:

- O-Cloud operational with a simulated deployment of workloads.

Execution steps:

1. **Data at rest:**
    a) Store simulated sensitive data (e.g., secrets) in O-Cloud storage.
    b) Verify encryption using tools designed to check for industry-standard encryption mechanisms, focusing on confirming that data is encrypted to current security standards.
2. **Data in use:**
    a) Process sensitive data using an application.
    b) Identify write operations involving sensitive data
        1. Employ process monitoring tools to monitor file I/O operations and capture all write activities.
        2. Look for write operations to temporary file paths, cache directories or outside of the application secure storage.
        3. Analyse the context of write operations – are they occurring during a process that handles sensitive data?
    c) Ensure that the application does not log sensitive information
        1. Review the application's logging configuration and log output.
        2. Verify that sensitive data is either not logged or appropriately anonymized/encrypted before being logged.
    d) Check for the use of secure enclaves, if applicable.
3. **Data in Transit:**
    a) Initiate data transfer between O-Cloud services.

b) Use packet-sniffing tools to capture the data packets.

c) Analyse the TLS encrypted data, ensuring TLS is used as specified in O-RAN Security Protocols Specifications [2], clause 4.2.

Expected Results:

1. Sensitive data in O-Cloud storage is encrypted according to current industry standards. Unauthorized access attempts are logged and denied.
2. Data processing is secure, with no plaintext data exposure in logs or disk. Secure enclaves are used where relevant.
3. All data transfers employ TLS as specified in O-RAN Security Protocols Specifications [2], clause 4.2.

Expected Format of Evidence:

1. Screenshots and logs showing encryption validation and the response to unauthorized access attempts.
2. Logs from process monitoring tools demonstrating the handling of sensitive data during processing.
3. Packet capture files confirming the data encryption in transit using TLS as specified in O-RAN Security Protocols Specifications [2], clause 4.2.

18.7.2 Secure data deletion in O-Cloud

Requirement Name: Secure data deletion in O-Cloud

Requirement Reference: SEC-CTL-OCLOUD-SS-2, REQ-SEC-OCLOUD-SS-4, clause 5.1.8.6.1, SEC-CTL-OCLOUD-SS-2, clause 5.1.8.6.2 O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-GEN-05

DUT/s: O-Cloud

Test Name: TC_DATA_DELETION_OCLOUD

Purpose: To ensure that the O-Cloud platform securely deletes data from addressable memory locations that are no longer in use, by overwriting them with specific binary patterns.

Procedure and execution steps:

Preconditions:

- O-Cloud platform operational with workloads.
- Tools for memory analysis are available.

  EXAMPLE: **dd** for Unix/Linux or **sdelete** for Windows environments
- File recovery tools for testing data recoverability after deletion.

  EXAMPLE: **TestDisk**

Execution steps:

1. Data preparation:

   a. Create files with identifiable data patterns.

   EXAMPLE: a file filled with a repeating pattern of '1234'

b. Store these files in the O-Cloud platform's memory or storage system.

2. Data deletion:

   a. Delete the files using the O-Cloud platform's standard deletion process, which should invoke secure deletion process.

3. Verification of secure deletion:

   a. Inspect the memory or storage locations where the files were stored to confirm that the data has been overwritten.

   b. Search for both the original data patterns and the specific overwriting patterns (zeroes, ones, random).

4. Data recovery attempt:

   a. Use data recovery tools to attempt to retrieve the deleted files or any part of them.

   b. Assess if any of the original data or identifiable patterns can be recovered.

Expected Results:

1. Deleted data locations are overwritten with the specified binary patterns.
2. File recovery attempts are not able to reconstruct any meaningful data from these locations.

Expected Format of Evidence:

1. Logs or screenshots from memory analysis tools showing the overwriting patterns.
2. Reports from file recovery tools indicating the failure to recover any meaningful data.

18.7.3 Data isolation in VM/Container reallocation

Requirement Name: Data isolation in VM/Container reallocation

Requirement Reference: REQ-SEC-OCLOUD-SS-3, clause 5.1.8.6.1, O-RAN Security Requirements and Controls Specifications

Requirement Description:

Threat References: T-GEN-05

DUT/s: O-Cloud

Test Name: TC_DATA_ISOLATION_VM_CONTAINER_OCLOUD

Purpose: To verify that the O-Cloud effectively prevents data contained in a resource (like memory or storage) from being accessible after it is de-allocated from one VM/Container and reallocated to another.

Procedure and execution steps:

Preconditions:

- Set up multiple VMs/Containers within the O-Cloud.
- Tools for analyzing memory and storage content

  EXAMPLE: **hexdump**, **dd**, memory inspection tools

Execution steps:

1. Resource allocation and data storage:
    - Allocate a dedicated resource (like a disk volume or memory segment) to a VM/Container.
    - Store known test data in this resource.
2. Resource de-allocation & re-allocation:
    - De-allocate the resource from the first VM/Container.
    - Re-allocate the same resource to a different VM/Container.
3. Data accessibility check:
    - Within the new VM/Container, attempt to access any residual data from the previous allocation.
    - Use data analysis tools to inspect the resource for traces of the previous data.
4. Verification of data isolation:
    - Confirm that no data from the first VM/Container is accessible or present in the resource after re-allocation.

Expected Results:

- No trace of the test data is found in the reallocated resource.
- The new VM/Container doesn't have access to any residual data from the previous allocation.

Expected Format of Evidence:

Logs or screenshots showing the absence of the test data in the re-allocated resource.

18.8 Chain of trust

18.8.1 Chain of Trust verification in static O-Cloud SW

Requirement Name: Support of root of trust and integrity verification of static O-Cloud SW (Firmware and BIOS/UEFI, Bootloader, OS kernel)

Requirement Reference: REQ-SEC-OCLOUD-COT-1, clause 5.1.8.7.1, SEC-CTL-OCLOUD-COT-1, clause 5.1.8.7.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VL-02

DUT/s: O-Cloud

Test Name: TC_OCLOUD_ CHAIN_OF_TRUST_STATIC_SW

Purpose: To confirm the presence and proper functioning of a secure boot process and integrity verification for O-Cloud static SW (Firmware and BIOS/UEFI, Bootloader, OS kernel), using hardware-based or software-based roots of trust mechanisms.

Procedure and execution steps:

Preconditions:

- Ensure the O-Cloud is set up with all components, including hardware, operating system, virtualization layer, and any applications or services running on the O-Cloud.

- Use tools capable of interfacing with the O-Cloud's root of trust mechanism.

  EXAMPLE: TPM management tools for hardware-based roots of trust, Keylime or any equivalent integrity verification system for automated integrity verification.

- Access requirements: Tester needs administrative access to the O-Cloud platform to execute integrity verification commands and to collect integrity verification reports.

Execution steps:

Hardware RoT verification:

Confirm the functionality of the root of trust mechanism in each O-Cloud node, whether it is hardware-based or an equivalent software-based solution.

EXAMPLE: Use a Kubernetes DaemonSet to run tpm2_pcrread on all nodes, which verifies TPM presence and functionality by reading the PCR (Platform Configuration Registers) values. The DaemonSet collects outputs and send them for verification.

Integrity check:

Verify the integrity measurements against known good baselines. These measurements ensure the boot process and static O-Cloud SW integrity.

EXAMPLE:

- Use a securely stored baseline to obtain the expected PCR values for a known secure state of the O-Cloud static SWs. This could involve securely storing PCR values following a clean installation or using manufacturer-provided values.
- Schedule Kubernetes CronJobs to use Keylime for periodic integrity verification. Keylime agents on nodes interact with the TPM to attest the integrity measurements, comparing them against known good values stored in Keylime's verifier.

Report collection and analysis:

Collect integrity reports and analyse them for discrepancies or signs of tampering.

EXAMPLE: Use Keylime's centralized reporting and alerting features to collect and analyse attestation data. Integrate Keylime with an ELK stack deployed within the Kubernetes cluster for enhanced log analysis and visualization of attestation outcomes.

Expected Results:

- All O-Cloud nodes demonstrate the presence of RoT.
- Integrity measurements align with known good baselines.

Expected Format of Evidence:

- Logs confirming RoT presence on each node.
- Logs indicating successful integrity verification against known good baselines.

18.8.2 Chain of Trust verification of dynamic O-Cloud SW

Requirement Name: Support of root of trust and integrity verification of dynamic O-Cloud SW (virtualization layer and workloads)

Requirement Reference: REQ-SEC-OCLOUD-COT-2, clause 5.1.8.7.1, SEC-CTL-OCLOUD-COT-3, clause 5.1.8.7.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-VL-02

DUT/s: O-Cloud

Test Name: TC_OCLOUD_INTEGRITY_VERIFICATION_DYNAMIC_SW

Purpose: To ensure the integrity of dynamic software in the O-Cloud through continuous verification.

Procedure and execution steps:

Preconditions:

- Ensure the O-Cloud is set up with all components, including hardware, operating system, virtualization layer, and any applications or services running on the O-Cloud.
- Use tools capable of interfacing with the O-Cloud's root of trust mechanism.

  EXAMPLE: TPM management tools for hardware-based roots of trust, Keylime to automate integrity measurements and attestation of dynamic software components, integrating with IMA for capturing runtime integrity measurements.
- Access requirements: Tester needs administrative access to the O-Cloud platform to execute integrity verification commands and to collect integrity verification reports.

Execution steps:

Dynamic software verification:

Initiate continuous integrity verification of the dynamic O-Cloud SW, including executable binaries and configuration files used by the container engine and workloads.

EXAMPLE: Implement an IMA policy to measure container images and runtime configurations upon execution. Configure Keylime to monitor the integrity of container runtime environments and deployed containers on Kubernetes nodes. This involves setting up Keylime agents within the cluster that automatically update integrity measurements for dynamic software components and verify them against expected values.

Attestation:

Perform attestation of the container engine configurations and active workloads to detect any unauthorized changes or potential integrity breaches.

