OAuth2 Client Authorization Support to NRF

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Products or Functional Area	AMF
Applicable Platforms	SMI
Feature Default Setting	Disabled – Configuration required to enable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	2022.04.0

Feature Description

This feature describes the authorization controls that are required for implementing all the network functions. The OAuth2 client authorization to NRF supports requesting AccessToken to NRF and validating AccessToken in the incoming requests.

All the network functions in the 5G core interact with each other using REST APIs. All these APIs are accessed without any authorization. These interactions between multiple network functions are processed without any access control mechanism. As per 3GPP standards, OAuth2 standards must be implemented for all the network functions to secure the access of REST APIs between these network functions.

As a mandatory security measure and as an upgraded requirement to protect the network function accessibility, the REST APIs are now accessed with an enhanced authorization mechanism. using OAuth2 standards.

Relationships

The network function of AMF can take two different roles, as the following:

- **NF Producer:** When a peer NF tries to access some of the service at AMF, it acts as NF producer.

  For more information, see AMF as NF Producer, on page 2.

- **NF Consumer:** When AMF as a client invokes a service at a peer NF, it acts as NF consumer.

  For more information, see AMF as NF Consumer, on page 4.

AMF as NF Producer

The AMF as an NF producer provides multiple services to the peer NFs. The following scenarios are processed for a successful communication to be established:

- The AMF must get registered with the NRF for a successful communication.
- Based on the configuration, the AMF Rest-EP must send the `oauth2Required` field set in the NF Service profile in the NF registration toward the NRF.
- The signing algorithm used to encrypt the token at the NRF must be configured using `access-token-jws-algo`. Currently, the following three algorithms are supported:
  - HS256—Where the shared secret key is provided at AMF.
  - RS256—Where private or public key is provided depending on the configuration at NRF.
  - ES256—Where private or public key is provided depending on the configuration at NRF.
- The AMF token validation is done by using a shared secret key or public key. It can be configured using `access-token-jws-key`.

How it Works

This section describes how this role works.

The AMF supports AccessToken validation in the incoming request. It is processed as in the following procedures:

- If an OAuth2 token is present in an incoming request from an NF consumer (such as SMF, UDM, peer AMF, and others), the AMF as an NF producer validates the token that is received in the incoming request.
- The signing algorithm used to encrypt the token at NRF can be accessed from `access-token-jws-algo`, and the respective shared secret key or public key can be accessed using `access-token-jws-key`.
- The AMF rejects an API request without the AccessToken or an API request with an invalid AccessToken. It returns the `status code 401` together with the www-authenticate header, with an error note as `invalid_token`.
- The AMF rejects an API request with an AccessToken validation token, for not having the required scopes to invoke the service operation. It returns the `status code 403` together with the www-authenticate header, with an error note as `insufficient_scope`.

Limitations

This feature has the following limitations:

- The AMF does not support the CLI changes for the NF-producer over the fly. They must be configured before the pod start-up.
- The AMF does not support the NSSF selection through NRF discovery and the access token will not be sent for NSSF.

Feature Configuration

To configure this feature, use the following configuration:

Note This configuration must be enabled in `amf-services` to register the AMF with NRF for the enablement of OAuth2 client authorization.

```
amf-services service_name
    amf-name amf_name
    locality locality_name
    oauth2-enabled
    access-token-jws-algo { HS256 | ES256 | RS256 }
    access-token-jws-key { shared_secret_key | public_key }
    exit
```

NOTES:

- **amf-services** *service_name*—Specify the name of the AMF service.
- **amf-name** *amf_name*—Specify the name of AMF.
- **locality** *locality_name*—Specify a name for the locality.
- **oauth2-enabled**—Enable the OAuth2 client authorization to register the AMF with NRF. The default value is false.

- **access-token-jws-algo { HS256 | ES256 | RS256 }**—Specify the type of the access token for the JWS Algorithm authorization.
- **access-token-jws-key { shared_secret_key | public_key }**—Specify the type of the access token for the JWS Key authorization.

Note When the `OAuth2-enabled` feature is configured, the options **access-token-jws-algo** and **access-token-jws-key** are mandatory.

Configuration Example

The following is an example configuration.

```
amf-services am1
 amf-name    amf1
 validate-Tais    false
 locality    LOC1
 oauth2-enabled
 access-token-jws-algo    { HS256 }
 access-token-jws-key    { public key }
 operator-policy-name    pem-file
 guamis mcc 123 mnc 456 region-id 1 set-id 14 pointer 3
 tai-groups test1
 exit
```

Configuration Verification

To verify the configuration:

```
show running-config amf-service am1
```

AMF as NF Consumer

The AMF as an NF consumer uses multiple services offered by the peer NFs.

