|  |
| --- |
| webwxgetmsgimg (7).jpegO-RAN.WG11.OAuth2.0-Security-TR.0-R004-v03.00 |

|  |
| --- |
| Technical Report |
|  |

|  |
| --- |
| O-RAN Work Group 11 (Security Work Group)  OAuth 2.0 Security |

|  |
| --- |
|  |
| Copyright © 2024 by the O-RAN ALLIANCE e.V.  The copying or incorporation into any other work of part or all of the material available in this specification in any form without the prior written permission of O-RAN ALLIANCE e.V. is prohibited, save that you may print or download extracts of the material of this specification for your personal use, or copy the material of this specification for the purpose of sending to individual third parties for their information provided that you acknowledge O-RAN ALLIANCE as the source of the material and that you inform the third party that these conditions apply to them and that they must comply with them.  O-RAN ALLIANCE e.V., Buschkauler Weg 27, 53347 Alfter, Germany  Register of Associations, Bonn VR 11238, VAT ID DE321720189 |

Contents

List of figures 4

List of tables 5

Foreword 6

Modal verbs terminology 6

1 Scope 7

1.1 Consideration 7

2 References 7

2.1 Informative references 7

3 Definition of terms, symbols and abbreviations 8

3.1 Terms 8

3.2 Symbols 9

3.3 Abbreviations 10

4 Study and recommendation from IETF, 3GPP and ETSI on OAuth 2.0 10

4.1 General 10

4.2 IETF 6749 10

4.3 IETF 8705 10

4.4 3GPP TS 33.122 10

4.5 ETSI NFV SEC022 10

5 Threats 10

5.1 O-RAN Elements & Interfaces 10

5.2 Threats Template 10

5.3 Potential Threat and Exploits 11

5.4 OAuth 2.0 Threats 11

6 Threat Analysis 12

6.1 General OAuth 2.0 Threats 12

7 Risk Assessment 21

8 Security Controls 21

9 OAuth 2.0 Authorization framework 21

9.1 Overview 21

9.2 Key Issues 22

9.2.1 Key Issue #1: Registration of O-RAN architecture elements in OAuth authorization server 22

9.2.2 Key Issue #2: Provisioning of client ID and secrets to O-RAN architecture elements 22

9.2.3 Key Issue #3: Provisioning of authorization grants for O-RAN architecture elements 22

9.2.4 Key Issue #4: Revocation of client ID and secrets for O-RAN architecture elements 22

9.2.5 Key Issue #5: Revocation of authorization grants for O-RAN architecture elements 22

9.2.6 Key Issue #6 Secure transfer of client secrets to clients 22

9.2.7 Key Issue #7: Secure transit and storage of access and refresh tokens 22

9.2.8 Key Issue #8: Defining expiration times for access tokens and client secrets 23

9.2.9 Key Issue #9: Consumers based on Client\_IDs created with valid administration 23

9.2.10 Key Issue #10: Scope creation 23

9.2.11 Key Issue #11: Need to support RBAC (Role Based Access Control) for the O-RAN resource servers 24

9.2.12 Key Issue #12: Need to support secure transmission of request and response between client and authorization server, between client and resource server 24

9.3 Solutions 25

9.3.1 Solution #1: Registration of rApp 25

9.3.2 Solution #2: Request access token for R1 services 27

9.3.3 Solution #3: R1 services request with access token 29

9.3.4 Solution #4: Secure transmission of request and response between client to authorization server, between client and resource server for R1, A1 and O2 32

10 Study and Recommendation an O-RAN Authorization Server Architecture 38

11 Conclusion 38

Annex: Change history/Change request (history) 39

# List of figures

Figure 9.1-1 OAuth 2.0 framework for O-RAN architecture elements and applications …………….21

Figure 9.3.1-1 rApp Registration use case sequence diagram 25

Figure 9.3.2-1 Request access token for R1 services use case sequence diagram 27

Figure 9.3.3-1 R1 Service request using access token use case sequence diagram 29

Figure 9.3.4-1 Transmission between client and authorization server……………………………………………………32

Figure 9.3.4-2 Transmission between client and resource server…………………………………………………………33

Figure 9.3.4-3 Tls\_client\_oauth method………………………………………………………………………….……… 34

Figure 9.3.4-4 Private\_key\_jwt with JWE method………………………………………………………………………..34

Figure 9.3.4-5 Private\_key\_jwt method…………………………………………………………………………………...34

Figure 9.3.4-6 Client\_secret\_jwt method………………………………………………………………………………….35

Figure 9.3.4-7 Client\_secret\_basic method………………………………………………………………………………..35

Figure 9.3.4-8 Client\_secret\_post method…………………………………………………………………………….…...35

Figure 9.3.4-9 Self\_signed\_tls\_client\_auth method………………………………………………………………….……36

# List of tables

Table 5.4-1 OAuth 2.0 Threats 11

Table 9.3.1-1 rApp Registration use case 25

Table 9.3.2-1 Request access token for R1 services use case 27

Table 9.3.3-1 R1 Service request using access token use case 29

Table 9.3.4-1 Use case of secure data transmission between client and authorization server……………………………31

Table 9.3.4-2 Authentication methods……………………………………………………………………………………33

Table 9.3.4-3 Authentication methods preferences……………………………………………………………………….36

# Foreword

This Technical Report (TR) 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 technical report defines the authorization framework for O-RAN elements using OAuth 2.0 and applies to the HTTP and Kafka based APIs [i.13] [i.18] used in the R1, O1, O2, A1, FH m-plane [i.17] and Y1 interfaces.

For this objective, the present document

* Defines the procedure flow for token registration process, token request process, token verification process and token authorization process. It will specify the token scope metadata for O-RAN.
* Analyses the existing specifications related to OAuth 2.0 such as O-RAN Security protocol specification, IETF RFC 8705[i.6], IETF RFC 6749[i.5], 3GPP TS29.501[i.3], 3GPP TS 33.122[i.4], 3GPP TS 28.319[i.19] and ETSI NFV SEC-022[i.7]. This is used as input to the present document.
* Analyses the security threat arising from the misuse of the access token and defines the security requirements associated to access token.
* Provides the study and recommendation of an authorization server architecture to overlay on the O-RAN architecture.

## 1.1 Consideration

This Technical Report makes the following considerations:

* O-RAN Alliance WG1 has identified the key functions and interface [i.9] adopted in O-RAN and this will influence the set of assets to be protected for OAuth 2.0 framework.
* This TR will perform a detailed study of the OAuth 2.0 Authorization server for O-RAN Architecture Elements and interface.
* Clause 4.7 in O-RAN Security Protocols Specifications [i.1] will be considered as the baseline text for the OAuth 2.0 framework.
* The recommended security requirements and controls provided in this report will be shared with the impacted O-RAN Alliance working group, such as WG1, WG2, WG3, WG6 and WG10 prior to publishing them as normative.