EXAMPLE: Deploy Keylime agents on each O-Cloud node to periodically attest the integrity of dynamic SW components based on IMA measurements. Use Keylime to trigger attestation procedures that verify IMA logs against expected integrity measurements for containerized applications.

Report collection and anomaly detection:

Collect attestation reports and analyse them for any discrepancies, unauthorized changes, or signs of tampering in the container engine and workloads.

EXAMPLE: Use Keylime's web interface or API integrated with an ELK stack for logging and monitoring attestation results. Set up alerts for any attestation failures or integrity breaches detected in dynamic software.

Expected Results:

- Dynamic software components are measured upon execution, with their integrity measurements securely recorded.

- Integrity measurements are successfully captured and verified against known good baselines, indicating no unauthorized modifications.

Expected Format of Evidence:

- Logs attesting the integrity of dynamic software components, including any alerts generated for integrity failures.
- Reports detailing the comparison of runtime integrity measurements against known good baselines, demonstrating continuous integrity verification of dynamic O-Cloud dynamic software.

18.9 Secure time synchronization for O-Cloud

Requirement Name: Secure time synchronization for O-Cloud

Requirement Reference: SEC-CTL-OCLOUD-TS-1, SEC-CTL-OCLOUD-TS-3, clause 5.1.8.12.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: It ensures all O-Cloud nodes exclusively connect to an authenticated time synchronization server for Time of Day (ToD) synchronization.

Threat Reference: T-TS-01

DUT/s: O-Cloud

Test Name: TC_OCLOUD_SECURE_TIME_SYN

Purpose: To verify that all O-Cloud nodes are configured to exclusively connect to an authenticated time synchronization server for Time of Day (ToD) synchronization.

Procedure and execution steps

Preconditions:

- A list of allowed secure time synchronization servers
- Configuration of O-Cloud nodes

Execution steps:

1. Access the configuration of each O-Cloud node to review the time synchronization settings.
2. Check that each O-Cloud node is configured to only connect to one or more time synchronization servers from the list of allowed secure time synchronization servers.
3. Ensure that no other time synchronization servers are configured.
4. Confirm that the cryptographic keying material for verifying the server’s authenticity is present in the configuration.

Expected results:

- Each O-Cloud node is configured to connect only to the allowed secure time synchronization server(s), and the cryptographic keying material for verifying each server’s authenticity is present in the configuration.

Expected format of evidence:

- Screenshots showing the O-Cloud node configuration settings with the allowed time synchronization server(s) details.

19 Security test of VNF/CNF

19.1 Overview

This clause contains security tests to validate the security protection mechanism specific to O-RAN architecture elements which are virtualized/containerized and deployed on the O-Cloud/SMO.

19.2 Executive environment protection

Requirement Name: secure executive environment provision

Requirement Reference: REQ-SEC-LCM-SD-5, REQ-SEC-LCM-SD-6, REQ-SEC-LCM-SD-7, clause 5.3.2.3.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: It ensures all O-RAN architecture elements only consume the resources from its defined resource quotas.

Threat References: T-AppLCM-04, T-AppLCM-05

DUT/s: O-CU, O-DU, Near-RT RIC, xApps, rApps

Test Name: TC_SECURE_EXECUTIVE_ENV_PROVISION

Purpose:

To test whether the NFO compares the Application's owned resource state with the defined resource quotas from the Application descriptor.

To test whether the NFO sends an alarm to the SMO if the two resource states are inconsistent.

Procedure and execution steps

Preconditions

There are: Application, O-Cloud, SMO, NFO, Application Descriptor, (or simulated O-Cloud, SMO) on the test environment.

Execution steps

Execute the following steps:

1. The tester utilizes the O-Cloud to change the resource state of Application (e.g. change vCPU size of the Application).
2. The tester uses the NFO to query the parsed resource state from the Application descriptor.
3. The tester checks whether the NFO sends an alarm to the SMO when the NFO receives the parsed resource state from the Application descriptor and finds that the owned resource state and the parsed resource state are inconsistent.

Expected Results

NFO sends an alarm to the SMO when the NFO receives the parsed resource state from the Application descriptor and find that the owned resource state and the parsed resource state are inconsistent.

Expected format of evidence:

Screenshots of logs containing the alarm on the SMO.

19.3 Signature validation during App image onboarding

Requirement Name: Signature validation during App image onboarding

Requirement Reference: REQ-SEC-ALM-PKG-5, REQ-SEC-ALM-PKG-6, clause 5.3.2.1.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-IMG-04, T-AppLCM-01

DUT/s: O-CU, O-DU, Near-RT RIC, xApps, rApps

Test Name: TC_SIGNATURE_VALIDATION_DURING_APP_IMAGE_ONBOARDING

Purpose: The purpose of this test is to validate the digital signature verification mechanism during the onboarding process of application images into the NFO.

Procedure and execution steps

Preconditions

- The Application document describes information regarding digital signature protection of Application images, including details of how the signature check is carried out, who makes the digital signature of Application image etc.
- One Application package included two trusted Application images and the Application package carries a correct digital signature of the Application package.
- Another Application package included untrusted Application image which carry wrong digital signature of Application image and the Application package carries a correct digital signature of the Application package.
- There are a NFO, or a simulated NFO. A certificate or public key which is used to verify the digital signature of Application image has been pre-configured in the NFO. This certificate is trusted by the operator. It means the digital signature of the Application image is successfully verified by using the public key in the certificate trusted by the operator.

Execution steps

Execute the following steps:

1. Review the documentation provided by the vendor describing how digital signature of the Application image is verified.
2. The tester uploads an Application package included two trusted Application images into a NFO. The NFO verifies the Application images by validating each digital signature of the Application image using the pre-configured certificate or the public key according to the documentation.
3. The tester uploads another Application package included un-trusted Application image into NFO. The NFO verifies the Application image(s) by validating each digital signature of the Application image using the pre-configured certificate or the public key according to the documentation.

Expected Results

In the step 2, the signatures of the Application images are successfully validated, and the Application package is successfully on boarded into the NFO;

In the step 3, the signature of the un-trusted Application image is failed to be validated and the Application package is not on boarded into the NFO.

Expected format of evidence:

Snapshots containing the result of the Application package on boarding.

19.4 Application image deployment security

Requirement Name: Application image deployment security

Requirement Reference: REQ-SEC-ALM-PKG-12, clause 5.3.2.1.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-IMG-04, T-AppLCM-02

DUT/s: O-CU, O-DU, Near-RT RIC, xApps, rApps

Test Name: TC_APP_IMAGE_VULNERABILITY_CHECK_ON_DEPLOY

Purpose: The purpose of this test is to verify that an Application image is free from known vulnerabilities.

Procedure and execution steps

Preconditions

O-Cloud with Application image scanning tools integrated.

Execution steps

1. Deploy an Application image known to have vulnerabilities:
    - Select an Application image with known vulnerabilities, such as an image with outdated software or documented security issues.
    - Attempt to deploy the image to an O-Cloud using the appropriate deployment configuration.
    - Monitor the deployment process and capture any error messages or logs.
2. Deploy an Application image with outdated or unapproved software libraries:
    - Create a custom Application image that includes outdated or unapproved software libraries.
    - Attempt to deploy the custom image to an O-Cloud using the appropriate deployment configuration.
    - Monitor the deployment process and capture any error messages or logs.

Expected Results

1. For the first step, the container image with known vulnerabilities is rejected or flagged as insecure, preventing its deployment.
2. For the second step, the container image with outdated or unapproved software is blocked from deployment, ensuring compliance with security policies.

Expected format of evidence:

1. Vulnerability scan reports generated by the Application image scanning tool, indicating the detected vulnerabilities and their severity.
2. Rejection logs or error messages from the Application image registry or O-Cloud, indicating the rejection or blocking of insecure images.

20 Security tests of Common Application Lifecycle Management

20.1 Overview

This clause contains security tests to validate the security protection relevant to Common App LCM.

20.2 Application package

20.2.1 Application package signature verification

Requirement Name: Application package authenticity and integrity protection

Requirement Reference: REQ-SEC-ALM-PKG-2, REQ-SEC-ALM-SU-1, clause 5.3.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: "The Application package shall be signed by the Application Provider prior to its delivery to the Service Provider to ensure its authenticity and integrity."

Threat References: T-IMG-01, T-NEAR-RT-02, T-rAPP-05, T-xApp-02

DUT/s: Non-RT RIC, Near-RT RIC, O-CU, O-DU, O-RU, xApps, rApps

Test Name: TC_App_Signature_Verification

Purpose: To verify the application package authenticity and integrity validation during onboarding and instantiation.

Procedure and execution steps

Preconditions

- The Application package is signed by the Application Provider prior to its delivery to the Service Provider.
- The Application provider certificate (used to sign the Application Package) is issued by a CA trusted by the Service provider, and the root certificate of the CA is pre-installed at SMO.

Execution steps

Upon reception of the Application package from the Application Provider, the Service Provider verifies the Application Provider signature using the execution steps in clause 9.5.2.

Expected Results

Validation of the Application package's Application Provider signature is successful.

Signing of the Application package by the Service Provider is successful.

Validation of Application package signature and the Service Provider signature during instantiation of the application is successful. If verification is unsuccessful, the Service Provider may suspend the application instantiation process.

Expected format of evidence: Log files for each step of the procedure.

20.2.2 Minimum Requirements

Requirement Name: Application package includes minimal artifacts.

Requirement Reference: REQ-SEC-ALM-PKG-3, clause 5.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-09

DUT/s: O-RU, O-DU, O-CU, Near-RT RIC, xApp, rApp

Test Name: TC_App_Pkg_Min_Artifacts

Purpose: The purpose of this test is to verify that an Application package includes minimal artifacts according to REQ-SEC-ALM-PKG-3.

Procedure and execution steps

Preconditions

Application package available for access.

Execution steps:

1. Access the Application package contents.
2. Verify the Application package includes minimally the following artifacts:
    a. Application software image
    b. Signing certificate
    c. Application provider signature(s)

Expected Results:

Application package includes the minimal artifacts required.