The following scenarios are processed for a successful communication to be established:

- The AMF as a consumer looking for OAuth2-enabled profiles from NRF, must enable the AccessToken that is required in each `profile nf-client nf-type` configuration.
- The AMF as a consumer communicates with any of the applicable `nf-types`, such as AMF, PCF, UDM, AUSF, SMF, and SMSF.
- The AMF as a consumer sends the `Nnrf_AccessToken` request to the NRF server based on the `nf-client` configuration.
- The AMF as a consumer sends a request with an AccessToken to `nf-producer`. If it gets rejected due to the AccessToken validation failure, then the AMF failure handling template (FHT) handles those responses appropriately.

How it Works

This section describes how this role works. The AMF supports OAuth2 client authorization to NRF. This process gets executed with the following procedures:

- Only when the `nf-client` profile gets configured with `OAuth2-Enabled`, where the value gets set as true for a `nf-type`, the AMF considers those profiles with `OAuth2-Enabled` as true value.
- The AMF internally sends the AccessToken request to the NRF server, stores the received token in the cache. The same token gets reused until it expires.
- When the profile gets selected and the token also received, the application includes the AccessToken in the Authorization header in the request toward NF producer.
- If the `nf-client` profile doesn't get configured, that's when OAuth2 gets disabled on the consumer side. The AMF ignores those profiles with the `oauth2Required` and selects the producer among the rest of the profiles received in the discovery response.
- For AMF to send an AccessToken request to NRF, endpoints must get configured in the CLI for service type OAuth2 and the same must be set in the profile `nf-pair` for each type, wherever OAuth2 already enabled.
- When the `OAuth2-Enabled` gets set as true in the CLI and none of the discovered profiles from NRF has `oauth2Required`, then no profiles from the discovery get selected. It then reverts to the locally configured profiles. The AccessToken requests not sent as a locally configured profile, as it gets assumed as a base for the local trust policy. The NRF has no information about this development.
- When the `OAuth2-Enabled` gets set to false status in the CLI and all the discovered profiles get `oauth2Required` enabled, then none of these profiles in the discovery get selected. It then reverts to the locally configured profiles. If none of these profiles get configured locally, then the call fails.
- During the traffic running with the OAuth feature enabled, minimal numbers of 401-Unauthorized errors could be seen on the AMF side. To mitigate this risk, you can configure the failure handling template for all the possible causes (such as 401 error codes) to avoid any failed scenario of an end-to-end call.

Feature Configuration

To configure this feature, use the following configuration:

The following configuration is enabled only when the AMF sends the `Nnrf_AccessToken` request to the NRF server, when the `nf-client` is configured.

```
profile nf-client nf-type nf_type_name
    oauthenabled { true | false }
    nf-type-profile nf_type_profile_name
    locality locality_name
    priority priority_number
    service name type service_name type_npcf_am_policy_control
    endpoint-profile endpoint_profile_details
    capacity capacity_number
    uri-scheme http
    endpoint-name endpoint_name
    priority priority_number
    primary ip-address ipv4 ipv4_address
```

```
    primary ip-address port port_address
    exit
```

The following configuration must be done for an NRF endpoint, to which the AMF will send the AccessToken request.

```
group nrf auth nrf_group_name
    service type nrf oauth2
        endpoint-profile endpoint_profile_details
        capacity capacity_number
        uri-scheme http
        endpoint-name endpoint_name
        priority priority_number
        primary ip-address ipv4 ipv4_address
        primary ip-address port port_address
        exit
```

The following configuration must be used to specify auth-groups containing the NRF endpoint details for each NF type.

```
profile nf-pair nf-type nf_type_name
    nrf-auth-group nrf_auth_group_name
    nrf-discovery-group nrf_discovery_group_name
    locality client client_name
    locality preferred-server server_name
    locality geo-server geo_server_name
    cache invalidation { true | false } timeout timeout_number
    exit
```

NOTES:

- **profile nf-client nf-type** *nf_type_name*—Specify the NF and the profile name.
- **oauthenabled { true | false }**—Enable the oauthenabled profile configuration. The default value is false.
- **nf-type-profile** *nf_type_profile_name*—Specify the NF profile name.
- **locality** *locality_name*—Specify the locality.
- **priority** *priority_number*—Specify the priority request. Must be in numbers.
- **service name type** *service_name type_npcf_am_policy_control*—Specify the service name and the type.
- **endpoint-profile** *endpoint_profile_details*—Specify the endpoint profile details.
- **capacity** *capacity_number*—Specify the capacity requirement in number.
- **uri-scheme http**—Specify the URI scheme.
- **endpoint-name** *endpoint_name*—Specify the endpoint name.
- **primary ip-address ipv4** *ipv4_address*—Specify the primary IPv4 address.
- **primary ip-address port** *port_address*—Specify the primary port address.
- **group nrf auth** *nrf_group_name*—Specify the NRF group name to authenticate. Must be a string.
- **service type nrf oauth2**—Specify the service and the type of NRF, which must be authenticated to enable the OAuth2 profile configuration.

- **profile nf-pair nf-type** *nf_type_name*—Specify the nf-type in the profile name to authenticate. Must be a string.
- **nrf-auth-group** *nrf_auth_group_name*—Specify the nrf-auth-group name.
- **nrf-discovery-group** *nrf_discovery_group_name*—Specify the nrf-discovery-group name.
- **locality client** *client_name*—Specify the client name in the locality details.
- **locality preferred-server** *server_name*—Specify the preferred-server or client name in the locality details.
- **locality geo-server** *geo_server_name*—Specify the geo-server name in the locality details.
- **cache invalidation { true | false }**—Enable the cache invalidation configuration. The default value is false.
- **timeout** *timeout_number*—Specify the timeout duration in seconds.

Configuration Example

The following is an example configuration.

```
profile nf-client nf-type pcf
 oauthenabled { true }
 pcf-profile PP1
  locality LOC1
   priority 30
   service name type npcf-am-policy-control
    endpoint-profile EP1
     capacity   30
     uri-scheme http
     endpoint-name EP1
      priority 56
      primary ip-address ipv4 209.165.201.30
      primary ip-address port 9049
     exit
    exit
   exit
  exit
 exit
exit
group nrf auth oauthep
  service type nrf oauth2
    endpoint-profile EP1
      capacity 30
      uri-scheme http
      endpoint-name EP1
        priority 56
        primary ip-address ipv4 209.165.201.30
        primary ip-address port 9049
      exit
    exit
  exit
exit
profile nf-pair nf-type PCF
 nrf-auth-group      oauthep
 nrf-discovery-group udmdiscovery
 locality client  LOC1
 locality preferred-server LOC1
 locality geo-server GEO
```

```
 cache invalidation true timeout 1000
exit
```

Configuration Verification

To verify the configuration:

```
show profile nf-client nf-type pcf
```

OAM Support

This section describes operations, administration, and maintenance support for this feature.

Bulk Statistics Support

The following statistics are supported for the OAuth2 Client Authorization Support to NRF, on page 1 feature.

NF AccessToken Statistics

Total AccessToken Message count:

- Labels:

  nf_oauth_messages_total

- Host:

  Refers to the host information (IP: Port) of the URL to which the access token request is being sent.

- Service Name: NF AccessToken service

  Example: **nnrf-oauth**

- Version: API version

  Example: **v1**

- NfType:

  peer nf type

- GrInstanceID:

  gr-instance-id

- Result:

  [Success | error response status code | RPC/Timeout error | Request parse failure | Response parse failure | Invalid notification event | Invalid Nf instance URI | Internal Error]

Token Validation Statistics for AMF as Producer

App infra statistics for outgoing responses:

- Labels: **outgoing_response_total**

- Invalid token format or signature mismatch:

```
{app_name="AMF", protocol="http", service_name="amf-rest-ep",
status="error", status_code="invalid_token"}
```

- Token payload verification fail:
  - Invalid scope:

```
{app_name="AMF", protocol="http", service_name="amf-rest-ep",
status="error", status_code="app_invalid_scope"}
```

  - Other IE failures:

```
{app_name="AMF", protocol="http", service_name="amf-rest-ep",
status="error", status_code="app_invalid_token"}
```

For more information on bulk statistics support for AMF, see the *UCC 5G AMF Metrics Reference* document.

Data Type Support

The following statistics are supported for the OAuth2 Client Authorization Support to NRF, on page 1 feature.

AccessTokenReq

- `grant_type`
- `nfInstanceId`
- `nfType`
- `targetNfType`
- `scope`

For information on data type support for AMF, see the *UCC 5G AMF Metrics Reference* document.