# 2 References

## 2.1 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.

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

[i.1] O-RAN ALLIANCE TS: “O-RAN Security Protocols Specifications”

[i.2] O-RAN ALLIANCE TS: “O-RAN Security Requirements and Controls Specifications”

[i.3] 3GPP TS 29.501:"5G System; Network Function Repository Services"

[i.4] 3GPP TS 33.122:"Security aspects of Common API Framework (CAPIF) for 3GPP northbound APIs"

[i.5] IETF RFC 6749: "The OAuth 2.0 Authorization Framework"

[i.6] IETF RFC 8705: "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens"

[i.7] ETSI GS NFV-SEC 022: "Network Functions Virtualisation (NFV) Release 2; Security; Access Token

Specification for API Access"

[i.8] IETF RFC 6819: “OAuth 2.0 Threat Model and Security Considerations”

[i.9] O-RAN ALLIANCE TS: “O-RAN Architecture Description”

[i.10]3GPP TS 23.222: "Common API Framework for 3GPP Northbound APIs"

[i.11]3GPP TS 29.222: "Common API Framework for 3GPP Northbound APIs"

[i.12] IETF RFC 2616: “Hypertext Transfer Protocol -- HTTP/1.1”

[i.13] O-RAN TS: "R1 interface: General Aspects and Principles" ("R1GAP")

[i.14] O-RAN TS: " Non-RT RIC & A1/R1 Interface: Use Cases and Requirements” ("UCR")

[i.15] O-RAN TR: “Decoupled SMO”

[i.16] O-RAN TR: “SMO Security”

[i.17] O-RAN TS: “Management Plane Specification”

[i.18] Kafka Documentation <https://kafka.apache.org/documentation.html>

[i.19] 3GPP TS 29.319: " Management and orchestration; Access control for management services"

[i.20] O-RAN ALLIANCE TR: "O-RAN Threat Modelling and Risk Assessment"

[i.21] OpenID Connect Core 1.0 incorporating errata set 2

[i.22] IETF RFC 7591: OAuth 2.0 Dynamic Client Registration Protocol

# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the following terms apply:

**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.

**Access Token**: credentials used to obtain access. It is a string, usually opaque to the client, representing an authorization issued to the client. Tokens represent specific scopes and durations of access, granted by the resource owner, and enforced by the resource server and authorization server (see [i.5], clause 1.4).

**Authorization code:** The authorization code is obtained by using an authorization server as an intermediary between the client and resource owner. Instead of requesting authorization directly from the resource owner, the client directs the resource owner to an authorization server (via its user-agent as defined in [RFC2616]), which in turn directs the resource owner back to the client with the authorization code. ([i.5], clause 1.3.1)

**Authorization Server:** The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization. ([i.5], clause 1.1)

**Client:** An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices). ([i.5], clause 1.1)

**Client Credentials:** The client credentials (or other forms of client authentication) can be used as an authorization grant when the authorization scope is limited to the protected resources under the control of the client, or to protected resources previously arranged with the authorization server. Client credentials are used as an authorization grant typically when the client is acting on its own behalf (the client is also the resource owner) or is requesting access to protected resources based on an authorization previously arranged with the authorization server. ([i.5], clause 1.3.4)

**CSRF:** Cross-site request forgery (CSRF) is a web-based attack whereby HTTP requests are transmitted from a user that the web site trusts or has authenticated (e.g., via HTTP redirects or HTML forms). ([i.8], clause 4.4.1.8)

**Near-Real-Time RAN Intelligent Controller:** An O-RAN Network Function (NF) that enables near-real-time control and optimization of RAN elements and resources via fine-grained data collection and actions over E2 interface. It may include AI/ML (Artificial Intelligence / Machine Learning) workflow including model training, inference and updates.

**Non-Real-Time RAN Intelligent Controller:** A functionality within SMO that drives the content carried across the A1 interface. It is comprised of the Non-RT RIC Framework and the Non-RT RIC Applications (rApps) whose services are defined below.

**Non-RT RIC Applications (rApps):** Modular applications that leverage the functionality exposed via the Non-RT RIC Framework’s R1 interface to provide added value services relative to RAN operation, such as driving the A1 interface, recommending values and actions that may be subsequently applied over the O1/O2 interface and generating “enrichment information” for the use of other rApps. The rApp functionality within the Non-RT RIC enables non-real-time control and optimization of RAN elements and resources and policy-based guidance to the applications/features in Near-RT RIC.

**O2**: Interface between SMO framework as specified in Clause 5.3.1 and the O-Cloud for supporting O-RAN virtual network functions. Please refer to [13] for more information.

**Open Redirector:** An open redirector is an endpoint using a parameter to automatically redirect a user-agent to the location specified by the parameter value without any validation. ([i.5], clause 10.15)

**R1 Interface:** Interface between rApps and Non-RT RIC Framework via which R1 Services can be produced and consumed.

**Refresh token:** Refresh tokens are credentials used to obtain access tokens. Refresh tokens are issued to the client by the authorization server and are used to obtain a new access token when the current access token becomes invalid or expires, or to obtain additional access tokens with identical or narrower scope (access tokens may have a shorter lifetime and fewer permissions than authorized by the resource owner). ([i.5], clause 1.5)

**Resource Owner:** An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user. ([i.5], clause 1.1)

**Resource Server:** The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens. ([i.5], clause 1.1)

**SMO:** A Service Management and Orchestration framework.

**User-Agent:** The User-Agent request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. ([i.12], clause 14.43)

**xApp:** An application designed to run on the Near-RT RIC. Such an application is likely to consist of one or more microservices and at the point of on-boarding will identify which data it consumes and which data it provides. The application is independent of the Near-RT RIC and may be provided by any third party. The E2 enables a direct association between the xApp and the RAN functionality.

**Y1**: An interface over which RAN analytics services are exposed by the Near-RT RIC to be consumed by Y1 consumers.

## 3.2 Symbols

For the purposes of the present document, the following symbols apply:

None

## 3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP 3rd Generation Partnership Project

Near-RT RIC Near-Real-Time RAN Intelligent Controller

rApp Non-RT RIC Application

SMO Service Management and Orchestration

xApp Near-RT RIC Application

# 4 Study and recommendation from IETF, 3GPP and ETSI on OAuth 2.0

## 4.1 General

## 4.2 IETF 6749

Editor’s note: This clause will do the study related to OAuth 2.0 framework from IETF RFC 6749

## 4.3 IETF 8705

Editor’s note: This clause will do the study related to OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens from IETF RFC 8705

## 4.4 3GPP TS 33.122

Editor’s note: This clause will do the study related to OAuth 2.0 recommended from 3GPP specification (23.222, 29.222, 33.122, TS29.501)

## 4.5 ETSI NFV SEC022

Editor’s note: This clause will do the study related to OAuth 2.0 mechanism and definition from ETSI NFV SEC022

# 5 Threats

## 5.1 O-RAN Elements & Interfaces

## 5.2 Threats Template

Template to present the threat characteristics:

|  |  |
| --- | --- |
| **Threat ID** | O-RAN threat ID |
| **Threat title** | [Threat Title will include suitable reference of 3GPP/IETF/ETSI] |
| **Threat description** | Introduction about the threat |
| **Threat type** | Spoofing  Tampering  Repudiation  Information disclosure  Denial of Service  Elevation of Privilege |
| **Vulnerability** |  |
| **Impact type** | Authenticity  Integrity  Non-repudiation  Confidentiality  Availability  Authorization |
| **Affected Assets** | Asset identification |

## 5.3 Potential Threat and Exploits

## 5.4 OAuth 2.0 Threats

The following threats to OAuth 2.0 have been identified and are analyzed in clause 6– Threat Analysis.