Expected format of evidence:

Screenshot(s)

20.2.3 App Package Change Log

Requirement Name: Application package shall have change logs.

Requirement Reference: REQ-SEC-ALM-PKG-15, clause 5.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-O-RAN-09

DUT/s: O-RU, O-DU, O-CU, Near-RT RIC, xApp, rApp

Test Name: TC_App_Pkg_Change_Log

Purpose: The purpose of this test is to verify that the Application packages contains a change log according to REQ-SEC-ALM-PKG-15.

Procedure and execution steps

Preconditions

Application package available for access.

Execution steps:

- Access the Application package and external artifacts if present.
- Verify the Application package or external artifacts includes change log.
- Verify latest version noted in change log matches the current Application version.

Expected Results:

Change log is included in the Application package.

Expected format of evidence:

Screenshot(s)

20.3 Secure Decommissioning

20.3.1 Post-Decommission Report

Requirement Name: A complete post-decommission report shall be generated.

Requirement Reference: REQ-SEC-ALM-DECOM-1, clause 5.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-AppLCM-06

DUT/s: O-RU, O-DU, O-CU, Near-RT RIC, xApp, rApp

Test Name: TC_App_Decomm_Report

Purpose: The purpose of this test is to ensure a decommissioning report is generated for a decommissioned Application.

Procedure and execution steps

Preconditions

Ensure Application subject to decommissioning has no running instance(s) on O-RAN system.

Execution steps:

1) Execute decommissioning of Application

2) Generate report of decommissioning whether through manual or automated means

Expected Results:

Decommissioning report documenting Application decommissioning is generated.

Expected format of evidence:

A report detailing:

- Decommissioned Application name and version
- Date and time of Application decommissioning
- Tasks performed during decommissioning

- Other pertinent details

20.3.2 Trust Artifact Revocation

Requirement Name: Trust artifacts revoked during Application decommissioning.

Requirement Reference: REQ-SEC-ALM-DECOM-3, clause 5.3, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description:

Threat References: T-AppLCM-06

DUT/s: O-RU, O-DU, O-CU, Near-RT RIC, xApp, rApp

Test Name: TC_App_Trust_Artifact_Revocation

Purpose: The purpose of this test is to verify that all trust artifacts associated with an Application are revoked at the time of Application decommissioning.

Procedure and execution steps

Preconditions

- Locate and prepare Application trust artifacts for revocation.
- Ensure Application subject to decommissioning has no running instance(s) on O-RAN system.

Execution steps:

1. Revoke trust artifacts from Application subject to decommissioning.
2. Perform a scan and verify that all trust artifacts associated with Application have been revoked.

   EXAMPLE: If trust artifact is a certificate, verify that certificate is in certification revocation list.
3. Execute decommissioning of Application.
4. Attempt to instantiate Application and verify that it cannot be re-instantiated without the trust artifacts.

Expected Results:

- All trust artifacts are removed from the Application and the decommissioned Application cannot be re-instantiated.

Expected format of evidence:

Screenshot(s), report

21 Security test of O-CU-CP

21.1 Overview

The present clause contains 3GPP security test cases applicable to O-CU-CP and O-RAN specific O-CU-CP test cases.

21.2 O-CU-CP 3GPP specific security functional requirements and test cases

Requirement Name: 3GPP specific O-CU-CP security

Requirement Reference: REQ-SEC-OCU-1, clause 5.1.4.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: "O-CU-CP and O-CU-UP shall meet the security requirements for gNB-CU-CP and gNB-CU-UP respectively", as specified in TS 33.501 [25]

DUT/s: O-CU-CP

Purpose: To verify the O-CU-CP meet the security requirements for gNB-CU-CP

Test Name: TC_O_CU_CP_3GPP_33_523_Cl_5_2_2 (As defined in clause 5.2.2 of TS 33.523 [23])

gNB-CU-CP specific security functional requirements and test cases specified in clause 5.2.2 of TS 33.523 [23] apply to O-CU-CP.

21.3 O-RAN specific security functional requirements and test cases

The TLS test cases in clause 6.3 of the present document apply to the O1 interface of O-CU-CP.

The IPsec test cases in clause 6.4 of the present document apply to the E2 interface of O-CU-CP.

22 Security test of O-CU-UP

22.1 Overview

The present clause contains 3GPP security test cases applicable to O-CU-UP and O-RAN specific O-CU-UP test cases.

22.2 O-CU-UP 3GPP specific security functional requirements and test cases

Requirement Name: 3GPP specific O-CU-UP security

Requirement Reference: REQ-SEC-OCU-1, clause 5.1.4.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: "O-CU-CP and O-CU-UP shall meet the security requirements for gNB-CU-CP and gNB-CU-UP respectively", as specified in TS 33.501 [25]

DUT/s: O-CU-UP

Test Name: TC_O_CU_UP_3GPP_33_523_Cl_6_2_2 (As defined in clause 6.2.2 of TS 33.523 [23])

Purpose: To verify the O-CU-CP meet the security requirements for gNB-CU-UP

gNB-CU-UP specific security functional requirements and test cases specified in clause 6.2.2 of TS 33.523 [23] apply to O-CU-UP.

22.3 O-RAN specific security functional requirements and test cases

The TLS test cases in clause 6.3 of the present document apply to the O1 interface of O-CU-UP.

The IPsec test cases in clause 6.4 of the present document apply to the E2 interface of O-CU-UP.

23 Security test of O-DU

23.1 Overview

The present clause contains 3GPP security test cases applicable to O-DU and O-RAN specific O-DU test cases.

23.2 O-DU 3GPP specific security functional requirements and test cases

Requirement Name: 3GPP specific O-DU security

Requirement Reference: REQ-SEC-ODU-1, clause 5.1.5.1, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: “O-DU shall meet the security requirements for gNB-DU” as specified in TS 33.501 [25].

DUT/s: O-DU

Test Name: TC_O_DU_3GPP_33_523_Cl_7_2_2 (As defined in clause 7.2.2 of TS 33.523 [23])

gNB-DU specific security functional requirements and test cases specified in clause 7.2.2 of TS 33.523 [23] apply to O-DU.

23.3 O-RAN specific security functional requirements and test cases

The 802.1X Authenticator Validation test cases in clause 11.2.1 applies to O-DU for the network configuration where O-DU acts as an 802.1X authenticator.

The 802.1X Supplicant Validation test cases in clause 11.2.2 apply to O-DU.

The TLS test cases in clause 6.3 of the present document apply to the O1 interface and M-Plane of O-DU.

The IPsec test cases in clause 6.4 of the present document apply to the E2 interface of O-DU.

The SSH Server & Client test cases in clause 6.2 of the present document apply to the M-Plane of O-DU.

24 End-to-End security test cases

24.0 Overview

This clause describes E2E tests evaluating and assessing the security aspects of an O-RAN conformant radio access network.

The O-RU, O-DU, O-CU-CP and O-CU-UP as defined in O-RAN Architecture Description [1] is the System under Test (SUT) and can be viewed as an integrated black box in the context of the E2E security testing.

24.1 3GPP Security Assurance Specification (SCAS)

For NR technology, Table 24-1 applies. The test cases referenced in this table are from 3GPP TS 33.511 [8], which are applied to the O-RAN system. The table also indicates the applicable technology, specifying whether each test case pertains to NR NSA (Non-Standalone) and/or NR SA (Standalone).

For LTE technology, Table 24-2 applies. The test cases referenced in this table are from 3GPP TS 33.216 [9], which are applied to the O-RAN system.

The tables also contain the information relative to the 3GPP releases affected for each test case.

Table 24-1: List of SCAS Test Cases for NR and applicable technology from Clause 4.2.2 of 3GPP TS 33.511

Test Case (O-RAN Ref. #)	Test Case (3GPP clause number and title)	Applicable Technology	Applicable 3GPP Releases
TC_SCAS_NR_E2E_24.1.1	4.2.2.1.1 Integrity protection of RRC-signalling	NR NSA (Options 3 and 4) NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.2	4.2.2.1.2 Integrity protection of user data	NR NSA (Options 4 and 7) NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.3	4.2.2.1.4 RRC integrity check failure	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.4	4.2.2.1.5 UP integrity check failure	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.5	4.2.2.1.6 Ciphering of RRC-signalling	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.6	4.2.2.1.7 Ciphering of user data	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.7	4.2.2.1.8 Replay protection of user data	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.8	4.2.2.1.9 Replay protection of RRC-signalling	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.9	4.2.2.1.10 Ciphering of user data based on the security policy sent by the SMF	NR NSA (Options 4 and 7) NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.10	4.2.2.1.11 Integrity of user data based on the security policy sent by the SMF	NR NSA (Options 4 and 7) NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.11	4.2.2.1.12 AS algorithms selection	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.12	4.2.2.1.13 Key refresh	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.13	4.2.2.1.14 Bidding down prevention in Xn-handovers	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.14	4.2.2.1.15 AS protection algorithm selection	NR NSA NR SA	16 17 18

TC_SCAS_NR_E2E_24.1.15	4.2.2.1.16 Control plane data confidentiality protection over N2/Xn interface	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.16	4.2.2.1.17 Control plane data integrity protection over S1/NG/Xn interface	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.17	4.2.2.1.18 Key update on dual connectivity	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.18	4.2.2.1.19 User plane security activation in Inactive scenario	NR NSA NR SA	16 17 18
TC_SCAS_NR_E2E_24.1.19	4.2.2.1.20 User plane data confidentiality protection over N3/Xn interface	NR NSA NR SA	18
TC_SCAS_NR_E2E_24.1.20	4.2.2.1.21 User plane data integrity protection over N3/Xn interface	NR NSA NR SA	18

Table 24-2: List of SCAS Test Cases for LTE and applicable technology from Clause 4.2.2 of 3GPP TS 33.216