Table 5.4-1 – OAuth 2.0 Threats

|  |  |
| --- | --- |
| **Threat-Id** | **Threat Description (Brief)** |
| **General OAuth 2.0 Threats** | |
| T-OAuth-001 | Obtaining client secrets |
| T-OAuth-002 | Obtaining refresh tokens |
| T-OAuth-003 | Obtaining access tokens |
| T-OAuth-004 | Eavesdropping access tokens |
| T-OAuth-005 | Disclosure of client credentials during transmission |
| T-OAuth-006 | Open redirectors on client |
| T-OAuth-016 | Obtaining authorization by code substitution (OAuth Login) |
| T-OAuth-020 | Access token leak in transport/endpoints |
| T-OAuth-021 | Malicious client obtains existing authorization by fraud |
| T-OAuth-022 | Malicious client obtains authorization |
| T-OAuth-023 | Accidental exposure of passwords at client side |
| T-OAuth-024 | Client obtains scopes without end-user authorization and possible misconfigurations |
| T-OAuth-025 | Client obtains refresh token through automatic authorization |
| T-OAuth-026 | Eavesdropping refresh tokens from authorization server |
| T-OAuth-027 | Obtaining refresh token by online guessing |
| T-OAuth-028 | Refresh token phishing by counterfeit authorization server |
| T-OAuth-029 | Guessing access tokens |
| T-OAuth-030 | Access token phishing by counterfeit resource server |
| T-OAuth-031 | Abuse of token by legitimate resource server or client |
| T-OAuth-032 | Token leakage via log files and http referrers |
| **Threats at OAuth Authorization Server** | |
| T-OAuth-007 | Password phishing by counterfeit authorization server |
| T-OAuth-008 | Malicious client obtains existing authorization by fraud |
| T-OAuth-009 | Obtaining access tokens from authorization server database |
| T-OAuth-010 | Obtaining client secret from authorization server database |
| T-OAuth-011 | Online guessing of authorization codes |
| T-OAuth-012 | DoS attacks that exhaust resources |
| T-OAuth-013 | CSRF attack against redirect-uri |
| T-OAuth-014 | Resource owner impersonation |
| T-OAuth-015 | DoS using manufactured authorization "codes" |
| T-OAuth-017 | Obtaining user passwords from authorization server database |
| T-OAuth-018 | Obtaining the resource owner password credentials by online guessing |
| T-OAuth-019 | Obtaining refresh token from authorization server database |
| **Access control related Threats** | |
| T-OAuth-033 | Attacks from the internet exploit weak authentication and access control to penetrate O-RAN network boundary |
| T-OAuth-034 | An attacker exploits insufficient/improper mechanisms for authentication and authorization to compromise O-RAN components |
| T-OAuth-035 | An attacker gains unauthorized access to R1 services |
| T-OAuth-036 | An attacker gains unauthorized access to data |
| T-OAuth-037 | Attacker exploits missing or improperly defined elements of application’s SecurityDescriptor |
| T-OAuth-038 | Malicious access to exposed services using valid accounts |
| T-OAuth-039 | Sensitive data at rest is exposed to an internal attacker |