Test Case (O-RAN Ref. #)	Test Case (3GPP clause number and title)	Applicable Technology	Applicable 3GPP Releases
TC_SCAS_LTE_E2E_24.1.1	4.2.2.1.1 Control plane data confidentiality protection over S1/X2	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.2	4.2.2.1.2 Control plane data integrity protection over S1/X2	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.3	4.2.2.1.3 User plane data ciphering	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.4	4.2.2.1.4 User plane data integrity protection	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.5	4.2.2.1.5 AS algorithms selection	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.6	4.2.2.1.6 RRC integrity protection	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.7	4.2.2.1.7 Selection of EIA0	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.8	4.2.2.1.8 (1) Key refresh (PDCP Count)	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.9	4.2.2.1.8 (2) Key refresh (DRB ID)	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.10	4.2.2.1.9 AS integrity algorithm selection	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.11	4.2.2.1.10 Bidding down prevention in X2-handovers	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.12	4.2.2.1.11 AS protection algorithm selection	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.13	4.2.2.1.12 RRC and UP downlink ciphering	LTE	16 17 18

TC_SCAS_LTE_E2E_24.1.14	4.2.2.1.13 Map a UE NR security capability	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.15	4.2.2.1.14 UE NR security capability	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.16	4.2.2.1.15 Bidding down prevention in X2-handovers	LTE	16 17 18
TC_SCAS_LTE_E2E_24.1.17	4.2.2.1.16 Integrity protection of user data	LTE	18
TC_SCAS_LTE_E2E_24.1.18	4.2.2.1.17 Select the right UP integrity protection policy	LTE	18
TC_SCAS_LTE_E2E_24.1.19	4.2.2.1.18 Select the right UP IP policy	LTE	18
TC_SCAS_LTE_E2E_24.1.20	4.2.2.1.19 Select the right UP IP policy in S1 handover	LTE	18
TC_SCAS_LTE_E2E_24.1.21	4.2.2.1.20 Bidding down prevention for UP IP Policy	LTE	18

24.2 DoS, fuzzing and blind exploitation test

Due to the open and disaggregated nature of the O-RAN system (SUT), the attack surfaces associated with some of its critical transport protocols and major interfaces of the O-RAN system become easy targets for potential attackers. Cyberattacks like DoS, fuzzing and blind exploitation types are easy to launch, require little information on the target system, and could cause significant performance degradation, or even the service interruption if not properly mitigated.

The duration of test TRAFFIC GENERATION specified in this clause shall be at minimum 3 minutes.

The **Table 24-3** summarizes the test cases and the applicable technology.

Table 24-3: End-to-end test cases and applicable technology

Test case		Applicable technology		
		LTE	NSA	SA
Test ID	Name			
24.2.1.1	TC_E2E_ODU_SPlane_DoS	N/A	Y	Y
24.2.1.2	TC_E2E_ODU_SPlane_Robustness	N/A	Y	Y
24.2.2.1	TC_E2E_ODU_CPlane_eCPRI_DoS	N/A	Y	Y
24.2.2.2	TC_E2E_ODU_CPlane_eCPRI_Robustness	N/A	Y	Y
24.2.2.3	TC_E2E_ORU_CPlane_eCPRI_DoS	N/A	Y	Y
24.2.2.4	TC_E2E_ORU_CPlane_eCPRI_Robustness	N/A	Y	Y
24.2.3.1	TC_E2E_NearRTRIC_A1_DoS	N/A	Y	Y
24.2.3.2	TC_E2E_NearRTRIC_A1_Robustness	N/A	Y	Y
24.2.3.3	TC_E2E_NearRTRIC_A1_Vulnerabilities	N/A	Y	Y
24.2.4.1	TC_E2E_OCloud_SideChannel_DoS	N/A	Y	Y

24.2.1 S-Plane

24.2.1.1 S-Plane PTP DoS Attack

Requirement Name: O-DU S-Plane DoS Attack

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5.1, O-RAN Security Requirements and Control Specification [5]

Requirement Description: “An O-RAN component with external network interface shall be able to withstand network transport protocol based volumetric DDoS attack without system crash and returning to service level after the attack.”.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ODU_SPlane_DoS

Purpose: To verify that a predefined volumetric DoS attack against O-DU S-Plane will not crash the SUT, returning to service level after the attack.

Procedure and execution steps

Preconditions

- The tester requires easy to access MAC address information of the O-DU’s open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

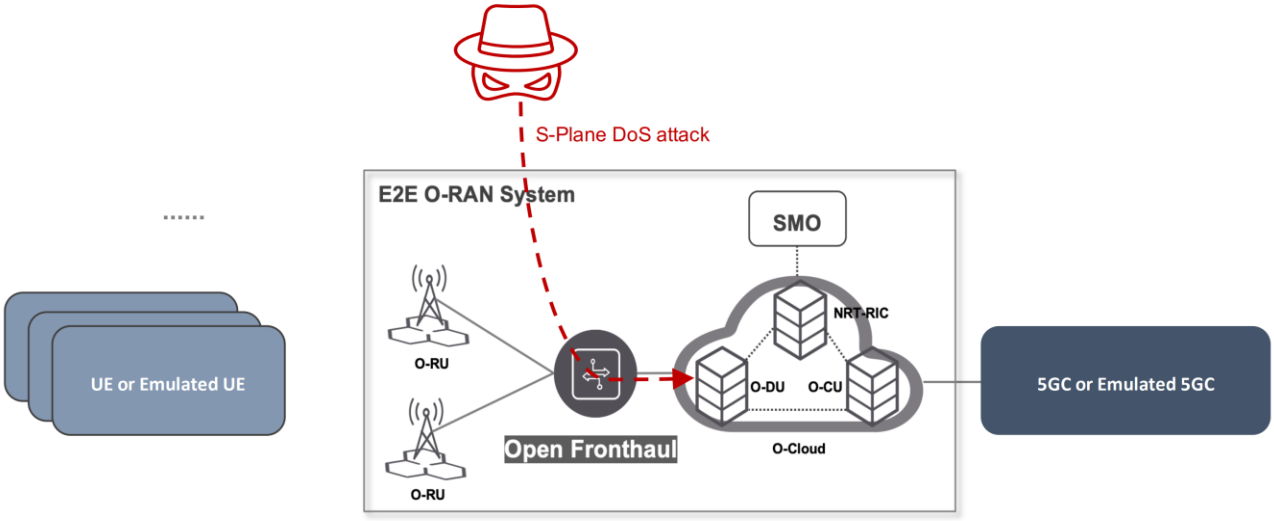

Figure 24-1: S-Plane O-DU Test setup

Execution steps

The tester uses a test tool to generate different types of volumetric DoS attack against the MAC address of the O-DU S-Plane

1. Volumetric tiers: 10Mbps, 100Mbps, 1Gbps
2. DoS Traffic random mixed of: generic Ethernet frames, PTP announce/sync message
3. DoS source address: spoofed MAC of T-GM/T-BC or T-TC (depending on the setup), random source MACs

Expected results

1. During the test, the 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

24.2.1.2 S-Plane PTP Unexpected Input

Requirement Name: O-DU S-Plane Robustness

Requirement Reference: REQ-SEC-OFSP-4, clause 5.2.5.3.2, O-RAN Security Requirements and Control Specification [5]

Requirement Description: The O-DU is able to detect and defend against application level attacks across the S-Plane interface, due to misbehaviour or malicious intent.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ODU_SPlane_Robustness

Purpose: To verify that an unexpected (not in-line with protocol specification) input sent towards O-DU S-Plane will not compromise the security of the SUT.

Procedure and execution steps

Preconditions

- The test requires easy to access MAC address information of the O-DU's open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

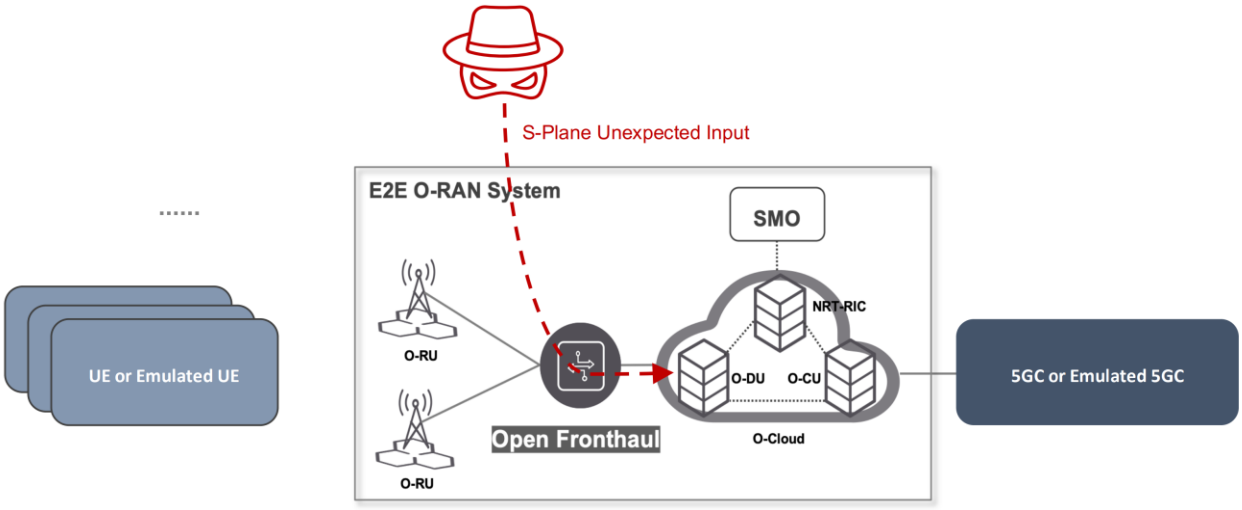

Figure 24-2: S-Plane PTP Unexpected Input Test Setup

Execution steps

1. The tester uses a packet capture tool to capture sample of legitimate PTP message sent towards the O-DU S-Plane
2. The tester uses fuzzing tool to replay the captured PTP message while mutating its content and keeping original source/destination MAC address. Send at least 250,000 iterations of mutated PTP message based on a random seed

Expected results

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

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Expected format of evidence: Log files, traffic captures and/or reports

24.2.2 C-Plane

24.2.2.1 C-Plane eCPRI DoS Attack

Requirement Name: O-DU C-Plane eCPRI DoS Attack

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5.1, O-RAN Security Requirements and Control Specification [5]

Requirement Description: "An O-RAN component with external network interface shall be able to withstand network transport protocol based volumetric DDoS attack without system crash and returning to service level after the attack.".

Threat References: T-CPLANE-O2, T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ODU_CPlane_eCPRI_DoS

Purpose: To verify that a predefined volumetric DoS attack against O-DU C-Plane will not crash the SUT, returning to service level after the attack.

Procedure and execution steps

Preconditions

- The test requires easy to access MAC address information of the O-DU's open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

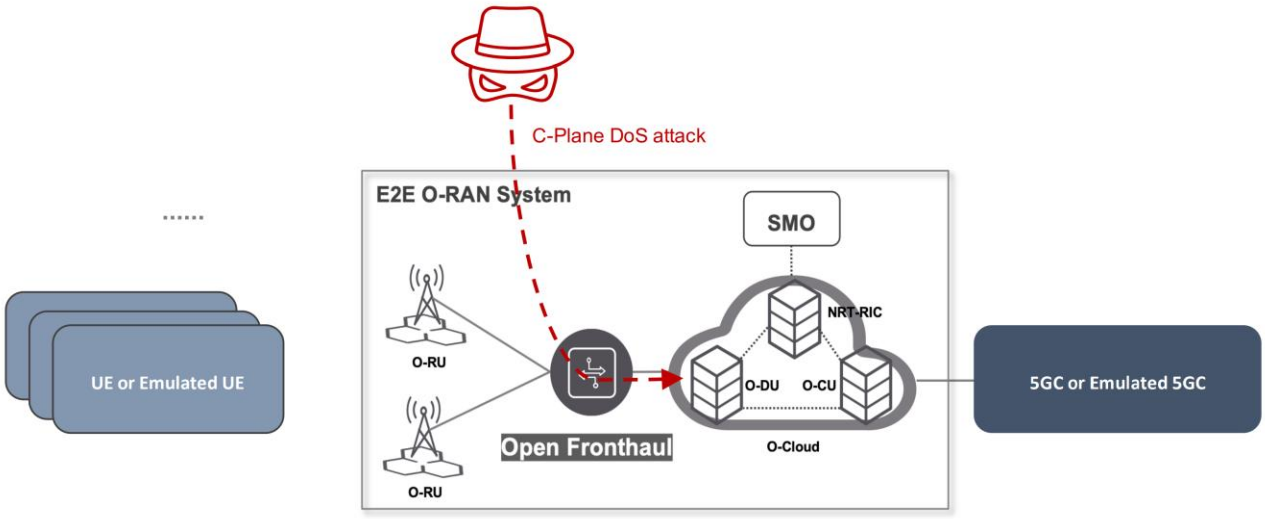

Figure 24-3: C-Plane eCPRI DoS Attack Test Setup

Execution steps

Use test tool to generate several types of volumetric DoS attack against the MAC address of the O-DU C-Plane

1. Volumetric tiers: 10Mbps, 100Mbps, 1Gbps
2. DoS Traffic types: eCPRI real-time Control data message over Ethernet. The C-Plane message types that are made to flow towards O-DU are i) LAA LBT Status and response messages, ii) Ack/Nack Messages, and iii) Wake-up Ready indication Messages. Refer to Figure 4.2-1 Lower layer fronthaul data flows in [26]
3. DoS source address random mixed of: spoofed MAC of O-RU(s), random source MACs

Expected results

1. During the test, the 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

24.2.2.2 C-Plane eCPRI Unexpected Input

Requirement Name: O-DU C-Plane Robustness

Requirement Reference: REQ-SEC-OFCP-2, clause 5.2.5.1.2, O-RAN Security Requirements and Control Specification [5]

Requirement Description: The O-DU is able to detect and defend against application level attacks across the C-Plane messages with O-RUs, due to misbehaviour or malicious intent.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ODU_CPlane_eCPRI_Robustness

Purpose: To verify that an unexpected (not in-line with protocol specification) input sent towards O-DU C-Plane will not compromise the security of the SUT.

Procedure and execution steps

Preconditions

- The test requires easy to access MAC address information of the O-DU's open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

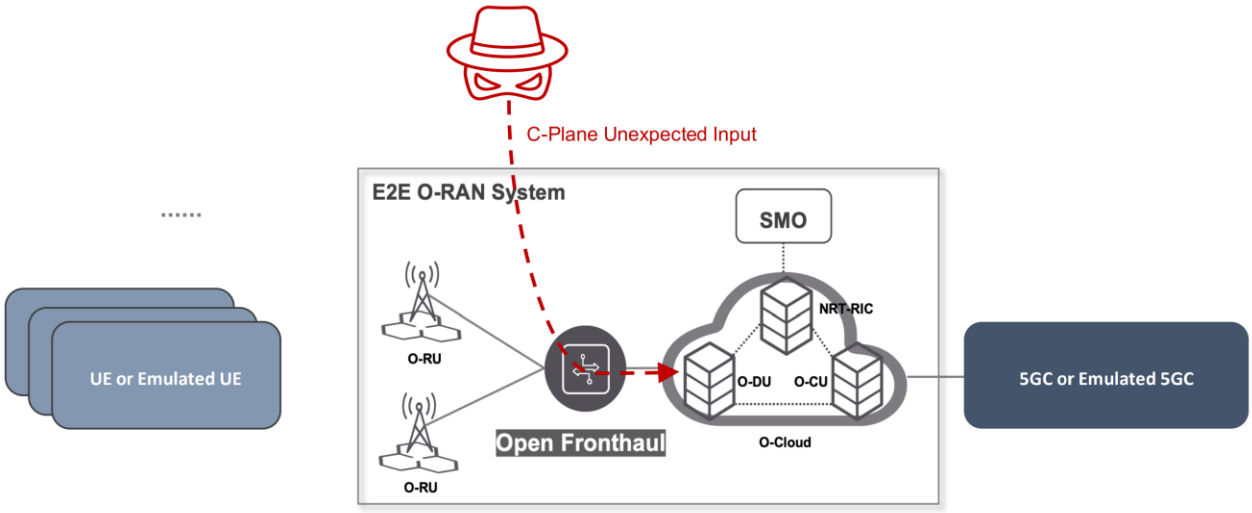

Figure 24-4: C-Plane eCPRI Unexpected Input Test Setup

Execution steps

1. The tester uses a packet capture tool to capture sample of legitimate eCPRI message sent towards the O-DU C-Plane
2. The tester uses a fuzzing tool to replay the captured eCPRI message while mutating its content (message type and/or payload) and keeping original source/destination MAC address. Send at least 250,000 iterations of mutated eCPRI message based on a random seed

Expected results

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

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

24.2.2.3 C-Plane eCPRI DoS Attack on O-RU

Requirement Name: O-RU C-Plane eCPRI DoS Attack

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5.1, O-RAN Security Requirements and Control Specification [5]

Requirement Description: “An O-RAN component with external network interface shall be able to withstand network transport protocol based volumetric DDoS attack without system crash and returning to service level after the attack.”.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ORU_CPlane_eCPRI_DoS

Procedure and execution steps

Purpose: To verify that a predefined volumetric DoS attack against O-RU C-Plane will not crash the SUT, returning to service level after the attack.

Preconditions

The test requires easy to access MAC address information of the O-RU's open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN TIFG End-to-End Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

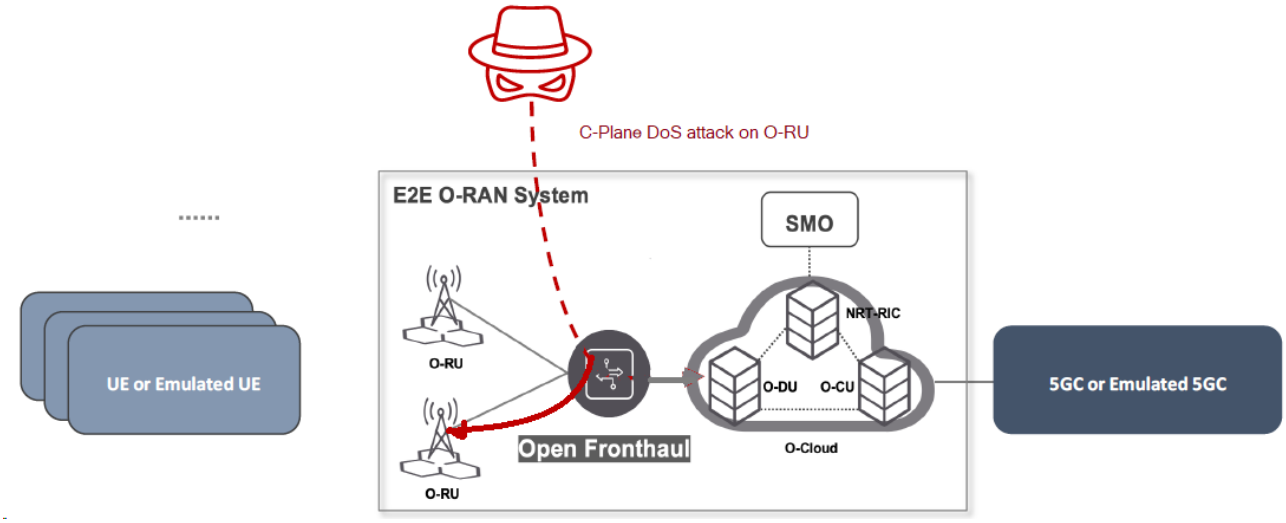

Figure 24-5: C-Plane eCPRI DoS Attack on O-RU Test Setup

Execution steps

Use test tool to generate several types of volumetric DoS attack against the MAC address of the O-RU C-Plane

1. Volumetric tiers: 10Mbps, 100Mbps, 1Gbps
2. DoS Traffic types: eCPRI real-time ctrl data message over Ethernet. The valid C-Plane message types that are made to flow towards O-DU [Reference Figure 4.3-1Lower layer front haul data flows [26]] are i) Scheduling commands (DL & UL) & Beamforming commands, ii) LAA LBT configuration commands and requests iii) UE Channel information
3. DoS source address, a mixed of: spoofed MAC of O-DU(s), random source MACs

Expected results

- During the test, the SUT maintains an operational level
- 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

24.2.2.4 C-Plane eCPRI Unexpected Input on O-RU

Requirement Name: O-RU C-Plane Robustness

Requirement Reference: REQ-SEC-OFCP-2, clause 5.2.5.1.2, O-RAN Security Requirements and Control Specification [5]

Requirement Description: The O-RU is able to detect and defend against application level attacks across the C-Plane messages with O-DUs, due to misbehaviour or malicious intent.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_ORU_CPlane_eCPRI_Robustness

Purpose: To verify that an unexpected (not in-line with protocol specification) input sent towards O-RU C-Plane will not compromise the security of the SUT.