# 6 Threat Analysis

## 6.1 General OAuth 2.0 Threats

|  |  |
| --- | --- |
| Threat ID | T-OAuth-001 |
| **Threat title** | Obtaining Client Secrets, IETF RFC 6819[i.8], clause [4.1.1](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) |
| **Threat description** | A malicious actor may be able to get access to the secrets of O-RAN elements and obtain the tokens in order to replay its refresh tokens and authorization codes. |
| **Threat type** | Information disclosure  Elevation of Privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-002 |
| **Threat title** | Obtaining refresh tokens, IETF RFC 6819[i.8], clause 4.1.2 |
| **Threat description** | A malicious actor may try to get the file system access of the local storage of O-RAN elements and read the refresh tokens. A malicious actor may use the compromised refresh token for a malicious application. |
| **Threat type** | Information disclosure |
| **Impact type** | Confidentiality |
| **Affected Asset** | O-RAN Architecture Elements that obtain an access token from an OAuth Authorization Server (eg: rApp, xApp) |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-003 |
| **Threat title** | Obtaining access tokens, IETF RFC 6819[i.8], clause 4.1.3 |
| **Threat description** | Access tokens may be stolen by a malicious actor from the local storage if the O-RAN elements store them in a local storage that is accessible to other O-RAN elements or applications without restriction. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | O-RAN Architecture Elements and Applications acting as OAuth clients (eg: rApp, xApp) |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-004 |
| **Threat title** | Eavesdropping Access Tokens, IETF RFC 6819[i.8], clause 4.3.1 |
| **Threat description** | A malicious actor may attempt to eavesdrop access tokens in transit from the authorization server to the O-RAN elements and applications |
| **Threat type** | Information disclosure |
| **Impact type** | Confidentiality |
| **Affected Asset** | O-RAN Architecture Elements and Applications acting as OAuth clients (eg: rApp, xApp) |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-005 |
| **Threat title** | Disclosure of Client Credentials during Transmission, IETF RFC 6819[i.8], clause 4.3.3 |
| **Threat description** | A malicious actor may attempt to eavesdrop the transmission of O-RAN elements credentials between the application and authorization server during the O-RAN elements authentication process or during OAuth token requests. |
| **Threat type** | Information disclosure |
| **Impact type** | Confidentiality |
| **Affected Asset** | O-RAN Architecture Elements and Applications acting as OAuth clients (eg: rApp, xApp) |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-006 |
| **Threat title** | Open Redirectors on Client, IETF RFC 6819[i.8], clause 4.1.5 |
| **Threat description** | A malicious actor may use an open redirector operated by the O-RAN Applications to construct a redirect URI that will pass the authorization server validation but will send authorization code to an endpoint under the control of a malicious actor. |
| **Threat type** | Information disclosure |
| **Impact type** | Authenticity |
| **Affected Asset** | O-RAN Elements and Applications acting as OAuth clients (eg: rApp, xApp) |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-007 |
| **Threat title** | Password phishing by counterfeit authorization server, IETF RFC 6819[i.8], clause 4.2.1 |
| **Threat description** | A malicious actor may steal confidential data (eg. Password) by intercepting the O-RAN Application’s request and exploit the URL endpoints managed by authorization server. |
| **Threat type** | Information disclosure |
| **Impact type** | Authenticity |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-008 |
| **Threat title** | Malicious Client Obtains Existing Authorization by Fraud, IETF RFC 6819[i.8], clause 4.2.3 |
| **Threat description** | A malicious actor may exploit the process of authorization server which automatically process the authorization requests from O-RAN Application that have been previously authorized and try to obtain the authorization code instead of the legitimate O-RAN Application. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-009 |
| **Threat title** | Obtaining access tokens from authorization server database, IETF RFC 6819[i.8], clause 4.3.2 |
| **Threat description** | A malicious actor may obtain access tokens from the authorization server's database by gaining access to the database or launching a SQL injection attack. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-010 |
| **Threat title** | Obtaining Client Secret from Authorization Server Database, IETF RFC 6819[i.8], clause 4.3.4 |
| **Threat description** | A malicious actor may obtain valid client\_id/secrets combinations from the authorization server's database by gaining access to the database or launching a SQL injection attack. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-011 |
| **Threat title** | Online Guessing of Authorization codes, IETF RFC 6819[i.8], clause 4.4.1.3 |
| **Threat description** | A malicious actor may try to guess valid authorization code values and send the guessed code value using the grant type code in order to obtain a valid access token. |
| **Threat type** | Information disclosure |
| **Impact type** | Confidentiality |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-012 |
| **Threat title** | DoS Attacks That Exhaust Resources, IETF RFC 6819[i.8], clause 4.4.1.11 |
| **Threat description** | A malicious actor may exhaust the pool of authorization codes of the authorization server by repeatedly directing the O-RAN Application to request the access tokens. |
| **Threat type** | Denial of Service |
| **Impact type** | Availability |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-013 |
| **Threat title** | CSRF Attack against redirect-uri, IETF RFC 6819[i.8], clause 4.4.1.8 |
| **Threat description** | CSRF attacks on OAuth approvals may allow a malicious actor to obtain authorization to OAuth protected resources without the consent of the O-RAN Applications. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-014 |
| **Threat title** | Resource Owner Impersonation, IETF RFC 6819[i.8], clause 4.4.1.10 |
| **Threat description** | A malicious actor may exploit an OAuth authorization server's ability to gain authorization without the OAuth 2.0 resource owner's consent by automatically approving the request using existing authenticated session. |
| **Threat type** | Elevation of Privilege |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-015 |
| **Threat title** | DoS Using Manufactured Authorization "codes", IETF RFC 6819 [i.8], clause 4.4.1.12 |
| **Threat description** | An attacker who owns a botnet may locate the redirect URIs of clients that listen on HTTP, access them with random authorization "codes", and cause a large number of HTTPS connections to be concentrated onto the authorization server. This can result in a denial-of-service (DoS) attack on the authorization server. |
| **Threat type** | Denial of Service |
| **Impact type** | Availability |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-016 |
| **Threat title** | Obtaining authorization by code substitution (OAuth Login), IETF RFC 6819 [i.8], clause 4.4.1.13 |
| **Threat description** | An attacker could attempt to log into an application using a victim's identity. Applications relying on identity data provided by an OAuth protected service API to login users are vulnerable to this threat. |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-017 |
| **Threat title** | Obtaining user passwords from authorization server database, IETF RFC 6819 [i.8], clause 4.4.3.5 |
| **Threat description** | An attacker may obtain valid username/password combinations from the authorization server's database by gaining access to the database or launching a SQL injection attack. |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization  Information Disclosure |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-018 |
| **Threat title** | Obtaining the resource owner password credentials by online guessing, IETF RFC 6819 [i.8], clause 4.4.3.6 |
| **Threat description** | An attacker may try to guess valid username/password combinations using the grant type "password". |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-019 |
| **Threat title** | Obtaining refresh token from authorization server database, IETF RFC 6819 [i.8], clause 4.5.2 |
| **Threat description** | This threat is applicable if the authorization server stores refresh tokens as handles in a database. An attacker may obtain refresh tokens from the authorization server's database by gaining access to the database or launching a SQL injection attack. |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-020 |
| **Threat title** | Access token leak in transport/endpoints, IETF RFC 6819 [i.8], clause 4.4.2.1 |
| **Threat description** | A token may be eavesdropped by an attacker. The token is sent from the server to the client via a URI fragment of the redirect URI. If the communication is not secured or the endpoint is not secured, the token could be leaked by parsing the returned URI. |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization |
| **Affected Asset** | OAuth Authorization Server, O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-021 |
| **Threat title** | Malicious Client Obtains Existing Authorization by Fraud, IETF RFC 6819 [i.8], clause 4.2.3 |
| **Threat description** | Authorization servers may wish to automatically process authorization requests from clients that have been previously authorized by the user. When the user is redirected to the authorization server's end-user authorization endpoint to grant access, the authorization server detects that the user has already granted access to that particular client. Instead of prompting the user for approval, the authorization server automatically redirects the user back to the client.  A malicious client may exploit that feature and try to obtain such an authorization "code" instead of the legitimate client. |
| **Threat type** | Spoofing  Repudiation |
| **Impact type** | Authorization  Non repudiation |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-022 |
| **Threat title** | Malicious Client Obtains Authorization, IETF RFC 6819 [i.8], clause 4.4.2.3 |
| **Threat description** | A malicious client may attempt to obtain a token by fraud. |
| **Threat type** | Spoofing  Repudiation |
| **Impact type** | Authorization  Non repudiation |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-023 |
| **Threat title** | Accidental exposure of passwords at client side, IETF RFC 6819 [i.8], clause 4.4.3.1 |
| **Threat description** | If the client does not provide enough protection, an attacker could retrieve the passwords for a user. |
| **Threat type** | Information Disclosure |
| **Impact type** | Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-024 |
| **Threat title** | Client obtains scopes without end-user authorization and possible misconfigurations, IETF RFC 6819 [i.8], clause 4.4.3.2 |
| **Threat description** | An O-RAN Application like a rApp or xApp intentionally or unintentionally obtains a token with scope unknown for, or unintended by, the resource owner. |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-025 |
| **Threat title** | Client Obtains Refresh Token through Automatic Authorization, IETF RFC 6819 [4.4.3.3](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) [i.8], clause [4.4.3.3](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) |
| **Threat description** | An O-RAN application like a rApp or xApp obtains intentionally or unintentionally a long-term authorization represented by a refresh token even if the resource owner did not intend so |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-026 |
| **Threat title** | Eavesdropping Refresh Tokens from Authorization Server, IETF RFC 6819 [i.8], clause [4.5.1](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) |
| **Threat description** | An O-RAN Application like a rApp or xApp may eavesdrop refresh tokens when they are transmitted from the authorization server to the client. |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-027 |
| **Threat title** | Obtaining Refresh Token by Online Guessing, IETF RFC 6819 [i.8], clause 4.5.3 |
| **Threat description** | An O-RAN Application like a rApp or xApp may try to guess valid refresh token values and send it using the grant type "refresh\_token" in order to obtain a valid access token. |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-028 |
| **Threat title** | Refresh Token Phishing by Counterfeit Authorization Server, IETF RFC 6819 [i.8], clause 4.5.4 |
| **Threat description** | An O-RAN Application like a rApp or xApp could obtain valid refresh tokens by proxying requests to the authorization server, given the assumption that the authorization server URL is well-known at development time or can at least be obtained from a well-known resource server |
| **Threat type** | Tampering  Spoofing  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the rApp, xApp |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-029 |
| **Threat title** | Guessing Access Tokens, IETF RFC 6819 [i.8], clause [4.6.3](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) |
| **Threat description** | Access to a single user’s data is possible where an attacker may place a malicious rApp or xApp and attempt to guess the access token values based on knowledge from other access tokens. |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the SMO, Near-RT RIC |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-030 |
| **Threat title** | Access Token Phishing by Counterfeit Resource Server, IETF RFC 6819[i.8], clause [4.6.4](https://datatracker.ietf.org/doc/html/rfc6819#section-4.1.1) |
| **Threat description** | An O-RAN Application like a rApp or xApp may sends a valid access token to a counterfeit resource server where the server in turn uses that token to access other services on behalf of the resource owner. |
| **Threat type** | Tampering  Spoofing  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the SMO, Near-RT RIC |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-031 |
| **Threat title** | Abuse of Token by Legitimate Resource Server or Client, IETF RFC 6819 [i.8], clause 4.6.5 |
| **Threat description** | An O-RAN legitimate resource server which is protecting the resources could possibly use an access token to access another resource server. In a similar manner an O-RAN Application like a rApp or xApp could try to use a token obtained for one server on another resource server |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the SMO, Near-RT RIC |

|  |  |
| --- | --- |
| Threat ID | T-OAuth-032 |
| **Threat title** | Token Leakage via Log Files and HTTP Referrers, IETF RFC 6819 [i.8], clause 4.6.7 |
| **Threat description** | An O-RAN Application like a rApp or xApp could be implemented wrongly with respect to sending the access tokens URI query parameters and subsequently leak the access tokens to the log files and the HTTP "referer" |
| **Threat type** | Tampering  Elevation of privilege |
| **Impact type** | Confidentiality  Authorization |
| **Affected Asset** | O-RAN Architecture Elements like the SMO, Near-RT RIC |

# 7 Risk Assessment

# 8 Security Controls

# 9 OAuth 2.0 Authorization framework

## 9.1 Overview

Figure 9.1 depicts the basic OAuth 2.0 framework for O-RAN architecture elements and applications. This general authorization framework is generalized from clause 1 of IETF RFC 6749 [i.5].