Procedure and execution steps

Preconditions

- The test requires easy to access MAC address information of the O-RU's open fronthaul interface and L2 connectivity (e.g. over L2 network switching device) 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the DUT.

Refer to the diagram below for the test setup and configuration:

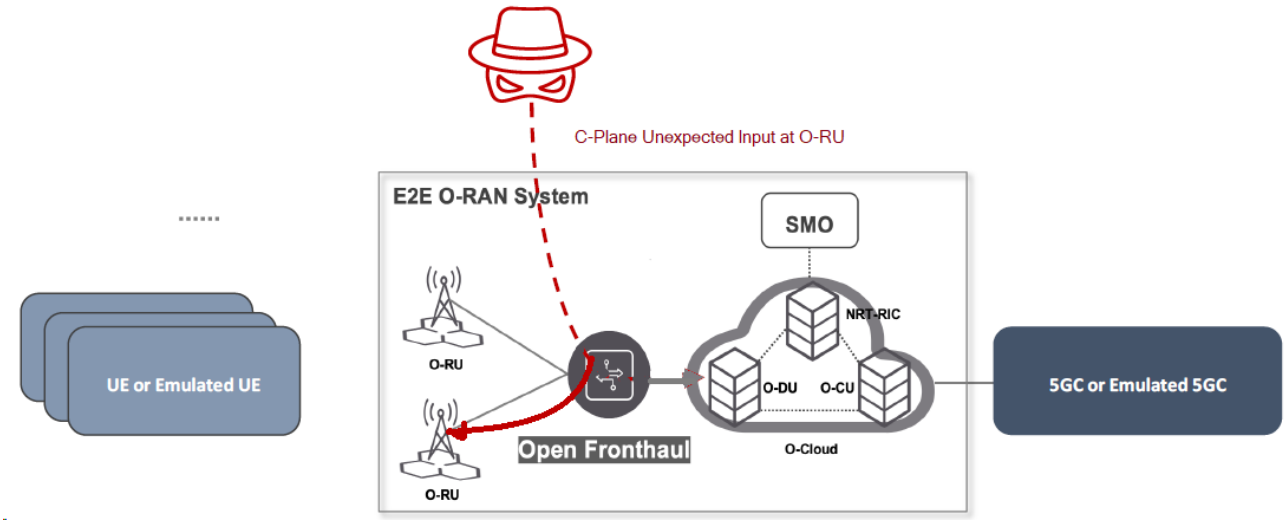

Figure 24-6: C-Plane eCPRI Unexpected Input on O-RU Test Setup

Execution steps

1. The tester uses a packet capture tool to capture sample of legitimate eCPRI message sent towards the O-RU C-Plane.
2. The tester uses a fuzzing tool to replay the captured eCPRI message while mutating its content (message type and/or payload) and keeping original source/destination MAC address. Send at least 250,000 iterations of mutated eCPRI message based on a random seed.

Expected results

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

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the DUT.

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

24.2.3 A1 interface

24.2.3.1 Near-RT RIC A1 Interface DoS Attack

Requirement Name: Near-RT RIC A1 interface DoS recover

Requirement Reference: REQ-SEC-NEAR-RT-6, clause 5.1.3.1, O-RAN Security Requirements and Control Specification [5]

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.”

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_NearRTRIC_A1_DoS

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 easy to access IP address information of the Near-RT RIC’s A1 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

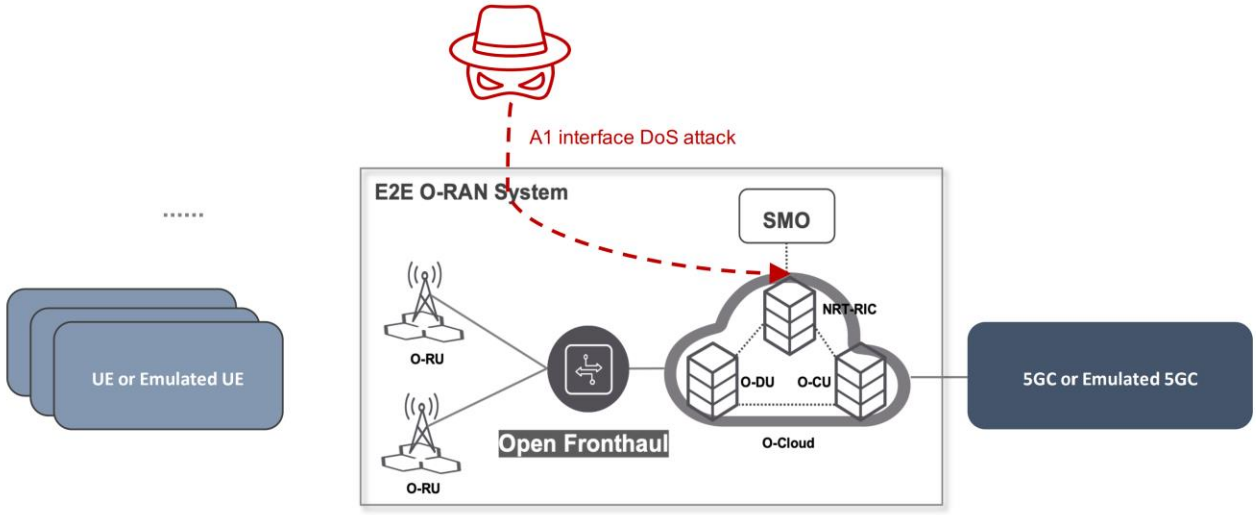

Figure 24-7: Near-RT RIC A1 Interface DoS Attack Test Setup

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 A1 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 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

24.2.3.2 Near-RT RIC A1 Interface Unexpected Input

Requirement Name: Near-RT RIC A1 Robustness

Requirement Reference: REQ-SEC-NEAR-RT-7, clause 5.1.3.1, O-RAN Security Requirements and Control Specification [5]

Requirement Description: “The Near-RT RIC shall be able to detect and defend against content-related attacks across the A1 interface, due to misbehaviour or malicious intent.”

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_NearRTRIC_A1_Robustness

Purpose: To verify that an unexpected (not in-line with protocol specification) input sent towards Near-RT RIC A1 interface will not compromise the security of the SUT.

Procedure and execution steps

Preconditions

- The test requires easy to access IP address information of the Near-RT RIC's A1 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.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

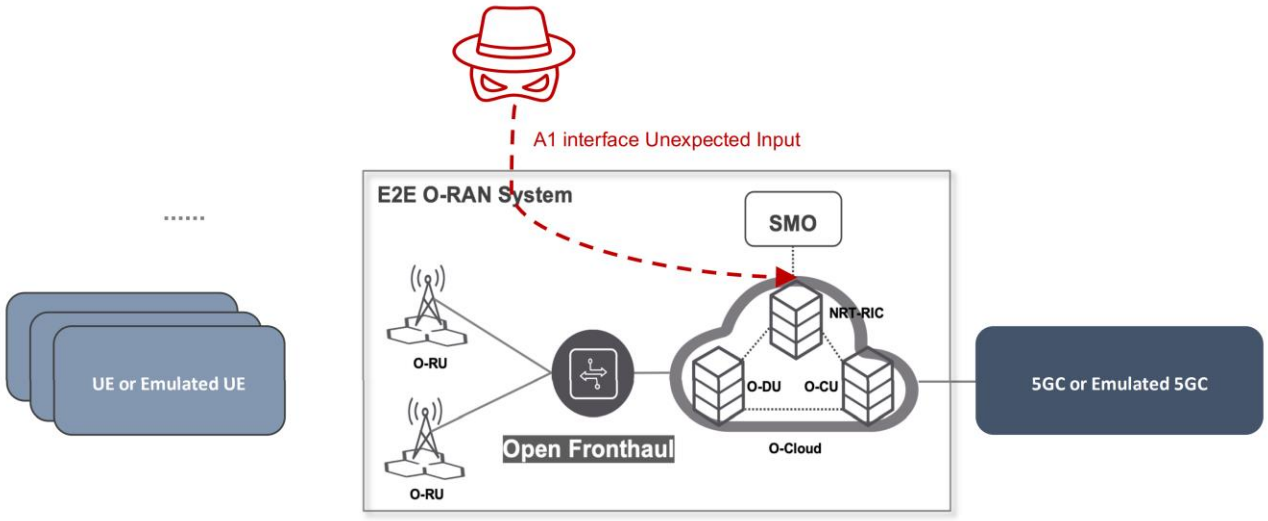

Figure 24-8: Near-RT RIC A1 Interface Unexpected Input Test Setup

Execution steps

1. The tester uses a packet capture tool to capture sample of legitimate HTTP/HTTPs REST API message sent towards the Near-RT RIC A1 interface
2. The tester uses a fuzzing tool to replay the captured HTTP/HTTPs REST API message while mutating its content and keeping original source/destination IP/port. Send at least 250,000 iterations of mutated HTTP/HTTPs REST API message based on a random seed

Expected results

1. During the execution of the test, the degradation of service availability and performance of the SUT is not noticeable.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

24.2.3.3 Near-RT RIC A1 Vulnerability Assessment

Requirement Name: Near-RT RIC A1 Vulnerability Assessment

Requirement Reference: REQ-SEC-SYS-1, clause 5.3.6, O-RAN Security Requirements and Control Specification [5]

Requirement Description: "Known vulnerabilities in the OS and applications of an O-RAN component shall be clearly identified".