[17

O-RAN OAuth 2.0 Framework

O-RAN Client

Resource Owner

Authorization

Server

Resource

Server

1.O-RAN Client Authorization Request

2. Resource Owner Authorization Grant

3. O-RAN Client Authorization Grant

6. Protected Resource

5. O-RAN Client Access Token

4. Authorization Server Access Token

Figure 9.1-1 OAuth 2.0 framework for O-RAN architecture elements and applications

O-RAN OAuth 2.0 framework defines four roles from clause 1 of IETF RFC 6749 [i.5]:

* O-RAN client: An O-RAN client is an O-RAN Architecture Element or Application like an rApp or xApp acting as an OAuth client.
* Authorization server: The server issuing access tokens to the O-RAN client after successfully authenticating the resource owner and obtaining authorization.
* Resource owner: “An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.” ([i.5], clause 1.1)
* Resource server: “The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.” ([i.5], clause 1.1)

## 9.2 Key Issues

## 9.2.1 Key Issue #1: Registration of O-RAN architecture elements in OAuth authorization server

## 9.2.2 Key Issue #2: Provisioning of client ID and secrets to O-RAN architecture elements

## 9.2.3 Key Issue #3: Provisioning of authorization grants for O-RAN architecture elements

## 9.2.4 Key Issue #4: Revocation of client ID and secrets for O-RAN architecture elements

## 9.2.5 Key Issue #5: Revocation of authorization grants for O-RAN architecture elements

## 9.2.6 Key Issue #6 Secure transfer of client secrets to clients

#### 9.2.6.1 Key issue detail

The need is to identify

* How the client secrets are transferred to potential clients
* Whether the client secrets will be known at development time or runtime?

#### 9.2.6.2 Security threats

|  |  |
| --- | --- |
| T-OAuth-001 | Obtaining client secrets |
| T-OAuth-005 | Disclosure of client credentials during transmission |
| T-OAuth-007 | Password phishing by counterfeit authorization server |
| T-OAuth-023 | Accidental exposure of passwords at client side |
| T-OAuth-010 | Obtaining client secret from authorization server database |

## 9.2.7 Key Issue #7: Secure transit and storage of access and refresh tokens

#### Key issue detail

It needs to be identified what possible mechanisms exists to ensure the integrity and confidentiality of access and refresh tokens.

#### Security threats

|  |  |
| --- | --- |
| T-OAuth-002 | Obtaining refresh tokens |
| T-OAuth-003 | Obtaining access tokens |
| T-OAuth-004 | Eavesdropping access tokens |
| T-OAuth-020 | Access token leak in transport/endpoints |
| T-OAuth-026 | Eavesdropping refresh tokens from authorization server |
| T-OAuth-027 | Obtaining refresh token by online guessing |
| T-OAuth-029 | Guessing access tokens |

## 9.2.8 Key Issue #8: Defining expiration times for access tokens and client secrets

#### Key issue detail

It needs to be identified if

* Access tokens need to be configured with the minimal expiration time and in what use cases
* Client secrets needs to be revoked and refreshed regularly and if so at what intervals

#### Security threats

|  |  |
| --- | --- |
| T-OAuth-007 | Password phishing by counterfeit authorization server |
| T-OAuth-008 | Malicious client obtains existing authorization by fraud |
| T-OAuth-009 | Obtaining access tokens from authorization server database |
| T-OAuth-010 | Obtaining client secret from authorization server database |

## 9.2.9 Key Issue #9: Consumers based on Client\_IDs created with valid administration

#### 9.2.9.1 Key issue detail

It needs to be identified if

* Consumers who connect are only those validated by administrators
* Dynamic clients are possible or not

#### Security threats

|  |  |
| --- | --- |
| T-OAuth-022 | Malicious client obtains authorization |

## 9.2.10 Key Issue #10: Scope creation

#### 9.2.10.1 Key issue detail

It needs to be checked as to how scopes are designed, provisioned and enforced. Key topics are

* How scopes are designed by the resource servers including the clearance and access levels
* How scopes are transferred the authorization server
* How scopes are requested by the client
* How scopes are created in the authorization server

#### 9.2.10.2 Security threats

|  |  |
| --- | --- |
| T-OAuth-022 | Malicious client obtains authorization |
| T-OAuth-024 | Client obtains scopes without end-user authorization and possible misconfigurations |

## 9.2.11 Key Issue #11: Need to support RBAC (Role Based Access Control) for the O-RAN resource servers

The O-RAN resource servers need to be accessed by only those users who are authorized to carry out the specific task. Hence, we need a mechanism to design and map the O-RAN resource server to the users for access control.

#### 9.2.11.1 Key issue detail

It needs to be checked as to

* How to limit the access of resources to specified users (least privilege) and only for the intended task.
* How to manage access (assign users to permissions related to the resource servers) in a fast and efficient manner.
* How to retain a role only for the period of time required and no longer.

#### 9.2.11.2 Security threats

The threats related to limited access control are below.

* Malicious attacks with open access to the entire system of assets and data
* Insider threats
* Privilege stagnation over a period of time

Editor notes: Some of the threats id’s such as T-OAuth-033, T-OAuth-034, T-OAuth-035, T-OAuth-036, T-OAuth-037, T-OAuth-038, T-OAuth-039 in clause 5.4 of the present document are mapped and identified in the O-RAN Threat Modelling and Remediation Analysis document [6], need to be discussed further.

## 9.2.12 Key Issue #12: Need to support secure transmission of request and response between client and authorization server, between client and resource server

The O-RAN related HTTP clients (e.g.: R1, A1, O2) need to make a request to the authorization server and receive a response with respect to the access token. After that the access token has to be transferred from the client to the resource server.

#### 9.2.12.1 Key issue detail

It needs to be checked as to

* How the client will send request details including credentials related to client\_id and client\_secret to the authorization server
* How the client will receive the response from the authorization server
* How the client will send the access token to the resource server

#### 9.2.12.2 Security threats

The threats related to insecure transmission of request and response are

|  |  |
| --- | --- |
| T-OAuth-001 | Obtaining client secrets |
| T-OAuth-003 | Obtaining access tokens |
| T-OAuth-005 | Disclosure of client credentials during transmission |

## 9.3 Solutions

## 9.3.1 Solution #1: Registration of rApp

9.3.1.1 Background

This use case allows an rApp to register to the SMO as specified in R1GAP [i.13][i.14]. The registration procedure is specified as part of Service Management and Exposure services in R1GAP [i.13][i.14].

9.3.1.2 Solution description

Table 9.3.1-1 rApp Registration use case

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution/Specification | <<Uses>>  Related use |
| Goal | 1. Register the rApp to the SME services Producer in SMO framework. |  |
| Actors and Roles | * rApp in the role of Service Producer and/or Service Consumer * SME in the role of SME services Producer. |  |
| Assumptions | n/a |  |
| Preconditions | The rApp is instantiated. |  |
| Begins when | The rApp Management SMOS is ready for registration. |  |
| Step 1 (M) | The rApp Management SMOS sends a registration request to the SME in the SMO passing relevant information needed. |  |
| Step 1 (M) | The SME process the rApp registration request that may include.  - performing authentication,  - if rApp is authenticated, rAppId will be assigned to rApp. |  |
| Step 3 (M) | Response to the registration request along with the rAppId. |  |
| Ends when | The rApp is allocated with an rAppId. |  |
| Exceptions | n/a |  |
| Post Conditions | The rApp is registered with the SMO framework and an rAppId is allocated to the rApp. |  |
| Traceability | REQ-R1-SME-rAppreg-FUN1, REQ-R1-SME-rAppreg-FUN2. |  |

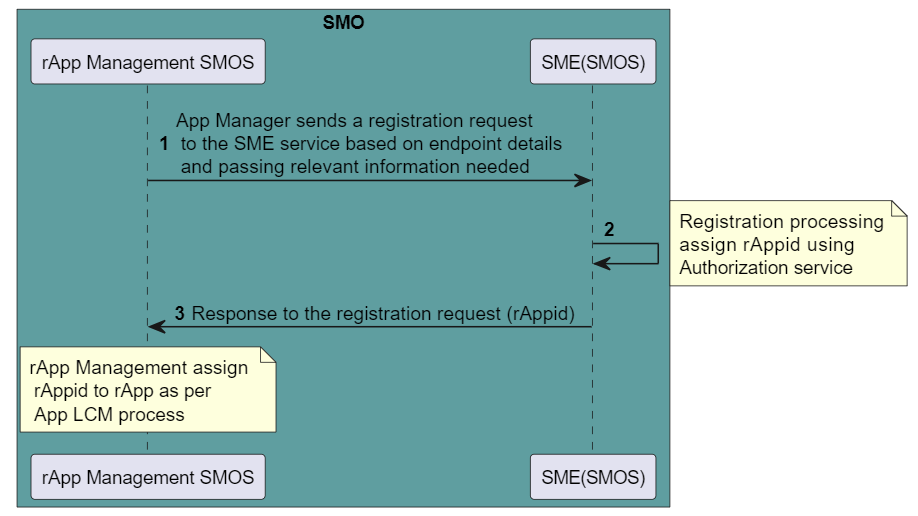


Figure 9.3.1-1 rApp Registration use case sequence diagram

9.3.1.3 Impacts on existing O-RAN architecture elements and interfaces

R1 interface, rApp and SME are the O-RAN architecture elements and interfaces involved in the proposed use cases as specified in Decoupled SMO [i.15] [i.16].

1) Service Management and Exposure (SME) SMOS

The SME service is introduced in the [i.8] section 3.1. While the SME was initially targeted for management and exposure of rApp services in the [i.8], its services are applicable to any entities within the SMO that fulfil the roles of Service Producer and/or Service Consumer.

In a decomposed service-based SMO architecture, the SME can be leveraged as a generic SMOS, handling service management and exposure for any SMOSs within SMO. The SME SMOS provides a set of common SMO capabilities that are needed for all SMO Services.

Several of the SME services are applicable to any SMOSs:

* SMOS registration,
* SMOS discovery,
* Authentication and authorization to access a SMOS,
* Communication support between SMOS Service Producers and SMOS Service Consumers (SMOS Communication),
* Bootstrap of new SMOSs (optional),
* Heartbeat of SMOSs (optional).

1. rApp Management (SMOS)

The rApp management service handles management aspects specific to rApps such as:

* rApp Configuration Management (CM): enables management of the rApp configuration settings,
* rApp Performance Management (PM): offers support for performance reporting on an rApp performance, or of the rApp communication with the non-RT RIC framework (rApp usage of R1 interface),
* rApp Fault Management (FM): provides the fault reporting capabilities to rApps, so they can consume it to report faults that are related to the rApp, or to the rApp communication with the non-RT RIC framework (rApp usage of R1 interface),
* rApp Logging Information: enables rApps to convey their logging information to the non-RT RIC framework,
* rApp lifecycle management services, that expose services that allow to trigger rApp software lifecycle management.

9.3.1.4 Solution evaluation

Key Issue #1: Registration of O-RAN architecture elements in OAuth authorization server

9.3.1.5 Potential security recommendations

* OAuth protection of the R1 interface is already mandatory in the O-RAN Security Requirements and Controls Specifications [i.2] SEC-CTL-R1-3.

## 9.3.2 Solution #2: Request access token for R1 services

9.3.2.1 Background

This use case allows an rApp to request an access token for access to R1 services as specified in R1GAP [i.13] [i.14]. The request access procedure is specified as part of Service Management and Exposure services in R1GAP [i.13][i.14].

An rApp in the role of Service Consumer will be able to request the token and once the token is available, the rApp in the role of Service Consumer will be able access the R1 Services for which it has been authorized.

9.3.2.2 Solution description

Table 9.3.2-1 Request access token for R1 services use case

| Use case stage | Evolution / Specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | To enable rApp to request an access token for access to R1 services. |  |
| Actors and Roles | rApp in the role of Service Consumer that requests token for access to R1 services.  SME as SMOS in the role of SME services Producer support authorization service |  |
| Assumptions | rAppId and credentials have been provisioned. |  |
| Pre-conditions | R1 interface is established |  |
| Begins when | rApp has determined need for an access token to access R1 services and resources. |  |
| Step 1 (M) | rApp sends Access token request containing the rAppId and optionally token scope, | SPS [i.1], clause 4.7.2.3 |
| Step 2 (M) | SME authenticate the rApp and if the rApp token request is authorized, then an access token is generated. | SPS [i.1], clause 4.7.2.3 |
| Step 3 (M) | SME respond with access token. | SPS [i.1], clause 4.7.2.3 |
| Ends when | rApp has received an access token for access to R1 services. |  |
| Exceptions | n/a |  |
| Post-conditions | The rApp is authorized for accessing R1 services. |  |
| Traceability | REQ-R1-A-FUN1 |  |

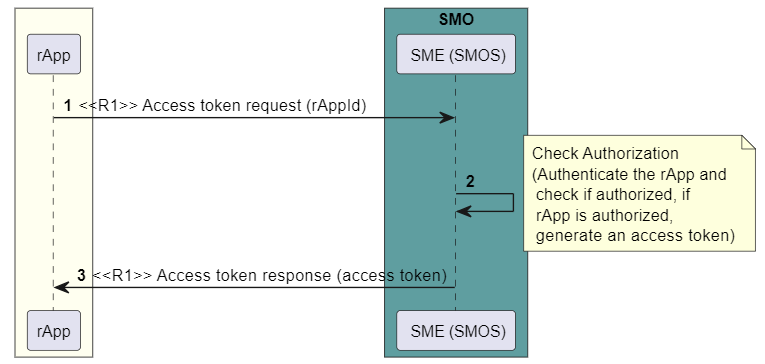


Figure 9.3.2-1 Request access token for R1 services use case sequence diagram

9.3.2.3 Impacts on existing O-RAN architecture elements and interfaces

R1 interface, rApp and SME are the O-RAN architecture elements and interfaces involved in the proposed use cases as specified in Decoupled SMO [i.15] [i.16].

1) Service Management and Exposure (SME) SMOS

The SME service is introduced in the [i.8] section 3.1. While the SME was initially targeted for management and exposure of rApp services in the [i.8], its services are applicable to any entities within the SMO that fulfil the roles of Service Producer and/or Service Consumer.

In a decomposed service-based SMO architecture, the SME can be leveraged as a generic SMOS, handling service management and exposure for any SMOSs within SMO. The SME SMOS provides a set of common SMO capabilities that are needed for all SMO Services. Several of the SME services are applicable to any SMOSs:

* SMOS registration,
* SMOS discovery,
* Authentication and authorization to access a SMOS,
* Communication support between SMOS Service Producers and SMOS Service Consumers (SMOS Communication),
* Bootstrap of new SMOSs (optional),
* Heartbeat of SMOSs (optional).

2) SME functions in the role of SME services Producer

* supports the functionality of OAuth 2.0 Authorization server,
* support authentication and authorization of an rApp to determine which R1 services an rApp can access,
* provide response of success or failure to access token request.