Threat References: T-OPENSRC-01, T-OPENSRC-02

SUT/s: O-RAN system

Test Name: TC_E2E_NearRTRIC_A1_Vulnerabilities

Purpose: To verify that exploitation attempts of well-known vulnerabilities executed blindly against Near-RT RIC A1 interface will not compromise security of the SUT

Procedure and execution steps

Preconditions

- The test requires easy to access IP address information of the Near-RT RIC's A1 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.
- The test requires the vulnerability scanning tool has up-to-date database of well-known vulnerabilities (signatures/plugins) based on Common Vulnerabilities and Exposures (CVE). Document the actual version of vulnerability database (signatures/plugins) for further reference.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

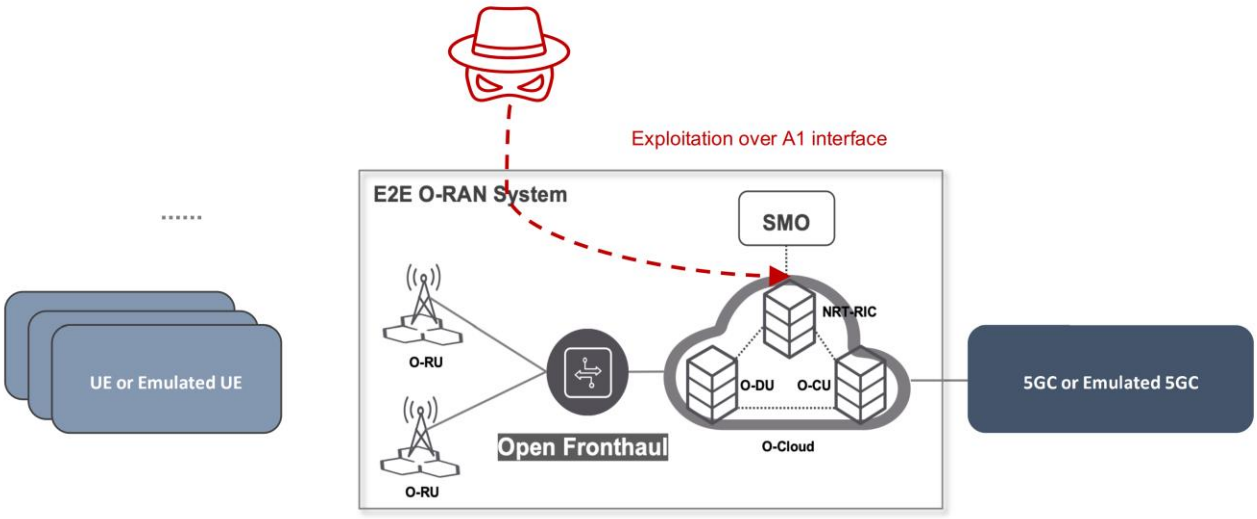

Figure 24-9: Near-RT RIC A1 Vulnerability Assessment Test Setup

Execution steps

The tester uses a vulnerability scanning tool to execute a scan against the IP address of the Near-RT RIC A1 interface. The scan should have the following parameters defined:

- TCP Ports: Port scanner scans all TCP ports in range 0-65535 on the IP interface of SUT. TCP SYN/ACK response by SUT are interpreted as open port.
- UDP Ports: All UDP ports documented in vendor-provided list. Other UDP ports may be considered as open for the purpose of service detection.
- Safe Checks: Disabled (to make sure that exploitation attempts of the vulnerabilities will be performed)

  NOTE: Due to the nature of UDP protocol, there is no simple method of open port detection similar to TCP/SCTP methods based on analysis of response message type (TCP: SYN/ACK, SCTP: INIT-ACK). In case of UDP, open port detection inevitably relies on service detection which is discussed in step 2 of this test procedure. In practice, port scans of entire UDP port range 0-65535 are impractical and time consuming. Typically, service detection is performed only for subset of UDP ports. UDP port subset selection is arbitrary and not standardized. Service detection in this test procedure is required for UDP ports from vendor-provided list and is optional for other UDP ports.

Expected results

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

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Expected format of evidence: Tool testing report and traffic captures

24.2.4 O-Cloud

24.2.4.1 O-Cloud side-channel DoS attack

Requirement Name: O-Cloud DoS Attack

Requirement Reference: REQ-SEC-DOS-1, clause 5.3.5.1, O-RAN Security Requirements and Control Specification [5]

Requirement Description: “An O-RAN component with external network interface shall be able to withstand network transport protocol based volumetric DDoS attack without system crash and returning to service level after the attack.”.

Threat References: T-O-RAN-09

SUT/s: O-RAN system

Test Name: TC_E2E_OCloud_SideChannel_DoS

Purpose: To verify that a noisy neighbour DoS attack against O-Cloud for resource starvation will not crash the SUT, returning to service level after the attack.

Procedure and execution steps

Preconditions

- The test requires access to the O-Cloud platform hosting the network slice(s) of the O-RAN system.
- 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. Network monitoring tools may be used.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from O-RAN E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

Refer to the diagram below for the test setup and configuration:

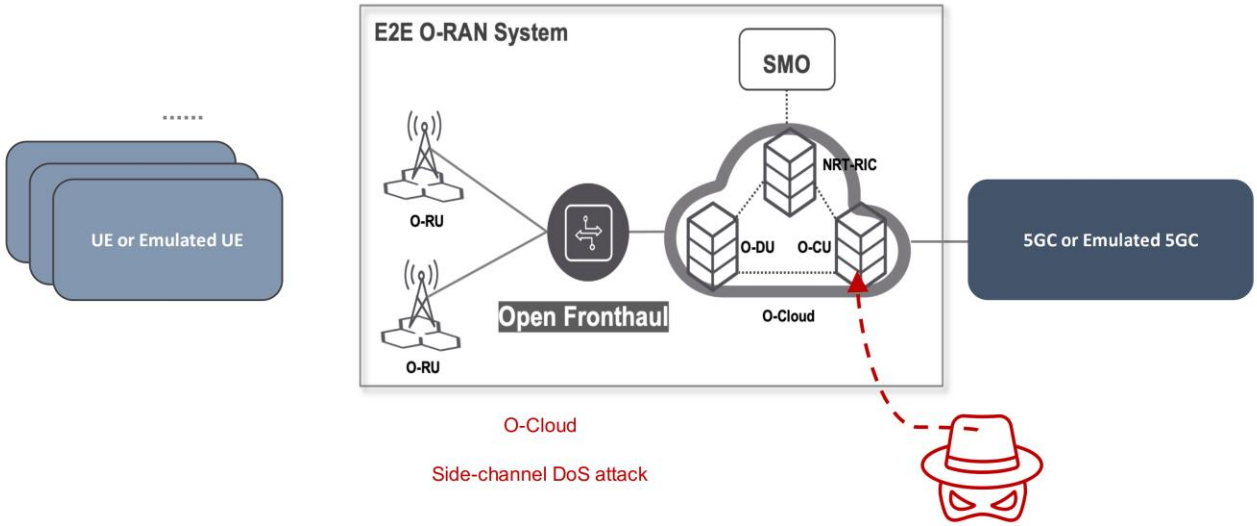

Figure 24-10: O-Cloud side-channel DoS attack Test Setup

Noisy neighbour VNF(s) can be deployed into an existing slice or a new slice of the shared resources with the existing slice under test.

Execution steps

1. The tester uses test tool (through O-Cloud MANO) to instantiate noisy neighbour VNFs.
    a. Noisy neighbour tenant: existing slice or a new slice of the shared resources with the existing slice.
    b. The Noisy Neighbour tenants utilizes these shared resources including CPU, memory, storage, and network. The Noisy Neighbour tenants exhaust all the remaining shared resources. The duration of the test is at least 3 minutes or long enough to cover the benchmarking tests.
2. Run the benchmark test again with the noisy neighbours.
3. Check the logs and alerts that are associated to the test at step 2 and 3.

Expected results

- After the execution of the test, the degradation of service availability and performance of the SUT is not noticeable.
- The Noisy Neighbour attack is properly logged and alerted by the O-Cloud.

RECOMMENDATION: Use clause 5.6 Bidirectional throughput in different radio conditions and clause 6.1 Data Services tests from TIFG E2E Test Specifications [4] as a benchmark for indicating correct behaviour of the SUT.

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

25 Security test of Shared O-RU

25.1 Overview

This clause contains security tests to validate security controls related to the Shared O-RU and the Shared O-RU architecture.

25.2 Shared O-RU test cases

25.2.1 mTLS for mutual authentication

Requirement Name: SEC-CTL-SharedORU-1

Requirement Reference: Clause 5.1.9.2, Security Controls, Shared O-RU, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: mTLS support on Shared O-RU

Threat References: T-SharedORU-06

DUT/s: Shared O-RU

Test Name: TC_SharedORU_mTLS

Purpose: To verify the Shared O-RU is able to mutually authenticate with an O-RU Controller using mTLS, with PKI-based X.509 certificates.

Procedure and execution steps

Preconditions

DUT shall be the Shared O-RU with mTLS 1.2, or 1.3, support enabled.

Execution steps

This test case follows the execution steps for mTLS specified in mTLS Execution steps, clause 6.3.

Expected results

The Shared O-RU supports mutual authentication with an O-RU Controller using mTLS.

Expected format of evidence: Log entries and packet captures.

25.2.2 NACM Authorization

Requirement Name: Open Fronthaul Interface security requirements

Requirement Reference: Clause 5.1.9.2, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: NACM support for Shared O-RU

Threat References: T-SharedORU-22, T-SharedORU-23

DUT/s: Shared O-RU

Test Name: TC_SharedORU_NACM_Authorization

Purpose: To verify the Shared O-RU through is able to enforce role-based least privilege access control on the Open Fronthaul by using NACM [14].