1. rApp

* supports initiating the procedure to request access token,
* supports initiating service request with the available access token.

9.3.2.4 Solution evaluation

Key Issue #3: Provisioning of authorization grants for R1 services

9.3.2.5 Potential security recommendation

* Use OAuth 2.0, as specified in O-RAN Security Protocols Specifications [i.1], clause 4.7 to implement authorization in the R1 interface.
* Enable rApp to request OAuth 2.0 access tokens for R1 services.
* Enable the SME to act as both an OAuth 2.0 resource and an OAuth 2.0 Authorization server.

## 9.3.3 Solution #3: R1 services request with access token

9.3.3.1 Background

This use case discuss R1 service Producer to verify that rApp is authorized to access the R1 service as specified in R1GAP [i.13] [i.14]. The service request procedure is specified as part of Service Management and Exposure services in R1GAP [i.13][i.14].

9.3.3.2 Solution description

Table 9.3.3-1 R1 Service request using access token use case

| Use case stage | Evolution/specification | <<Uses>> Related use |
| --- | --- | --- |
| Goal | To enable R1 service Producer to verify that rApp is authorized to access the R1 service. |  |
| Actors and Roles | rApp in the role of Service Consumer that requests access to an R1 service with access token.  R1 service producer in the role of SME services Producer that supports the validation of the access token and provides access to the requested service. |  |
| Assumptions | rApp has requested and received an access token for the R1 service for which service access is requested. |  |
| Pre-conditions | R1 interface is established. |  |
| Begins when | rApp initiates a request for an R1 Service to R1 Service producer. |  |
| Step 1 (M) | rApp sends service request containing an access token to R1 service Producer |  |
| Step 2 (M) | R1 service Producer verifies integrity and claims in the access token. If successful, the requested service is executed. | SPS [i.1] clause 4.7.2.4] |
| Step 3 &4 (M) | R1 service Producer sends response to the service request if it has valid token |  |
| Step 5 & 6 (O) | The verification of integrity and claims in step 3 was unsuccessful due to invalid or expired token, the requested service is not executed, and R1 service Producer sends a response with appropriate error code. |  |
| Ends when | R1 service Producer has responded to the service request. |  |
| Post-conditions | - |  |
| Traceability | REQ-R1-SME-A-FUN2 |  |

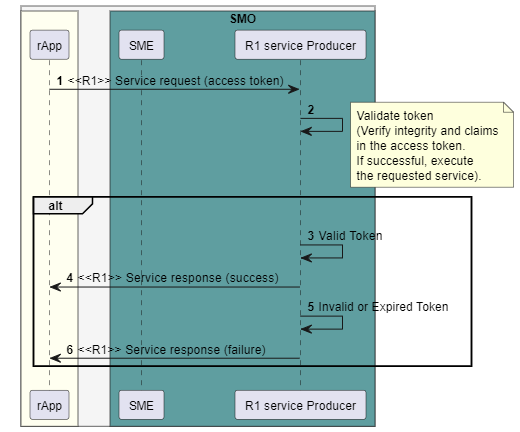


Figure 9.3.3-1 R1 Service request using access token use case sequence diagram

Editor Notes: “whether or not it is required for all API accesses to go through the SME need to be discussed further”.

9.3.3.3 Impacts on existing O-RAN architecture elements and interfaces

R1 interface, rApp and SME are the O-RAN architecture elements and interfaces involved in the proposed use cases as specified in Decoupled SMO [i.15] [i.16].

1) Service Management and Exposure (SME) SMOS

The SME service is introduced in the [i.13] section 3.1. While the SME was initially targeted for management and exposure of rApp services in the [i.13], its services are applicable to any entities within the SMO that fulfil the roles of Service Producer and/or Service Consumer.

In a decomposed service-based SMO architecture, the SME can be leveraged as a generic SMOS, handling service management and exposure for any SMOSs within SMO. The SME SMOS provides a set of common SMO capabilities that are needed for all SMO Services.

Several of the SME services are applicable to any SMOSs:

* SMOS registration,
* SMOS discovery,
* Authentication and authorization to access a SMOS,
* Communication support between SMOS Service Producers and SMOS Service Consumers (SMOS Communication),
* Bootstrap of new SMOSs (optional),
* Heartbeat of SMOSs (optional).

1. SME in the role of SME services Producer

* supports the functionality of OAuth 2.0 Authorization server,
* support authentication and authorization of an rApp to determine which R1 services an rApp can access,
* provide response of success or failure to access token request

1. rApp

* initiating the procedure to request access token,
* initiating service request with the available access token.

1. R1 Service Producer

* provider of an R1 service.

9.3.3.4 Solution evaluation

Key issue #6: Provisioning of service access based on access token

* + - 1. Potential security recommendations
* Use OAuth 2.0, as specified in O-RAN Security Protocols Specifications [3], clause 4.7 to implement authorization in the R1 interface.
* Enable the SME to act as both an OAuth 2.0 resource and an OAuth 2.0 Authorization server.

## 9.3.4 Solution #4: Secure transmission of request and response between client to authorization server, between client and resource server for R1, A1 and O2

9.3.4.1 Background

This solution elaborates on the possibilities to ensure that data for the below flows is secure during transmission,

1. Between client and authorization server
   1. The client needs to send the credentials and the authorization request details which need to be secured during transmission.
2. Between client and resource server
   1. The client needs to send the access token to the authentic resource server which needs to be secured during transmission.

9.3.4.2 Solution description

Table 9.3.4-1 use case of secure data transmission between client and authorization server

|  |  |  |
| --- | --- | --- |
| Use Case Stage | Evolution/Specification | <<Uses>>  Related use |
| Goal | 1. Transfer the data from client to authorization server securely |  |
| Actors and Roles | Below are examples of HTTP clients   * rApp in the role of R1 Service Consumer * SMO towards near RT RIC in the role of A1 consumer * O-RAN assets towards O-cloud in the role of O2 consumer |  |
| Assumptions | TLS is the underlying secure protocol on which the below is built upon. |  |
| Preconditions | The consumer is registered with valid client\_id and client\_secret  The authorization server has a valid X.509 certificate  The consumer has the trust anchor for the authorization server’s X.509 certificate |  |
| Begins when | The consumer needs to contact the authorization server to send an authorization request |  |
| Step 1 (M) | The client authenticates to the authorization server |  |
| Step 1 (M) | The client sends the request details to authorization server after successful authentication |  |
| Ends when | The consumer receives the access token from the authorization server |  |
| Exceptions | n/a |  |
| Post Conditions | The consumer is in possession of the access token securely |  |
| Traceability | n/a |  |

@startuml

hide footbox

skinparam sequenceActorBackgroundColor White

skinparam sequenceActorBorderColor Black

skinparam sequenceParticipantBackgroundColor White

skinparam sequenceParticipantBorderColor Black

skinparam sequenceArrowColor Black

skinparam sequenceLifeLineBackgroundColor White

skinparam sequenceLifeLineBorderColor Black

skinparam shadowing false

skinparam NoteBackgroundColor White

skinparam NoteBorderColor Black

Title Secure transmission between client and authorization server

Actor "client" as client

participant "OAuth Authorization Server" as AS

client -> AS:authenticate with client\_id and client\_secret

client -> AS:send authorization Request

AS -> client: send access token

@enduml

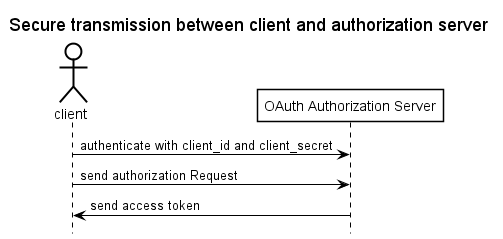


Figure 9.3.4-1 Transmission between client and authorization server

@startuml

hide footbox

skinparam sequenceActorBackgroundColor White

skinparam sequenceActorBorderColor Black

skinparam sequenceParticipantBackgroundColor White

skinparam sequenceParticipantBorderColor Black

skinparam sequenceArrowColor Black

skinparam sequenceLifeLineBackgroundColor White

skinparam sequenceLifeLineBorderColor Black

skinparam shadowing false

skinparam NoteBackgroundColor White

skinparam NoteBorderColor Black

Title Secure transmission between client and resource server

Actor "client" as client

participant "resource server" as RS

client <-> RS:mutual authentication

client -> RS:send access token as part of API operation

@enduml

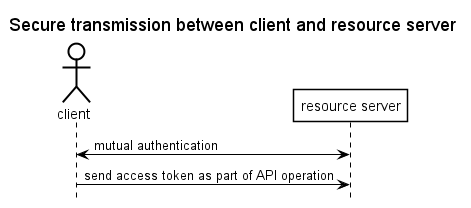


Figure 9.3.4-2 Transmission between client and resource server

9.3.4.2.1 Authentication methods between client, authorization server and resource server

The solutions below are various options to transfer the information securely between the O-RAN OAuth clients and the various servers in the interaction including the authorization server and the various O-RAN resource servers.

Below is a summary:

Table 9.3.4-2 Authentication methods

|  |  |  |
| --- | --- | --- |
| OAuth token end point authentication method | Reference | Short description |
| Tls\_client\_auth | RFC 8705[i.6] | Usage of certificates for both client and server authentication. Usage of certificate hash for certificate bound access tokens |
| private\_key\_JWT (with JWE) | RFC 8705[i.6] | Usage of certificates for both integrity and confidentiality protection(encryption) |
| private\_key\_jwt | OpenID foundation [i.21] | Usage of certificates for integrity protection |
| Client\_secret\_jwt | OpenID foundation [i.21] | Usage of shared secret for example client\_secret for integrity protection |
| Client\_secret\_basic | RFC 7591[i.22] | Client\_id and client\_secret transmission in the header |
| Client\_secret\_post | RFC 7591[i.22] | Client\_id and client\_secret transmission in the payload |
| Self\_signed\_tls\_client\_auth | RFC 8705[i.6] | Usage of client side self signed certificates in mutual authentication |

Below are the details of the methods explained above

1. Tls\_client\_auth
   1. Part of RFC 8705.
   2. Mutual authentication of both client and server. Here certificates are issued by a certificate authority. Also supports token binding.
   3. For certificate bound access tokens every party proves that they are in possession of their private key.
   4. Who is in possession of the private key used to establish the connection can use the access token (puts the hash of the certificate(cnf) that has been used).

A computer screen shot of a computer

Description automatically generated

Figure 9.3.4-3 - tls\_client\_oauth method

1. private\_key\_jwt(with JWE)
   1. Part of OpenID foundation specification.
   2. Public and private key is used to sign and encrypt the JWT token using asymmetric cryptography.
   3. Contents of the message cannot be seen as they are encrypted.

A screen shot of a computer

Description automatically generated

Figure 9.3.4-4 Private\_key\_jwt with JWE method

1. private\_key\_jwt
   1. Part of OpenID foundation specification.
   2. The client\_id and client\_secret is part of the JWT token which is integrity protected. The JWT is a string which is attached as a client assertion. Here we AVOID putting the secret in message, instead in the JWT token.
   3. Confidential client create JSON Web Token along with the public and private key. Hence structure of the JWT is the header, body and the Signature. Hence JWS client has the private key and shares the public key to the authorization server for validation of the request**.** Hence the contents of the message can be seen but secure from manipulation.

A screenshot of a computer

Description automatically generated

Figure 9.3.4-5 Private\_key\_jwt method

1. Client\_secret\_jwt
   1. Part of OpenID foundation specification.
   2. Client\_secret\_JWT - client\_id and client\_secret in the JWT token which is integrity protected with a shared secret key (client\_secret known to both parties) using HMAC. The JWT is a string which is attached as a client assertion. Here we AVOID putting the secret in message. Instead in the JWT token which is with a MAC (hash function+Secret).

A screenshot of a computer

Description automatically generated

Figure 9.3.4-6 Client\_secret\_jwt method

1. Client\_secret\_basic
   1. client\_id and client\_secret in the header of the message.
   2. Defined in RFC 7591.

A screenshot of a computer

Description automatically generated

Figure 9.3.4-7 Client\_secret\_basic method

1. Client\_secret\_post
   1. client\_id and client\_secret in the body of the message.
   2. Defined in RFC 7591.

A screenshot of a computer

Description automatically generated

Figure 9.3.4-8 Client\_secret\_post method

1. Self\_signed\_tls\_client\_auth
   1. RFC 8705.
   2. Mutual authentication using self signed certificates. Here certificate is created by self signing.

A screenshot of a computer

Description automatically generated

Figure 9.3.4-9 Self\_signed\_tls\_client\_auth method

9.3.4.3 Impacts on existing O-RAN architecture elements and interfaces

R1 client, A1 client and the O2 client needs to choose the authentication method from the options explained above depending on the client and authorization server capabilities:

9.3.4.4 Solution evaluation

There is a need to support secure transmission of request and response between client and authorization server, between client to resource server. Below is the order of preference of various methods explained above in the order of most preferred to least preferred depending on the authorization server capabilities as well as the security level preferred.

Table 9.3.4-3 Authentication method preferences

|  |  |
| --- | --- |
| OAuth token end point authentication method | Evaluation |
| Tls\_client\_auth | Highly preferred |
| private\_key\_JWT(with JWE) | Preferred |
| private\_key\_jwt | Preferred |
| Client\_secret\_jwt | Preferred |
| Client\_secret\_basic | Preferred |
| Client\_secret\_post | Preferred |
| Self\_signed\_tls\_client\_auth | Least preferred |

# 10 Study and Recommendation an O-RAN Authorization Server Architecture

Editor’s note: This section will do a study and recommendation on an authorization server architecture to overlay on the O-RAN architecture.

# 11 Conclusion

Editor’s note : This clause will provide the summary for the TR and mapping tables the key issues and solutions and it will submit the recommendations to WG1 for O-RAN Authorization server.

Annex:   
Change history/Change request (history)

|  |  |  |
| --- | --- | --- |
| Date | Revision | Description |
| 2023.05.10 | 00.00.01 | Initial TS skeleton |
| 2023.08.23 | 00.00.02 | Updated the scope and reference for the TR |
| 2023.10.16 | 00.00.03 | Updated the TR based on CR approved in OAuth 2.0 meeting for Nov train |
| 2023.10.24 | 00.00.04 | Updated the TR based on CR approved during O-RAN F2F phoenix meeting |
| 2023.11.21 | 01.00 | Final version 01.00 |
| 2024.02.07 | 01.00.01 | Updated the TR based on CR approved for OAuth 2.0 TR WI meeting in Dec 2023 and Jan 2024 |
| 2024.03.05 | 01.00.02 | Updated the TR based on CR approved for OAuth 2.0 TR WI meeting in Feb 2024 |
| 2024.06.26 | 02.00.01 | Updated the TR based on CR approved for OAuth 2.0 TR WI meeting from April to June 2024 |
| 2024.07.05 | 02.00.02 | Updated the TR based on the review comments during July 2024 plenary approvals |
| 2024.07.26 | 02.00.03 | Updated the TR based on the review comments during July 2024 WG11 voting |