Procedure and execution steps

Preconditions

DUT shall be the Shared O-RU with:

- IP enabled Open Fronthaul M-Plane interface, reachable from the authentication server;
- Valid certificate loaded for the server and necessary certificate authorities (CAs)
- Client's root CA required to validate NETCONF client certificate
- Valid TLS Client-to-NETCONF username mapping

Execution steps

First set up a host/device with TLS client software installed, valid client certificates, keys, root CA certificate for the server (Shared O-RU), and all intermediate CA certificates required to validate the client certificate.

The following test steps shall be validated:

5. Start the NETCONF-over-TLS session using OpenSSL s_client command to connect with DUT using TLSv1.2 or TLSv1.3
6. Verify the session is established and mapped to the correct NETCONF user
7. Verify the global NACM enforcement control setting of
    a. enable-nacm = true
    b. read-default = permit
    c. write-default = deny
    d. exec-default = deny
    e. enable-external-groups = true
8. Verify the NACM rule sets for the pre-defined groups
9. Close the NETCONF session and TLS connection

Upon availability of the NETCONF operations set(s) definition per NACM group, the NACM rule set(s) enforcement by the DUT shall be validated for each of those pre-defined groups listed above.

Expected results

The Shared O-RU shall support the NETCONF over TLS session over its Open Fronthaul M-Plane interface and NACM enforcement control settings.

Expected format of evidence: Log entries and packet captures.

25.2.3 TLS across Open Fronthaul

Requirement Name: SEC-CTL-SharedORU-4

Requirement Reference: Clause 5.1.9.2, Security Controls, Shared O-RU, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: TLS on Shared O-RU

Threat References: T-SharedORU-27, T-SharedORU-28

DUT/s: Shared O-RU

Test Name: TC_SharedORU_TLS

Purpose: To verify the Shared O-RU is able to establish a TLS session with an O-RU Controller and provide confidentiality and integrity protection for messages exchanged with the O-RU Controller.

Procedure and execution steps

Preconditions

DUT shall be the Shared O-RU with TLS support enabled.

Execution steps

This test case shall follow the execution steps for TLS specified in TLS Execution steps, clause 6.3.

Expected results

Shared O-RU shall provide confidentiality and integrity protection for data in transit on the Open Fronthaul M-Plane interface.

Expected format of evidence: Log entries and packet captures.

25.2.4 Reject Password-based authentication

Requirement Name: SEC-CTL-SharedORU-2

Requirement Reference: Clause 5.1.9.2, Security Controls, Shared O-RU, O-RAN Security Requirements and Controls Specifications [5]

Requirement Description: Shared O-RU is able to reject password-based authentication on the Open Fronthaul M-Plane

Threat References: T-SharedORU-04

DUT/s: Shared O-RU

Test Name: TC_SharedORU_SSH_Password_Based_Authentication

Purpose: To verify the Shared O-RU can reject a SSH session using password-based authentication on the Open Fronthaul.

Procedure and execution steps

Preconditions

DUT shall be the Shared O-RU with password-based authentication for SSH on the Open Fronthaul disabled.

Execution steps

1. Enable SSH on the Open Fronthaul for the Shared O-RU. Ensure password-based authentication is not enabled on the Shared O-RU for SSH on the Open Fronthaul.

2. Configure the O-RU Controller as the SSH client with password-based authentication on the Open Fronthaul M-Plane.

3. Attempt to establish the Open Fronthaul M-Plane session between the Shared O-RU and O-RU Controller.

Expected results

The Shared O-RU rejects the Open Fronthaul M-Plane session with the O-RU Controller.

Expected format of evidence: Log entries, packet captures, and/or screenshots.

Annex A (informative): Example of Security Testing Tools / Toolset

Table Annex A-1: List of sample open source security testing tools/toolset

Testing Tool	Example(s)
DTLS scanning tool	open source "pySSLScan": https://github.com/DinoTools/pysslscan
IPsec IKE scanning tool	open source "ike-scan": https://github.com/royhills/ike-scan
Port scanner	open source "Nmap": https://nmap.org/
SSH audit tool	open source "ssh-audit": https://github.com/jtesta/ssh-audit
TLS scanning tool	open source "sslyze": https://github.com/nabla-c0d3/sslyze
Software image signing tool	open source "Sigstore": https://github.com/sigstore

Annex B (informative): Template of test report

A. GENERAL INFORMATION

A1 Name of test campaign	
A2 Version of the report – reference ID	A3 Date(s) of testing
A4 Contact person (tester) – incl. Name, Organization, E-mail address	
A4 Test location (lab) – incl. the address	
A5 Description of test campaign, summary of test results, conclusions	

List of tests - details of each test can be found in Section E.

Test No.	Test name	Test status [PASS / FAIL / -]
01		
02		
03		

B. TEST AND MEASUREMENT EQUIPMENT AND TOOLS

#	Equipment or tool	Type	Manufacture	Version (HW/SW)	Notes*
01					
02					
03					

* Specific details such as the sub-module version (such as vulnerability database version)

C. SYSTEM UNDER TEST

C1 Total number of DUTs included in SUT	C2 Deployment architecture
C3 Description of SUT – connection/block diagram	

DUT 1*

C3 Type	C4 Serial Number	C5 Supplier (manufacture)
C6 SW version	C7 HW version (if applicable)	
C8 Interface/IOT profile(s) if applied		
C9 Description incl. parameters, setting/configuration		

* If SUT contains more DUTs, please copy the table

D. TEST CONFIGURATION

D1 Function(s) and Service(s) setting	D2 Network setting

E. TEST RESULTS

E1 Test No.	E2 Test name
E3 Date(s) of test execution	E4 Reference to test specification
E5 Utilized test and measurement equipment and tools, incl. the specific setting/configuration – reference to Section B	
E6 Test setup – connection/block diagram – deployment scenario	
E7 Execution steps – describe differences in comparison with the execution steps defined in test spec. – limitations	
E8 Test results –including outputs of the test properties and the attachment of log file(s) and/or screenshots	
E9 Notes, including observed issues with the solutions	
E10 Conclusions – pass/fail – assessment of test results in comparison with the expected results – gap analysis	

Revision history

Date	Revision	Description
2021.11.10	01.00	Final initial version 01.00
2022.03.23	02.00.07	Updated clauses: 7.4 Network Protocol Fuzzing 9.4 Software Bill of Materials (SBOM) 11.2 Open Fronthaul Point-to-Point LAN Segment 17.2 O1 Interface Network Configuration Access Control Model (NACM) Validation
2022.07.19	03.00.01	Applied the latest O-RAN technical specifications template Updated clauses: 2.1 Normative references 3.2 Abbreviations 6.3 TLS 6.6 OAuth 2.0 9.5 Software Image Signing and Verification 13.2 Testing of IPSec on E2 14.2 Testing of TLS on A1
2022.07.25	03.00.02	Updated cross-reference clause numbers of the test cases to align with latest WG11 specs Updated table of contents
2023.03.09	04.00.00	Updated clauses: • 14 Security Test of xApps • 15 Security test of Non-RT RIC • 16 Security test of rApps Added content to: • 14.2 xApp Signing and Verification • 16.2 rApp Signing and Verification Change wording in many places to align document with ETSI PAS

2023.07.10	05.00.00	Added content to: • 9.4.3 SBOM Format • 9.4.4 SBOM Depth • 9.5.2 Software Signature Verification • 12.4 O-RU Security functional requirement and test cases • 13.2 IPSec on E2 interface • 13.3 Transactional APIs • 18.2 O2 Interface • 18.3 O-Cloud virtualization layer • 20 Security tests of Common Application Lifecycle Management • 21 Security test of O-CU-CP • 22 Security test of O-CU-UP • 23 Security test of O-DU Alignment for ETSI PAS
2024.03.20	07.00.00	• OAuth2.0 in Near-RT RIC • OCloud tests • Reorganization of interfaces testing (A1, R1) • Update of SCTP test cases, removing not related with security • TIFG E2E test cases adoption into clause 24, and needed update • SBOM and Package testing increased • E2 data validation tests for Near-RT RIC

2024.07.17	08.00.00	Added content to: • 6.9 X.509 • 7.5 Denial of Service/Message flooding • 7.6 Input validation and error handling • 7.7 Secure configuration enforcement • 11 Security tests of O-RAN interfaces • 18.4 Application instantiation by O-Cloud • 18.9 Secure time synchronization for O-Cloud • 20.2 Application Package Removed content from: • 6.2 SSH • 6.3 TLS • 6.4 DTLS • 6.7 NACM • 6.8 802.1X • 7.5 Denial of Service/Message flooding • 7.7 Secure configuration enforcement • 9.4 SBOM • 11 Security tests of O-RAN interfaces

2023.11.27	09.00.00	Changed content in: • 6 Security Protocol & APIs Validation • 7 Common Network Security Tests for O-RAN architecture elements • 8 System security evaluation for O-RAN architecture element • 9 Software security evaluation for O-RAN architecture elements • 11.1 FH • 14 Security test of xApps • 18.4 Application instantiation deployment by O-Cloud • 19.2 Executive environment protection • 24 End-to-End security test cases Removed content from: • 6.10.1 eCPRI Session Management • 6.10.4 eCPRI Access Control • 6.10.6 eCPRI Timeout Error Handling • 8.4.9 Storage • 9.4.5 SBOM completeness check • 9.4.10 SBOM OSC Components • 9.5.1 Software Image/Application Package Signing

Annex (informative): Change History

Date	Revision	Description
2024.11.27	09.00.00	Published as Final version 09.00
2024.07.17	08.00.00	Published as Final version 08.00
2024.03.20	07.00.00	Published as Final version 07.00
2023.11.27	06.00.00	Published as Final version 06.00
2023.07.11	05.00.00	Published as Final version 05.00
2023.03.05	04.00.00	Published as Final version 04.00
2022.07.19	03.00.01	Published as Final version 03.00
2022.03.23	02.00.07	Published as Final version 02.00
2021.11.10	01.00	Published as Final version 01.00

